ACL 2017

The 55th Annual Meeting of the Association for Computational Linguistics

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

July 30 - August 4, 2017
Vancouver, Canada

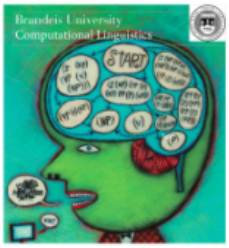
Platinum Sponsors:

Gold Sponsors:

Silver Sponsors:

Bronze Sponsors:

Supporters:



Order copies of this and other ACL proceedings from:

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

ISBN 978-1-945626-75-3 (Volume 1)
ISBN 978-1-945626-76-0 (Volume 2)

Preface: General Chair

Welcome to ACL 2017 in Vancouver, Canada! This is the 55th annual meeting of the Association for Computational Linguistics. A tremendous amount of knowledge has been presented at more than half a century's worth of our conferences. Hopefully, some of it is still relevant now that deep learning has solved language. We are anticipating one of the largest ACL conferences ever. We had a record number of papers submitted to the conference, and a record number of industry partners joining us as sponsors of the conference. We are on track to be one of the best attended ACL conferences to date. I hope that this year's conference is intellectually stimulating and that you take home many new ideas and techniques that will help extend your own research.

Each year, the ACL conference is organized by a dedicated team of volunteers. Please thank this year's organizers for their service to the community when you see them at the conference. Without these people, this conference would not happen: Regina Barzilay and Min-Yen Kan (Program Co-Chairs), Priscilla Rasmussen and Anoop Sarkar (Local Organizing Committee), Wei Xu and Jonathan Berant (Workshop Chairs), Maja Popović and Jordan Boyd-Graber (Tutorial Chairs), Wei Lu, Sameer Singh and Margaret Mitchell (Publication Chairs), Heng Ji and Mohit Bansal (Demonstration Chairs), Spandana Gella, Allyson Ettinger, and Matthieu Labeau (Student Research Workshop Organizers), Cecilia Ovesdotter Alm, Mark Dredze, and Marine Carpuat (Faculty Advisors to the Student Research Workshop), Charley Chan (Publicity Chair), Christian Federmann (Conference Handbook Chair), Maryam Siahbani (Student Volunteer Coordinator), and Nitin Madnani (Webmaster and Appmaster).

The organizers have been working for more than a year to put together the conference. Far more than a year in advance, the ACL 2017 Coordinating Committee helped to select the venue and to pick the General Chair and the Program Co-Chairs. This consisted of members from NAACL and ACL executive boards. Representing NAACL we had Hal Daumé III, Michael White, Joel Tetreault, and Emily Bender. Representing ACL we had Pushpak Bhattacharyya, Dragomir Radev, Graeme Hirst, Yejin Choi, and Priscilla Rasmussen. I would like to extend a personal thanks to Graeme and Priscilla who often serve as the ACL's institutional memory, and who have helped fill in many details along the way.

I would like to extend a special thanks to our Program Co-Chairs, Regina Barzilay and Min-Yen Kan. They documented their work creating the program by running a blog. They used their blog as a platform for engaging the ACL community in many of the decision making processes including soliciting suggestions for the conference's area chairs and invited speakers. They hosted discussions with Marti Hearst and Joakim Nivre about the value of publishing pre-prints of submitted paper on arXiv and how they relate to double blind reviewing. They even invited several prominent members of our community to provide last-minute writing advice. If you weren't following the blog in the lead-up to the conference, I highly recommend taking a look through it now. You can find it linked from the ACL 2017 web page.

This year's program looks like it will be excellent! We owe a huge thank you to Regina Barzilay and Min-Yen Kan. They selected this year's papers from 1,318 submissions with the help of 44 area chairs and more than 1,200 reviewers. Thanks to Regina, Min, the area chairs, the reviewers and the authors. Beyond the papers, we have talks by luminaries in the field of NLP, including ACL President Joakim Nivre, invited speakers Mirella Lapata and Noah Smith, and the recipient of this year's Lifetime Achievement Award. We also have an excellent set of workshops and tutorials. On the tutorial day, there will also be a special workshop on Women and Underrepresented Minorities in Natural Language Processing. Thank you to our workshop organizers and tutorial presenters.

This year's conference features two outreach activities that I would like to highlight. First, on Sunday, July 30, 2017, there will be a workshop on Women and Underrepresented Minorities in Natural Language Processing organized by Libby Barak, Isabelle Augenstein, Chloé Braud, He He, and Margaret Mitchell. The goals of the workshop are to increase awareness of the work women and underrepresented

groups do, support women and underrepresented groups in continuing to pursue their research, and motivate long-term resources for underrepresented groups within ACL. Second, for the first time ever, ACL is offering subsidized on-site childcare at the conference hotel. The goal of this is to allow ACL participants with children to more readily be able to attend the conference. Since childcare duties often fall disproportionately on women, our hope is that by having professional childcare on-site that we will allow more women to participate, and therefore to help promote their careers. My hope is that the childcare will be continued in future conferences.

I would like to thank our many sponsors for their generous contributions. Our platinum sponsors are Alibaba, Amazon, Apple, Baidu, Bloomberg, Facebook, Google, Samsung and Tencent. Our gold sponsors are eBay, Elsevier, IBM Research, KPMG, Maluuba, Microsoft, Naver Line, NEC, Recruit Institute of Technology, and SAP. Our silver sponsors are Adobe, Bosch, CVTE, Duolingo, Huawei, Nuance, Oracle, and Sogou. Our bronze sponsors are Grammarly, Toutiao, and Yandex. Our supporters include Newsela and four professional master's degree programs from Brandeis, Columbia, NYU and the University of Washington. We would like to acknowledge the generous support of the National Science Foundation which has awarded a $15,000 grant to the ACL Student Research Workshop. Finally, NVIDIA donated several Titan X GPU cards for us to raffle off during the conference.

Lastly, I would like to thank everyone else who helped to make this conference a success. Thank you to our area chairs, our army of reviewers, our workshop organizers, our tutorial presenters, our invited speakers, and our authors. Best regards to all of you.

Welcome to ACL 2017!

Chris Callison-Burch
General Chair

Preface: Program Committee Co-Chairs

Welcome to the 55th Annual Meeting of the Association for Computational Linguistics! This year, ACL received 751 long paper submissions and 567 short paper submissions[1]. Of the long papers, 195 were accepted for presentation at ACL — 117 as oral presentations and 78 as poster presentations (25% acceptance rate). 107 short papers were accepted — 34 as oral and 73 as poster presentations (acceptance rate of 18%). In addition, ACL will also feature 21 presentations of papers accepted in the *Transactions of the Association for Computational Linguistics* (TACL). Including the student research workshop and software demonstrations, the ACL program swells to a massive total of 367 paper presentations on the scientific program, representing the largest ACL program to date.

ACL 2017 will have two distinguished invited speakers: Noah A. Smith (Associate Professor of Computer Science and Engineering at the University of Washington) and Mirella Lapata (Professor in the School of Informatics at the University of Edinburgh). Both are well-renowned for their contributions to the field of computational linguistics and are excellent orators. We are honored that they have accepted our invitation to address the membership at this exciting juncture in our field's history, addressing key issues in representation learning and multimodal machine translation.

To manage the tremendous growth of our field, we introduced some changes to the conference. With the rotation of the annual meeting to the Americas, we anticipated a heavy load of submissions and early on we decided to have both the long and short paper deadlines merged to reduce reviewing load and to force authors to take a stand on their submissions' format. The joint deadline allowed us to only load our reviewers once, and also enabled us to have an extended period for more lengthy dialogue among authors, reviewers and area chairs.

In addition, oral presentations were shortened to fourteen (twelve) minutes for long (short) papers, plus time for questions. While this places a greater demand on speakers to be concise, we believe it is worth the effort, allowing far more work to be presented orally. We also took advantage of the many halls available and expanded the number of parallel talks to five during most of the conference sessions.

In keeping with changes introduced in the ACL community from last year, we continued the practice of recognizing outstanding papers at ACL. The 22 outstanding papers (15 long, 7 short, 1.6% of submissions) represent a broad spectrum of exciting contributions and have been specially placed on the final day of the main conference where the program is focused into two parallel sessions of these outstanding contributions. From these, a best paper and a best short paper those will be announced in the awards session on Wednesday afternoon.

Chris has already mentioned our introduction of the chairs' blog[2], where we strove to make the selection process of the internal workings of the scientific committee more transparent. We have publicly documented our calls for area chairs, reviewers and accepted papers selection process. Via the blog, we communicated several innovations in the conference organization workflow, of which we would call attention to two key ones here.

In the review process, we pioneered the use of the Toronto Paper Matching System, a topic model based approach to the assignment of reviewers to papers. We hope this decision will spur other program chairs to adopt the system, as increased coverage will better the reviewer/submission matching process, ultimately leading to a higher quality program.

For posterity, we also introduced the usage of hyperlinks in the bibliography reference sections of papers,

[1] These numbers exclude papers that were not reviewed due to formatting, anonymity, or double submission violations or that were withdrawn prior to review, which was unfortunately a substantial number.

[2] https://chairs-blog.acl2017.org/

and have worked with the ACL Anthology to ensure that digital object identifiers (DOIs) appear in the footer of each paper. These steps will help broaden the long-term impact of the work that our community has on the scientific world at large.

There are many individuals we wish to thank for their contributions to ACL 2017, some multiple times:

- The 61 area chairs who volunteered for our extra duty. They recruited reviewers, led discussions on each paper, replied to authors' direct comments to them and carefully assessed each submission. Their input was instrumental in guiding the final decisions on papers and selecting the outstanding papers.
- Our full program committee of BUG hard-working individuals who reviewed the conference's 1,318 submissions (including secondary reviewers).
- TACL editors-in-chief Mark Johnson, Lillian Lee, and Kristina Toutanova, for coordinating with us on TACL presentations at ACL.
- Noah Smith and Katrin Erk, program co-chairs of ACL 2016 and Ani Nenkova and Owen Rambow, program co-chairs of NAACL 2016, who we consulted several times on short order for help and advice.
- Wei Lu and Sameer Singh, our well-organized publication chairs, with direction and oversight from publication chair mentor Meg Mitchell. Also, Christian Federmann who helped with the local handbook.
- The responsive team at Softconf led by Rich Gerber, who worked quickly to resolve problems and who strove to integrate the use of the Toronto Paper Matching System (TPMS) for our use.
- Priscilla Rasmussen and Anoop Sarkar and the local organization team, especially webmaster Nitin Madnani.
- Christopher Calliston-Burch, our general chair, who kept us coordinated with the rest of the ACL 2017 team and helped us free our time to concentrate on the key duty of organizing the scientific program.
- Key-Sun Choi, Jing Jiang, Graham Neubig, Emily Pitler, and Bonnie Webber who carefully reviewed papers under consideration for best paper recognition.
- Our senior correspondents for the blog, who contributed guest posts and advice for writing and reviewing: Waleed Ammar, Yoav Artzi, Tim Baldwin, Marco Baroni, Claire Cardie, Xavier Carreras, Hal Daumé, Kevin Duh, Chris Dyer, Marti Hearst, Mirella Lapata, Emily M. Bender, Aurélien Max, Kathy McKeown, Ray Mooney, Ani Nenkova, Joakim Nivre, Philip Resnik, and Joel Tetreault. Without them, the participation of the community through the productive comments, and without you the readership, our blog for disseminating information about the decision processes would not have been possible and a success.

We hope that you enjoy ACL 2017 in Vancouver!

ACL 2017 program co-chairs
Regina Barzilay, Massachusetts Institute of Technology
Min-Yen Kan, National University of Singapore

Organizing Committee

General Chair:

Chris Callison-Burch, University of Pennsylvania

Program Co-Chairs:

Regina Barzilay, Massachusetts Institute of Technology
Min-Yen Kan, National University of Singapore

Local Organizing Committee:

Priscilla Rasmussen, ACL
Anoop Sarkar, Simon Fraser University

Workshop Chairs:

Wei Xu, Ohio State University
Jonathan Berant, Tel Aviv University

Tutorial Chairs:

Maja Popović, Humboldt-Universität zu Berlin
Jordan Boyd-Graber, University of Colorado, Boulder

Publication Chairs:

Wei Lu, Singapore University of Technology and Design
Sameer Singh, University of California, Irvine
Margaret Mitchell, Google (Advisory)

Demonstration Chairs:

Heng Ji, Rensselaer Polytechnic Institute
Mohit Bansal, University of North Carolina, Chapel Hill

Student Research Workshop Organizers:

Spandana Gella, University of Edinburgh
Allyson Ettinger, University of Maryland, College Park
Matthieu Labeau, Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur (LIMSI)

Faculty Advisors to the Student Research Workshop:

Cecilia Ovesdotter Alm, Rochester Institute of Technology
Mark Dredze, Johns Hopkins University
Marine Carpuat, University of Maryland, College Park

Publicity Chair:

Charley Chan, Bloomberg

Conference Handbook Chair:

Christian Federmann, Microsoft

Student Volunteer Coordinator:

Maryam Siahbani, University of the Fraser Valley

Webmaster and Appmaster:

Nitin Madnani, Educational Testing Service

Program Committee

Program Committee Co-Chairs

Regina Barzilay, Massachusetts Institute of Technology
Min-Yen Kan, National University of Singapore

Area Chairs

Mausam (Information Extraction and NLP Applications Area)
Omri Abend (Multilingual Area)
Eugene Agichtein (Information Extraction and NLP Applications Area)
Ron Artstein (Dialogue and Interactive Systems Area)
Alexandra Balahur (Sentiment Analysis and Opinion Mining Area)
Mohit Bansal (Vision, Robotics and Grounding Area)
Chia-Hui Chang (Information Extraction and NLP Applications Area)
Grzegorz Chrupała (Machine Learning Area)
Mona Diab (Multilingual Area)
Jason Eisner (Phonology, Morphology and Word Segmentation Area)
Manaal Faruqui (Semantics Area)
Raquel Fernandez (Dialogue and Interactive Systems Area)
Karën Fort (Multidisciplinary Area)
Amir Globerson (Machine Learning Area)
Hannaneh Hajishirzi (Semantics Area)
Chiori Hori (Speech Area)
Tommi Jaakkola (Machine Learning Area)
Yangfeng Ji (Discourse and Pragmatics Area)
Jing Jiang (Information Extraction and NLP Applications Area)
Sarvnaz Karimi (Information Extraction and NLP Applications Area)
Anna Korhonen (Semantics Area)
Zornitsa Kozareva (Information Extraction and NLP Applications Area)
Lun-Wei Ku (Sentiment Analysis and Opinion Mining Area)
Nate Kushman (Vision, Robotics and Grounding Area)
Chia-ying Lee (Speech Area)
Oliver Lemon (Dialogue and Interactive Systems Area)
Roger Levy (Cognitive Modeling and Psycholinguistics Area)
Sujian Li (Discourse and Pragmatics Area)
Wenjie Li (Summarization and Generation Area)
Kang Liu (Information Extraction and NLP Applications Area)
Tie-Yan Liu (Information Extraction and NLP Applications Area)
Yang Liu (Machine Translation Area)
Zhiyuan Liu (Social Media Area)
Minh-Thang Luong (Machine Translation Area)
Saif M Mohammad (Sentiment Analysis and Opinion Mining Area)
Alexander M Rush (Summarization and Generation Area)
Haitao Mi (Machine Translation Area)
Alessandro Moschitti (Information Extraction and NLP Applications Area)
Smaranda Muresan (Information Extraction and NLP Applications Area)
Preslav Nakov (Semantics Area)
Graham Neubig (Machine Translation Area)

Aurélie Névéol (Biomedical Area)
Shimei Pan (Social Media Area)
Michael Piotrowski (Multidisciplinary Area)
Emily Pitler (Tagging, Chunking, Syntax and Parsing Area)
Barbara Plank (Tagging, Chunking, Syntax and Parsing Area)
Sujith Ravi (Machine Learning Area)
Verena Rieser (Summarization and Generation Area)
Sophie Rosset (Resources and Evaluation Area)
Mehroosh Sadrzadeh (Semantics Area)
Hinrich Schütze (Phonology, Morphology and Word Segmentation Area)
Anders Søgaard (Cognitive Modeling and Psycholinguistics Area)
Karin Verspoor (Biomedical Area)
Aline Villavicencio (Semantics Area)
Svitlana Volkova (Social Media Area)
Bonnie Webber (Discourse and Pragmatics Area)
Deyi Xiong (Machine Translation Area)
William Yang Wang (Machine Learning Area)
Wajdi Zaghouani (Resources and Evaluation Area)
Yue Zhang (Tagging, Chunking, Syntax and Parsing Area)
Hai Zhao (Tagging, Chunking, Syntax and Parsing Area)

Primary Reviewers

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

Balamurali A R, Mourad Abbas, Omri Abend, Amjad Abu-Jbara, Gilles Adda, Heike Adel, Stergos Afantenos, Apoorv Agarwal, Eneko Agirre, Željko Agić, Alan Akbik, Ahmet Aker, Mohammed Alam, Hanan Aldarmaki, Enrique Alfonseca, Afra Alishahi, Laura Alonso Alemany, David Alvarez-Melis, Maxime Amblard, Maryam Aminian, Silvio Amir, Waleed Ammar, Daniel Andrade, Jacob Andreas*, Nicholas Andrews*, Ion Androutsopoulos, Galia Angelova, Jean-Yves Antoine*, Emilia Apostolova, Jun Araki, Yuki Arase, Lora Aroyo, Philip Arthur, Yoav Artzi*, Masayuki Asahara, Giuseppe Attardi, AiTi Aw, Ahmed Hassan Awadallah, Wilker Aziz

Collin Baker, Alexandra Balahur, Niranjan Balasubramanian, Timothy Baldwin, Tyler Baldwin, Miguel Ballesteros, David Bamman, Rafael E. Banchs, Carmen Banea, Ritwik Banerjee, Srinivas Bangalore, Libby Barak, Alistair Baron, Marco Baroni, Alberto Barrón-Cedeño, Roberto Basili, David Batista, Daniel Bauer, Timo Baumann, Daniel Beck, Srikanta Bedathur, Beata Beigman Klebanov, Kedar Bellare, Charley Beller, Islam Beltagy, Anja Belz, Yassine Benajiba, Fabrício Benevenuto, Luciana Benotti*, Jonathan Berant, Taylor Berg-Kirkpatrick, Sabine Bergler*, Robert Berwick, Laurent Besacier, Steven Bethard, Archna Bhatia, Chris Biemann, Joachim Bingel, Or Biran, Alexandra Birch, Arianna Bisazza, Yonatan Bisk, Prakhar Biyani, Johannes Bjerva, Anders Björkelund, Philippe Blache, Frédéric Blain, Eduardo Blanco, Nate Blaylock, Bernd Bohnet, Gemma Boleda, Danushka Bollegala, Claire Bonial, Francesca Bonin, Kalina Bontcheva, Benjamin Börschinger, Johan Bos, Elizabeth Boschee, Florian Boudin, Fethi Bougares, Samuel Bowman, Johan Boye, Kristy Boyer, Cem Bozsahin, David Bracewell, S.R.K. Branavan, Pavel Braslavski, Adrian Brasoveanu, Ted Briscoe, Chris Brockett, Julian Brooke, Elia Bruni, William Bryce, Marco Büchler, Christian Buck, Paul Buitelaar, Harry Bunt, Manuel Burghardt, David Burkett, Hendrik Buschmeier, Miriam Butt

José G. C. de Souza, Deng Cai, Jose Camacho-Collados, Berkant Barla Cambazoglu, Erik Cambria, Burcu Can, Marie Candito, Hailong Cao, Kris Cao*, Yuan Cao, Ziqiang Cao, Cornelia Caragea, Jesus Cardeńosa, Giuseppe Carenini, Marine Carpuat, Xavier Carreras, John Carroll, Paula Carvalho, Francisco Casacuberta, Helena Caseli, Tommaso Caselli*, Taylor Cassidy, Vittorio Castelli, Giuseppe Castellucci, Asli Celikyilmaz*, Daniel Cer, Özlem Çetinoğlu, Mauro Cettolo, Arun Chaganty, Joyce Chai, Soumen Chakrabarti, Gaël de Chalendar, Yllias Chali, Nathanael Chambers, Jane Chandlee, Muthu Kumar Chandrasekaran, Angel Chang*, Baobao Chang, Kai-Wei Chang, Ming-Wei Chang, Snigdha Chaturvedi, Wanxiang Che, Ciprian Chelba, Bin Chen, Boxing Chen, Chen Chen, Hsin-Hsi Chen, John Chen, Kehai Chen, Kuang-hua Chen, Qingcai Chen, Tao Chen, Wenliang Chen, Xinchi Chen, Yubo Chen, Yun-Nung Chen, Zhiyuan Chen, Jianpeng Cheng, Colin Cherry, Sean Chester, Jackie Chi Kit Cheung*, David Chiang, Jen-Tzung Chien, Hai Leong Chieu, Laura Chiticariu, Eunsol Choi, Kostadin Cholakov, Shamil Chollampatt, Jan Chorowski, Christos Christodoulopoulos, Tagyoung Chung, Kenneth Church, Mark Cieliebak*, Philipp Cimiano, Alina Maria Ciobanu*, Alexander Clark*, Jonathan Clark, Stephen Clark, Ann Clifton, Maximin Coavoux, Kevin Cohen, Nigel Collier, Michael Collins, Sandra Collovini, Miriam Connor, John Conroy*, Matthieu Constant, Danish Contractor, Mark Core, Ryan Cotterell, Benoit Crabbé, Danilo Croce*, Fabien Cromieres, Montse Cuadros, Heriberto Cuayahuitl, Silviu-Petru Cucerzan, Aron Culotta*

Luis Fernando D'Haro, Giovanni Da San Martino, Walter Daelemans, Daniel Dahlmeier, Amitava Das, Dipanjan Das, Rajarshi Das, Pradeep Dasigi, Johannes Daxenberger, Munmun De Choudhury, Eric De La Clergerie, Thierry Declerck, Luciano Del Corro, Louise Deléger, Felice Dell'Orletta, Claudio Delli Bovi, Li Deng, Lingjia Deng, Pascal Denis, Michael Denkowski, Tejaswini Deoskar, Leon Derczynski, Nina Dethlefs, Ann Devitt, Jacob Devlin, Lipika Dey, Barbara Di Eugenio, Giuseppe Di Fabbrizio, Gaël Dias, Fernando Diaz, Georgiana Dinu, Liviu P. Dinu, Stefanie Dipper, Dmitriy Dligach, Simon Dobnik, Ellen Dodge, Jesse Dodge, Daxiang Dong, Li Dong, Doug Downey, Gabriel Doyle, A. Seza Doğruöz, Eduard Dragut, Mark Dras*, Markus Dreyer, Lan Du, Nan Duan, Xiangyu Duan, Kevin Duh*, Long Duong, Emmanuel Dupoux, Nadir Durrani, Greg Durrett, Ondřej Dušek*, Marc Dymetman

Judith Eckle-Kohler, Steffen Eger*, Markus Egg, Koji Eguchi, Patrick Ehlen, Maud Ehrmann*, Vladimir Eidelman, Andreas Eisele, Jacob Eisenstein*, Heba Elfardy, Michael Elhadad*, Desmond Elliott*, Micha Elsner, Nikos Engonopoulos, Messina Enza, Katrin Erk, Arash Eshghi, Miquel Esplà-Gomis

James Fan, Federico Fancellu, Licheng Fang, Benamara Farah, Stefano Faralli, Richárd Farkas, Afsaneh Fazly, Geli Fei, Anna Feldman, Minwei Feng, Yansong Feng, Olivier Ferret, Oliver Ferschke, Simone Filice, Denis Filimonov, Katja Filippova*, Andrew Finch, Nicolas Fiorini, Orhan Firat*, Radu Florian, Evandro Fonseca, Markus Forsberg, Eric Fosler-Lussier, George Foster, James Foulds*, Marc Franco-Salvador, Alexander Fraser, Dayne Freitag, Lea Frermann, Annemarie Friedrich, Piotr W. Fuglewicz, Akinori Fujino, Fumiyo Fukumoto, Robert Futrelle

Robert Gaizauskas, Olivier Galibert*, Irina Galinskaya, Michel Galley*, Michael Gamon, Kuzman Ganchev, Siva Reddy Gangireddy, Jianfeng Gao, Claire Gardent*, Matt Gardner, Guillermo Garrido, Justin Garten, Milica Gasic, Eric Gaussier, Tao Ge, Georgi Georgiev, Kallirroi Georgila, Pablo Gervás*, George Giannakopoulos, C Lee Giles, Kevin Gimpel*, Maite Giménez*, Roxana Girju, Adrià de Gispert, Dimitra Gkatzia*, Goran Glavaš, Amir Globerson, Yoav Goldberg, Dan Goldwasser, Carlos Gómez-Rodríguez*, Graciela Gonzalez, Edgar Gonzàlez Pellicer, Kyle Gorman, Matthew R. Gormley, Isao Goto, Cyril Goutte, Amit Goyal, Kartik Goyal, Pawan Goyal, Joao Graca, Yvette Graham, Roger Granada, Stephan Greene, Jiatao Gu, Bruno Guillaume, Liane Guillou, Curry Guinn, Hongyu Guo, James Gung, Jiang Guo, Weiwei Guo, Yufan Guo, Yuhong Guo, Abhijeet Gupta, Rahul Gupta, Yoan Gutiérrez, Francisco Guzmán,

Thanh-Le Ha, Christian Hadiwinoto, Gholamreza Haffari, Matthias Hagen, Udo Hahn, Jörg Hakenberg, Dilek Hakkani-Tur, Keith Hall, William L. Hamilton, Michael Hammond, Xianpei Han, Sanda Harabagiu, Christian Hardmeier, Kazi Saidul Hasan, Sadid A. Hasan, Saša Hasan, Eva Hasler, Hany Hassan, Helen Hastie, Claudia Hauff, He He*, Hua He, Luheng He, Shizhu He, Xiaodong He, Yulan He, Peter Heeman, Carmen Heger, Serge Heiden, Georg Heigold, Michael Heilman, James Henderson, Matthew Henderson, Aron Henriksson, Aurélie Herbelot*, Ulf Hermjakob, Daniel Hershcovich, Jack Hessel, Kristina Hettne, Felix Hieber, Ryuichiro Higashinaka, Erhard Hinrichs, Tsutomu Hirao, Keikichi Hirose, Julian Hitschler, Cong Duy Vu Hoang, Julia Hockenmaier, Kai Hong*, Yu Hong, Ales Horak, Andrea Horbach, Takaaki Hori, Yufang Hou, Julian Hough, Dirk Hovy*, Eduard Hovy, Chun-Nan Hsu, Baotian Hu, Yuening Hu, Yuheng Hu, Hen-Hsen Huang, Hongzhao Huang, Liang Huang, Lifu Huang, Minlie Huang, Ruihong Huang, Songfang Huang, Xuanjing Huang, Yi-Ting Huang, Luwen Huangfu, Mans Hulden, Tim Hunter, Seung-won Hwang

Ignacio Iacobacci, Nancy Ide, Marco Idiart, Gonzalo Iglesias, Ryu Iida, Kenji Imamura, Diana Inkpen, Naoya Inoue, Hitoshi Isahara, Mohit Iyyer

Tommi Jaakkola, Cassandra L. Jacobs, Guillaume Jacquet, Evan Jaffe, Jagadeesh Jagarlamudi, Siddharth Jain, Aren Jansen, Sujay Kumar Jauhar, Laura Jehl, Minwoo Jeong, Yacine Jernite, Rahul Jha, Donghong Ji, Guoliang Ji, Sittichai Jiampojamarn, Hui Jiang, Antonio Jimeno Yepes, Salud María Jiménez-Zafra, Richard Johansson, Kyle Johnson, Melvin Johnson Premkumar, Kristiina Jokinen, Arne Jonsson, Aditya Joshi, Mahesh Joshi, Shafiq Joty, Dan Jurafsky*, David Jurgens

Besim Kabashi, Ákos Kádár, Sylvain Kahane*, Juliette Kahn, Herman Kamper, Jaap Kamps, Hiroshi Kanayama, Hung-Yu Kao, Justine Kao, Mladen Karan, Dimitri Kartsaklis, Arzoo Katiyar, David Kauchak, Daisuke Kawahara, Anna Kazantseva, Hideto Kazawa, Andrew Kehler, Simon Keizer, Frank Keller, Casey Kennington, Mitesh M. Khapra, Douwe Kiela, Halil Kilicoglu*, Jin-Dong Kim, Jooyeon Kim, Seokhwan Kim, Suin Kim, Yoon Kim, Young-Bum Kim, Irwin King, Brian Kingsbury, Svetlana Kiritchenko, Dietrich Klakow, Alexandre Klementiev, Sigrid Klerke, Roman Klinger, Julien Kloetzer, Simon Kocbek, Arne Köhn*, Daniël de Kok, Prasanth Kolachina, Varada Kolhatkar, Mamoru Komachi, Kazunori Komatani, Rik Koncel-Kedziorski, Fang Kong, Lingpeng Kong, Ioannis Konstas*, Selcuk Kopru, Valia Kordoni, Yannis Korkontzelos, Bhushan Kotnis, Alexander Kotov, Mikhail Kozhevnikov, Martin Krallinger, Julia Kreutzer*, Jayant Krishnamurthy*, Kriste Krstovski, Canasai Kruengkrai, Germán Kruszewski, Mark Kröll, Lun-Wei Ku*, Marco Kuhlmann, Jonas Kuhn, Roland Kuhn, Shankar Kumar, Jonathan K. Kummerfeld, Sadao Kurohashi, Polina Kuznetsova, Tom Kwiatkowski,

Igor Labutov, Wai Lam, Patrik Lambert, Man Lan, Ian Lane, Ni Lao, Mirella Lapata, Shalom Lappin, Romain Laroche, Kornel Laskowski, Jey Han Lau, Alon Lavie, Angeliki Lazaridou, Phong Le*, Joseph Le Roux, Robert Leaman, Kenton Lee, Lung-Hao Lee, Moontae Lee, Sungjin Lee, Yoong Keok Lee, Young-Suk Lee, Els Lefever, Tao Lei, Jochen L. Leidner, Alessandro Lenci, Yves Lepage*, Johannes Leveling, Tomer Levinboim, Gina-Anne Levow*, Omer Levy*, Roger Levy, Dave Lewis, Mike Lewis, Binyang Li, Chen Li, Cheng-Te Li, Chenliang Li, Fangtao Li, Haizhou Li, Hang Li, Jiwei Li, Junhui Li, Junyi Jessy Li, Lishuang Li, Peifeng Li, Peng Li, Qi Li, Qing Li, Shaohua Li, Sheng Li, Shoushan Li, Xiaoli Li, Yanran Li, Yunyao Li, Zhenghua Li, Maria Liakata*, Kexin Liao*, Xiangwen Liao, Chin-Yew Lin, Chu-Cheng Lin, Chuan-Jie Lin, Shou-de Lin, Victoria Lin, Ziheng Lin, Wang Ling, Xiao Ling, Tal Linzen, Christina Lioma, Pierre Lison, Marina Litvak, Bing Liu, Fei Liu, Hongfang Liu, Jiangming Liu, Lemao Liu, Qian Liu, Qun Liu, Tie-Yan Liu, Ting Liu, Xiaobing Liu, Yang Liu, Nikola Ljubešić, Chi-kiu Lo, Henning Lobin, Varvara Logacheva, Lucelene Lopes, Adam Lopez, Oier Lopez de Lacalle, Aurelio Lopez-Lopez, Annie Louis, Bin Lu, Yi Luan, Andy Luecking, Michal Lukasik, Xiaoqiang Luo, Anh Tuan Luu

Ji Ma, Qingsong Ma, Shuming Ma, Xuezhe Ma, Wolfgang Macherey, Nitin Madnani, Saad Mahamood, Cerstin Mahlow, Wolfgang Maier, Prodromos Malakasiotis, Andreas Maletti, Shervin Malmasi, Titus von der Malsburg, Suresh Manandhar, Gideon Mann, Diego Marcheggiani, Daniel Marcu, David Mareček, Matthew Marge, Benjamin Marie, Katja Markert, Marie-Catherine de Marneffe, Erwin Marsi, Patricio Martinez-Barco, André F. T. Martins*, Sebastian Martschat, Héctor Martínez Alonso, Eugenio Martínez-Cámara*, Fernando Martínez-Santiago, Yann Mathet, Shigeki Matsubara, Yuichiroh Matsubayashi, Yuji Matsumoto, Takuya Matsuzaki, Austin Matthews, Jonathan May, David McClosky, Tara McIntosh, Kathy McKeown, Michael McTear, Yashar Mehdad, Sameep Mehta, Hongyuan Mei*, Yelena Mejova, Oren Melamud, Fandong Meng, Adam Meyers, Yishu Miao, Rada Mihalcea, Todor Mihaylov, Timothy Miller, Tristan Miller*, David Mimno, Bonan Min, Zhao-Yan Ming, Shachar Mirkin, Seyed Abolghasem Mirroshandel, Abhijit Mishra, Prasenjit Mitra, Makoto Miwa, Daichi Mochihashi, Ashutosh Modi, Marie-Francine Moens, Samaneh Moghaddam, Abdelrahman Mohamed, Behrang Mohit, Mitra Mohtarami, Karo Moilanen, Luis Gerardo Mojica de la Vega, Manuel Montes, Andres Montoyo, Taesun Moon, Michael Moortgat, Roser Morante, Hajime Morita, Lili Mou, Dana Movshovitz-Attias, Arjun Mukherjee, Philippe Muller, Yugo Murawaki, Brian Murphy, Gabriel Murray*, Reinhard Muskens, Sung-Hyon Myaeng

Masaaki Nagata, Ajay Nagesh, Vinita Nahar, Iftekhar Naim, Tetsuji Nakagawa, Mikio Nakano, Yukiko Nakano, Ndapandula Nakashole, Ramesh Nallapati, Courtney Napoles, Jason Naradowsky, Karthik Narasimhan*, Shashi Narayan, Alexis Nasr, Vivi Nastase, Borja Navarro, Roberto Navigli, Adeline Nazarenko*, Mark-Jan Nederhof, Arvind Neelakantan, Sapna Negi, Matteo Negri, Aida Nematzadeh, Guenter Neumann, Mariana Neves, Denis Newman-Griffis, Dominick Ng, Hwee Tou Ng, Jun-Ping Ng, Vincent Ng, Dong Nguyen*, Thien Huu Nguyen, Truc-Vien T. Nguyen, Viet-An Nguyen, Garrett Nicolai, Massimo Nicosia, Vlad Niculae*, Jian-Yun Nie, Jan Niehues, Luis Nieto Piña*, Ivelina Nikolova, Malvina Nissim*, Joakim Nivre*, Hiroshi Noji, Gertjan van Noord, Joel Nothman

Brendan O'Connor, Timothy O'Donnell, Yusuke Oda, Stephan Oepen, Kemal Oflazer*, Alice Oh*, Jong-Hoon Oh, Tomoko Ohta, Kiyonori Ohtake, Hidekazu Oiwa, Naoaki Okazaki, Manabu Okumura, Hiroshi G. Okuno, Constantin Orasan, Vicente Ordonez, Petya Osenova, Mari Ostendorf*, Myle Ott, Katja Ovchinnikova, Cecilia Ovesdotter Alm

Muntsa Padró, Valeria de Paiva, Alexis Palmer, Martha Palmer, Alessio Palmero Aprosio, Sinno Jialin Pan*, Xiaoman Pan, Denis Paperno, Ankur Parikh, Cecile Paris, Seong-Bae Park, Tommaso Pasini, Marco Passarotti*, Peyman Passban, Panupong Pasupat, Siddharth Patwardhan, Michael J. Paul*, Adam Pauls, Ellie Pavlick*, Adam Pease, Pavel Pecina, Ted Pedersen, Nanyun Peng, Xiaochang Peng, Gerald Penn, Marco Pennacchiotti, Bryan Perozzi, Casper Petersen, Slav Petrov, Eva Pettersson, Anselmo Peñas, Hieu Pham, Nghia The Pham, Lawrence Phillips, Davide Picca, Karl Pichotta, Olivier Pietquin, Mohammad Taher Pilehvar, Yuval Pinter, Paul Piwek, Thierry Poibeau, Tamara Polajnar, Heather Pon-Barry, Simone Paolo Ponzetto, Andrei Popescu-Belis, Maja Popović, Fred Popowich, François Portet*, Matt Post*, Christopher Potts, Vinodkumar Prabhakaran, Daniel Preoţiuc-Pietro, Emily Prud'hommeaux*, Laurent Prévot, Jay Pujara, Matthew Purver, James Pustejovsky

Juan Antonio Pérez-Ortiz, Ashequl Qadir, Peng Qi, Longhua Qian, Xian Qian, Lu Qin, Long Qiu, Xipeng Qiu, Lizhen Qu, Ariadna Quattoni, Chris Quirk* Alexandre Rademaker, Will Radford, Alessandro Raganato, Afshin Rahimi*, Altaf Rahman, Maya Ramanath, Rohan Ramanath, A Ramanathan, Arti Ramesh, Gabriela Ramirez-de-la-Rosa, Carlos Ramisch, Anita Ramm, Vivek Kumar Rangarajan Sridhar, Ari Rappoport, Mohammad Sadegh Rasooli, Pushpendre Rastogi, An-

toine Raux, Sravana Reddy, Ines Rehbein*, Georg Rehm, Roi Reichart, Ehud Reiter, Zhaochun Ren, Corentin Ribeyre, Matthew Richardson, Martin Riedl, Jason Riesa, German Rigau, Ellen Riloff*, Laura Rimell*, Alan Ritter, Brian Roark*, Molly Roberts, Tim Rocktäschel, Oleg Rokhlenko, Salvatore Romeo, Andrew Rosenberg, Sara Rosenthal*, Paolo Rosso, Benjamin Roth, Michael Roth, Sascha Rothe, Masoud Rouhizadeh, Mickael Rouvier, Alla Rozovskaya, Josef Ruppenhofer, Delia Rusu, Attapol Rutherford

Mrinmaya Sachan, Kugatsu Sadamitsu, Fatiha Sadat, Mehrnoosh Sadrzadeh, Markus Saers, Kenji Sagae, Horacio Saggion, Saurav Sahay, Magnus Sahlgren, Patrick Saint-dizier, Hassan Sajjad, Sakriani Sakti, Mohammad Salameh, Bahar Salehi, Avneesh Saluja, Rajhans Samdani, Mark Sammons, Germán Sanchis-Trilles, Ryohei Sasano, Agata Savary*, Asad Sayeed, Carolina Scarton, Tatjana Scheffler, Christian Scheible, David Schlangen, Natalie Schluter, Allen Schmaltz*, Helmut Schmid, Alexandra Schofield, William Schuler, Sebastian Schuster, Lane Schwartz, Roy Schwartz*, Christof Schöch, Diarmuid Ó Séaghdha, Djamé Seddah, Abigail See, Nina Seemann, Satoshi Sekine, Mark Seligman, Minjoon Seo, Burr Settles, Lei Sha, Kashif Shah, Rebecca Sharp, Shiqi Shen, Shuming Shi, Hiroyuki Shindo, Koichi Shinoda, Chaitanya Shivade, Eyal Shnarch*, Milad Shokouhi, Ekaterina Shutova, Advaith Siddharthan, Avirup Sil, Carina Silberer, Yanchuan Sim, Patrick Simianer, Kiril Simov, Kairit Sirts, Amy Siu, Gabriel Skantze, Kevin Small, Noah A. Smith, Pavel Smrz, Richard Socher, Artem Sokolov, Thamar Solorio, Swapna Somasundaran, Hyun-Je Song, Min Song, Sa-kwang Song, Yang Song, Yangqiu Song, Radu Soricut, Aitor Soroa, Matthias Sperber, Caroline Sporleder, Vivek Srikumar, Somayajulu Sripada, Shashank Srivastava, Edward Stabler, Jan Šnajder*, Sanja Štajner, Gabriel Stanovsky*, Manfred Stede, Mark Steedman, Josef Steinberger, Amanda Stent, Mark Stevenson, Brandon Stewart, Matthew Stone, Svetlana Stoyanchev, Veselin Stoyanov, Carlo Strapparava, Karl Stratos, Kristina Striegnitz*, Emma Strubell, Tomek Strzalkowski, Sara Stymne, Maik Stührenberg, Jinsong Su, Keh-Yih Su, Yu Su, L V Subramaniam, Katsuhito Sudoh, Ang Sun, Huan Sun, Le Sun, Weiwei Sun, Xu Sun, Simon Suster, Hisami Suzuki, Jun Suzuki, Yoshimi Suzuki, Swabha Swayamdipta, Stan Szpakowicz, Idan Szpektor, Felipe Sánchez-Martínez, Pascale Sébillot, Anders Søgaard

Prasad Tadepalli, Kaveh Taghipour, Hiroya Takamura, David Talbot, Partha Talukdar, Aleš Tamchyna, Akihiro Tamura, Chenhao Tan*, Liling Tan, Niket Tandon, Duyu Tang, Jiliang Tang, Christoph Teichmann, Serra Sinem Tekiroglu, Irina Temnikova, Joel Tetreault, Kapil Thadani*, Sam Thomson, Jörg Tiedemann, Ivan Titov, Katrin Tomanek, Gaurav Singh Tomar, Marc Tomlinson, Sara Tonelli, Antonio Toral, Kentaro Torisawa, Ke M. Tran, Isabel Trancoso, Ming-Feng Tsai, Richard Tzong-Han Tsai, Reut Tsarfaty*, Oren Tsur, Yoshimasa Tsuruoka, Yulia Tsvetkov, Cunchao Tu, Zhaopeng Tu, Gokhan Tur, Marco Turchi, Ferhan Ture, Oscar Täckström

Raghavendra Udupa, Stefan Ultes, Lyle Ungar, Shyam Upadhyay, L. Alfonso Urena Lopez, Olga Uryupina

Alessandro Valitutti*, Benjamin Van Durme*, Tim Van de Cruys, Lucy Vanderwende*, Vasudeva Varma, Paola Velardi, Sumithra Velupillai, Sriram Venkatapathy, Yannick Versley, Tim Vieira, David Vilar, Martín Villalba*, Veronika Vincze, Sami Virpioja*, Andreas Vlachos, Rob Voigt, Ngoc Thang Vu, Ivan Vulić, Yogarshi Vyas, V.G.Vinod Vydiswaran, Ekaterina Vylomova

Houfeng Wang, Henning Wachsmuth, Joachim Wagner, Matthew Walter*, Stephen Wan, Xiaojun Wan, Baoxun Wang, Chang Wang, Chong Wang, Dingquan Wang, Hongning Wang, Lu Wang, Mingxuan Wang, Pidong Wang, Rui Wang, Shaojun Wang, Tong Wang, Yu-Chun Wang, Wei Wang, Wenya Wang, William Yang Wang, Xiaolin Wang, Xiaolong Wang, Yiou Wang, Zhiguo Wang, Zhongqing Wang*, Leo Wanner, Nigel Ward, Shinji Watanabe, Taro Watanabe, Aleksander Wawer, Bonnie Webber, Ingmar Weber, Julie Weeds, Furu Wei, Zhongyu Wei, Gerhard Weikum, David Weir, Michael White, Antoine Widlöcher, Michael Wiegand*, Jason D Williams*, Shuly Wintner, Sam Wiseman, Michael Witbrock, Silke Witt-Ehsani, Travis Wolfe, Kam-Fai Wong, Jian Wu, Yuanbin Wu, Joern Wuebker

Aris Xanthos, Rui Xia, Yunqing Xia, Bing Xiang, Min Xiao, Tong Xiao, Xinyan Xiao, Boyi Xie, Pengtao Xie, Shasha Xie, Chenyan Xiong, Feiyu Xu, Hua Xu, Ruifeng Xu, Wei Xu, Wenduan Xu, Huichao Xue, Nianwen Xue

Yadollah Yaghoobzadeh, Ichiro Yamada, Bishan Yang, Cheng Yang, Diyi Yang, Grace Hui Yang, Min Yang, Yaqin Yang, Yi Yang, Roman Yangarber, Mark Yatskar, Meliha Yetisgen, Wen-tau Yih, Pengcheng Yin, Wenpeng Yin, Anssi Yli-Jyrä, Dani Yogatama, Naoki Yoshinaga, Bei Yu, Dian Yu, Dianhai Yu, Kai Yu, Liang-Chih Yu, Mo Yu, Zhou Yu, François Yvon

Marcos Zampieri, Menno van Zaanen, Fabio Massimo Zanzotto, Amir Zeldes*, Daojian Zeng, Xiaodong Zeng, Kalliopi Zervanou*, Luke Zettlemoyer, Deniz Zeyrek, Feifei Zhai, Congle Zhang, Dongdong Zhang, Guchun Zhang, Jiajun Zhang, Jian Zhang, Jianwen Zhang, Lei Zhang, Meishan Zhang, Min Zhang, Qi Zhang, Renxian Zhang, Sicong Zhang, Wei Zhang*, Zhisong Zhang, Bing Zhao, Dongyan Zhao, Jun Zhao, Shiqi Zhao, Tiejun Zhao, Wayne Xin Zhao, Alisa Zhila, Guodong Zhou, Xinjie Zhou, Muhua Zhu, Xiaodan Zhu, Xiaoning Zhu, Ayah Zirikly, Chengqing Zong, Bowei Zou

Secondary Reviewers

Naveed Afzal, Yamen Ajjour

Jeremy Barnes, Joost Bastings, Joachim Bingel, Luana Bulat

Iacer Calixto, Lea Canales, Kai Chen, Tongfei Chen, Hao Cheng, Jianpeng Cheng, Yiming Cui

Marco Damonte, Saman Daneshvar, Tobias Domhan, Daxiang Dong, Li Dong

Mohamed Eldesouki

Stefano Faralli, Bin Fu

Srinivasa P. Gadde, Qiaozi Gao, Luca Gilardi, Sujatha Das Gollapalli, J. Manuel Gomez, Stig-Arne Grönroos, Lin Gui

Casper Hansen, Lihong He, Martin Horn

Oana Inel

Gongye Jin

Roman Kern, Vaibhav Kesarwani, Joo-Kyung Kim, Seongchan Kim, Christine Köhn, Santosh Kosgi

Ronja Laarmann-Quante, Egoitz Laparra, Anais Lefeuvre-Halftermeyer, Guanlin Li, Jing Li, Minglei Li, Xiang Li, Xiaolong Li, Chen Liang, Ming Liao, Sijia Liu, Pranay Lohia

Chunpeng Ma, Shuming Ma, Tengfei Ma, Ana Marasovic

Toan Nguyen, Eric Nichols, Sergiu Nisioi

Gözde Özbal

Alexis Palmer, Suraj Pandey, Nikolaos Pappas, José M. Perea-Ortega, Marten Postma

Longhua Qian

Masoud Rouhizadeh Abeed Sarker, Andrew Schneider, Roxane Segers, Pararth Shah, Samiulla Shaikh, Xing Shi, Tomohide Shibata, Samiulla Shiekh, Miikka Silfverberg

Bo Wang*, Boli Wang, Jianxiang Wang, Jingjing Wang, Rui Wang, Shuai Wang, Shuting Wang, Tsung-Hsien Wen, John Wieting

Nan Yang, Yi Yang, Mark Yatskar, Yichun Yin

Sheng Zhang, Kai Zhao, Imed Zitouni

Outstanding Papers

With twin upward trends in the interest in computational linguistics and natural language processing and the size of our annual meeting, ACL has begun the practice of recognizing outstanding papers that represent a select cross-section of the entire field, as nominated by reviewers and vetted by the area chairs and program co-chairs. These papers have been centrally located in the program, on the last day of our meeting, in a more focused two parallel tracks format.

This year, we have nominated 15 long papers and 7 short papers, representing 1.8% of all submissions and approximately 5% of the accepted ACL program. Congratulations, authors!

(in alphabetical order by first author surname)

Long Papers

- Jan Buys and Phil Blunsom. *Robust Incremental Neural Semantic Graph Parsing.*
- Xinchi Chen, Zhan Shi, Xipeng Qiu and Xuanjing Huang. *Adversarial Multi-Criteria Learning for Chinese Word Segmentation.*
- Ryan Cotterell and Jason Eisner. *Probabilistic Typology: Deep Generative Models of Vowel Inventories.*
- Yanzhuo Ding, Yang Liu, Huanbo Luan and Maosong Sun. *Visualizing and Understanding Neural Machine Translation.*
- Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham and Nigel Collier. *Vancouver Welcomes You! Minimalist Location Metonymy Resolution.*
- Daniel Hershcovich, Omri Abend and Ari Rappoport. *A Transition-Based Directed Acyclic Graph Parser for UCCA.*
- Shuhei Kurita, Daisuke Kawahara and Sadao Kurohashi. *Neural Joint Model for Transition-based Chinese Syntactic Analysis.*
- Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio and Joelle Pineau. *Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses.*
- Yasuhide Miura, Motoki Taniguchi, Tomoki Taniguchi and Tomoko Ohkuma. *Unifying Text, Metadata, and User Network Representations with a Neural Network for Geolocation Prediction.*
- Ramakanth Pasunuru and Mohit Bansal. *Multi-Task Video Captioning with Visual and Entailment Generation.*
- Maxim Rabinovich, Mitchell Stern and Dan Klein. *Abstract Syntax Networks for Code Generation and Semantic Parsing.*
- Ines Rehbein and Josef Ruppenhofer. *Detecting annotation noise in automatically labelled data.*
- Jiwei Tan, Xiaojun Wan and Jianguo Xiao. *Abstractive Document Summarization with a Graph-Based Attentional Neural Model.*
- Mingbin Xu, Hui Jiang and Sedtawut Watcharawittayakul. *A Local Detection Approach for Named Entity Recognition and Mention Detection.*
- Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou and Bo Xu. *Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme.*

Short Papers

- Xinyu Hua and Lu Wang. *Understanding and Detecting Diverse Supporting Arguments on Controversial Issues.*
- Jindřich Libovický and Jindřich Helcl. *Attention Strategies for Multi-Source Sequence-to-Sequence Learning.*
- Bogdan Ludusan, Reiko Mazuka, Mathieu Bernard, Alejandrina Cristia and Emmanuel Dupoux. *The Role of Prosody and Speech Register in Word Segmentation: A Computational Modelling Perspective.*
- Afshin Rahimi, Trevor Cohn and Timothy Baldwin. *A Neural Model for User Geolocation and Lexical Dialectology.*
- Keisuke Sakaguchi, Matt Post and Benjamin Van Durme. *Error-repair Dependency Parsing for Ungrammatical Texts.*
- Alane Suhr, Mike Lewis, James Yeh and Yoav Artzi. *A Corpus of Compositional Language for Visual Reasoning.*
- Yizhong Wang, Sujian Li and Houfeng Wang. *A Two-stage Parsing Method for Text-level Discourse Analysis.*

Invited Talk: Squashing Computational Linguistics

Noah A. Smith

Paul G. Allen School of Computer Science and Engineering, University of Washington

Abstract

The computational linguistics and natural language processing community is experiencing an episode of deep fascination with representation learning. Like many other presenters at this conference, I will describe new ways to use representation learning in models of natural language. Noting that a data-driven model always assumes a theory (not necessarily a good one), I will argue for the benefits of language-appropriate inductive bias for representation-learning-infused models of language. Such bias often comes in the form of assumptions baked into a model, constraints on an inference algorithm, or linguistic analysis applied to data. Indeed, many decades of research in linguistics (including computational linguistics) put our community in a strong position to identify promising inductive biases. The new models, in turn, may allow us to explore previously unavailable forms of bias, and to produce findings of interest to linguistics. I will focus on new models of documents and of sentential semantic structures, and I will emphasize abstract, reusable components and their assumptions rather than applications.

Biography

Noah Smith is an Associate Professor in the Paul G. Allen School of Computer Science and Engineering at the University of Washington. Previously, he was an Associate Professor in the School of Computer Science at Carnegie Mellon University. He received his Ph.D. in Computer Science from Johns Hopkins University and his B.S. in Computer Science and B.A. in Linguistics from the University of Maryland. His research spans many topics in natural language processing, machine learning, and computational social science. He has served on the editorial boards of CL, JAIR, and TACL, as the secretary-treasurer of SIGDAT (2012–2015), and as program co-chair of ACL 2016. Alumni of his research group, Noah's ARK, are international leaders in NLP in academia and industry. Smith's work has been recognized with a UW Innovation award, a Finmeccanica career development chair at CMU, an NSF CAREER award, a Hertz Foundation graduate fellowship, numerous best paper nominations and awards, and coverage by NPR, BBC, CBC, the New York Times, the Washington Post, and Time.

Invited Talk: Translating from Multiple Modalities to Text and Back

Mirella Lapata

Professor, School of Informatics, University of Edinburgh

Abstract

Recent years have witnessed the development of a wide range of computational tools that process and generate natural language text. Many of these have become familiar to mainstream computer users in the from of web search, question answering, sentiment analysis, and notably machine translation. The accessibility of the web could be further enhanced with applications that not only translate between different languages (e.g., from English to French) but also within the same language, between different modalities, or different data formats. The web is rife with non-linguistic data (e.g., video, images, source code) that cannot be indexed or searched since most retrieval tools operate over textual data.

In this talk I will argue that in order to render electronic data more accessible to individuals and computers alike, new types of translation models need to be developed. I will focus on three examples, text simplification, source code generation, and movie summarization. I will illustrate how recent advances in deep learning can be extended in order to induce general representations for different modalities and learn how to translate between these and natural language.

Biography

Mirella Lapata is professor of natural language processing in the School of Informatics at the University of Edinburgh. Her research focuses on getting computers to understand, reason with, and generate. She is as an associate editor of the Journal of Artificial Intelligence Research and has served on the editorial boards of Transactions of the ACL and Computational Linguistics. She was the first recipient of the Karen Sparck Jones award of the British Computer Society, recognizing key contributions to NLP and information retrieval. She received two EMNLP best paper awards and currently holds a prestigious Consolidator Grant from the European Research Council.

Table of Contents

Conference Program

Monday, July 31st

10:30–11:45 Session 1A: Information Extraction 1 (NN)

10:30–10:48 *Adversarial Multi-task Learning for Text Classification*
Pengfei Liu, Xipeng Qiu and Xuanjing Huang

10:49–11:07 *Neural End-to-End Learning for Computational Argumentation Mining*
Steffen Eger, Johannes Daxenberger and Iryna Gurevych

11:08–11:26 *Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision*
Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus and Ni Lao

11:27–11:45 *Neural Relation Extraction with Multi-lingual Attention*
Yankai Lin, Zhiyuan Liu and Maosong Sun

Monday, July 31st

10:30–11:45 Session 1B: Semantics 1

10:30–10:48 *Learning Structured Natural Language Representations for Semantic Parsing*
Jianpeng Cheng, Siva Reddy, Vijay Saraswat and Mirella Lapata

10:49–11:07 *Morph-fitting: Fine-Tuning Word Vector Spaces with Simple Language-Specific Rules*
Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young and Anna Korhonen

11:08–11:26 *Skip-Gram - Zipf + Uniform = Vector Additivity*
Alex Gittens, Dimitris Achlioptas and Michael W. Mahoney

11:27–11:45 *The State of the Art in Semantic Representation*
Omri Abend and Ari Rappoport

Monday, July 31st

10:30–11:26 Session 1C: Discourse 1

10:30–10:48 *Joint Learning for Event Coreference Resolution*
Jing Lu and Vincent Ng

10:49–11:07 *Generating and Exploiting Large-scale Pseudo Training Data for Zero Pronoun Resolution*
Ting Liu, Yiming Cui, Qingyu Yin, Wei-Nan Zhang, Shijin Wang and Guoping Hu

11:08–11:26 *Discourse Mode Identification in Essays*
Wei Song, Dong Wang, Ruiji Fu, Lizhen Liu, Ting Liu and Guoping Hu

Monday, July 31st

10:30–11:07 Session 1D: Machine Translation 1

10:30–10:48 *A Convolutional Encoder Model for Neural Machine Translation*
Jonas Gehring, Michael Auli, David Grangier and Yann Dauphin

10:49–11:07 *Deep Neural Machine Translation with Linear Associative Unit*
Mingxuan Wang, Zhengdong Lu, Jie Zhou and Qun Liu

Monday, July 31st

10:30–11:45 Session 1E: Generation 1

10:30–10:48 *Neural AMR: Sequence-to-Sequence Models for Parsing and Generation*
Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi and Luke Zettlemoyer

10:49–11:07 *Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems*
Wang Ling, Dani Yogatama, Chris Dyer and Phil Blunsom

11:08–11:26 *Automatically Generating Rhythmic Verse with Neural Networks*
Jack Hopkins and Douwe Kiela

11:27–11:45 *Creating Training Corpora for NLG Micro-Planners*
Claire Gardent, Anastasia Shimorina, Shashi Narayan and Laura Perez-Beltrachini

Monday, July 31st

13:40–14:55 Session 2A: Question Answering 1

13:40–13:58 *Gated Self-Matching Networks for Reading Comprehension and Question Answering*
Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang and Ming Zhou

13:59–14:17 *Generating Natural Answers by Incorporating Copying and Retrieving Mechanisms in Sequence-to-Sequence Learning*
Shizhu He, Cao Liu, Kang Liu and Jun Zhao

14:18–14:36 *Coarse-to-Fine Question Answering for Long Documents*
Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste and Jonathan Berant

14:37–14:55 *An End-to-End Model for Question Answering over Knowledge Base with Cross-Attention Combining Global Knowledge*
Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu and Jun Zhao

Monday, July 31st

13:40–14:55 Session 2B: Vision 1

13:40–13:58 *Translating Neuralese*
Jacob Andreas, Anca Dragan and Dan Klein

13:59–14:17 *Obtaining referential word meanings from visual and distributional information: Experiments on object naming*
Sina Zarrieß and David Schlangen

14:18–14:36 *FOIL it! Find One mismatch between Image and Language caption*
Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich, Aurélie Herbelot, Moin Nabi, Enver Sangineto and Raffaella Bernardi

14:37–14:55 *Verb Physics: Relative Physical Knowledge of Actions and Objects*
Maxwell Forbes and Yejin Choi

Monday, July 31st

13:40–14:36 Session 2C: Syntax 1

13:40–13:58 *A* CCG Parsing with a Supertag and Dependency Factored Model*
Masashi Yoshikawa, Hiroshi Noji and Yuji Matsumoto

13:59–14:17 *A Full Non-Monotonic Transition System for Unrestricted Non-Projective Parsing*
Daniel Fernández-González and Carlos Gómez-Rodríguez

14:18–14:36 *Aggregating and Predicting Sequence Labels from Crowd Annotations*
An Thanh Nguyen, Byron Wallace, Junyi Jessy Li, Ani Nenkova and Matthew Lease

Monday, July 31st

13:40–15:14 Session 2D: Machine Learning 1 (NN)

13:40–13:58 *Multi-space Variational Encoder-Decoders for Semi-supervised Labeled Sequence Transduction*
Chunting Zhou and Graham Neubig

13:59–14:17 *Scalable Bayesian Learning of Recurrent Neural Networks for Language Modeling*
Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su and Lawrence Carin

14:18–14:36 *Learning attention for historical text normalization by learning to pronounce*
Marcel Bollmann, Joachim Bingel and Anders Søgaard

14:37–14:55 *Deep Learning in Semantic Kernel Spaces*
Danilo Croce, Simone Filice, Giuseppe Castellucci and Roberto Basili

14:56–15:14 *Topically Driven Neural Language Model*
Jey Han Lau, Timothy Baldwin and Trevor Cohn

Monday, July 31st

13:40–14:55 Session 2E: Sentiment 1 (NN)

13:40–13:58 *Handling Cold-Start Problem in Review Spam Detection by Jointly Embedding Texts and Behaviors*
Xuepeng Wang, Kang Liu and Jun Zhao

13:59–14:17 *Learning Cognitive Features from Gaze Data for Sentiment and Sarcasm Classification using Convolutional Neural Network*
Abhijit Mishra, Kuntal Dey and Pushpak Bhattacharyya

14:18–14:36 *An Unsupervised Neural Attention Model for Aspect Extraction*
Ruidan He, Wee Sun Lee, Hwee Tou Ng and Daniel Dahlmeier

14:37–14:55 *Other Topics You May Also Agree or Disagree: Modeling Inter-Topic Preferences using Tweets and Matrix Factorization*
Akira Sasaki, Kazuaki Hanawa, Naoaki Okazaki and Kentaro Inui

Monday, July 31st

15:45–16:41 Session 3A: Information Extraction 2 / Biomedical 1

15:45–16:03 *Automatically Labeled Data Generation for Large Scale Event Extraction*
Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu and Jun Zhao

16:04–16:22 *Time Expression Analysis and Recognition Using Syntactic Token Types and General Heuristic Rules*
Xiaoshi Zhong, Aixin Sun and Erik Cambria

16:23–16:41 *Learning with Noise: Enhance Distantly Supervised Relation Extraction with Dynamic Transition Matrix*
Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan and Dongyan Zhao

Monday, July 31st

15:45–17:00 Session 3B: Semantics 2 (NN)

15:45–16:03 *A Syntactic Neural Model for General-Purpose Code Generation*
Pengcheng Yin and Graham Neubig

16:04–16:22 *Learning bilingual word embeddings with (almost) no bilingual data*
Mikel Artetxe, Gorka Labaka and Eneko Agirre

16:23–16:41 *Abstract Meaning Representation Parsing using LSTM Recurrent Neural Networks*
William Foland and James H. Martin

16:42–17:00 *Deep Semantic Role Labeling: What Works and What's Next*
Luheng He, Kenton Lee, Mike Lewis and Luke Zettlemoyer

Monday, July 31st

15:45–17:00 Session 3C: Speech 1 / Dialogue 1

15:45–16:03 *Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access*
Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed and Li Deng

16:04–16:22 *Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-Based Chatbots*
Yu Wu, Wei Wu, Chen Xing, Ming Zhou and Zhoujun Li

16:23–16:41 *Learning Word-Like Units from Joint Audio-Visual Analysis*
David Harwath and James Glass

16:42–17:00 *Joint CTC/attention decoding for end-to-end speech recognition*
Takaaki Hori, Shinji Watanabe and John Hershey

Monday, July 31st

15:45–16:22 Session 3D: Multilingual 1

15:45–16:03 *Found in Translation: Reconstructing Phylogenetic Language Trees from Translations*
Ella Rabinovich, Noam Ordan and Shuly Wintner

16:04–16:22 *Predicting Native Language from Gaze*
Yevgeni Berzak, Chie Nakamura, Suzanne Flynn and Boris Katz

Monday, July 31st

15:45–16:03 Session 3E: Phonology 1

15:45–16:03 *MORSE: Semantic-ally Drive-n MORpheme SEgment-er*
Tarek Sakakini, Suma Bhat and Pramod Viswanath

Tuesday, August 1st

10:30–11:45 Session 4A: Information Extraction 3 (NN)

10:30–10:48 *Deep Pyramid Convolutional Neural Networks for Text Categorization*
Rie Johnson and Tong Zhang

10:49–11:07 *Improved Neural Relation Detection for Knowledge Base Question Answering*
Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang and Bowen Zhou

11:08–11:26 *Deep Keyphrase Generation*
Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky and Yu Chi

11:27–11:45 *Attention-over-Attention Neural Networks for Reading Comprehension*
Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu and Guoping Hu

Tuesday, August 1st

10:30–11:26 Session 4B: Cognitive Modelling 1 / Vision 2

10:30–10:48 *Alignment at Work: Using Language to Distinguish the Internalization and Self-Regulation Components of Cultural Fit in Organizations*
Gabriel Doyle, Amir Goldberg, Sameer Srivastava and Michael Frank

10:49–11:07 *Representations of language in a model of visually grounded speech signal*
Grzegorz Chrupała, Lieke Gelderloos and Afra Alishahi

11:08–11:26 *Spectral Analysis of Information Density in Dialogue Predicts Collaborative Task Performance*
Yang Xu and David Reitter

Tuesday, August 1st

10:30–12:04 Session 4C: Dialogue 2

10:30–10:48 *Affect-LM: A Neural Language Model for Customizable Affective Text Generation*
Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency and Stefan Scherer

10:49–11:07 *Domain Attention with an Ensemble of Experts*
Young-Bum Kim, Karl Stratos and Dongchan Kim

11:08–11:26 *Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders*
Tiancheng Zhao, Ran Zhao and Maxine Eskenazi

11:27–11:45 *Hybrid Code Networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning*
Jason D Williams, Kavosh Asadi and Geoffrey Zweig

11:46–12:04 *Generating Contrastive Referring Expressions*
Martin Villalba, Christoph Teichmann and Alexander Koller

Tuesday, August 1st

10:30–11:07 Session 4D: Machine Translation 2

10:30–10:48 *Modeling Source Syntax for Neural Machine Translation*
Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang and Guodong Zhou

10:49–11:07 *Sequence-to-Dependency Neural Machine Translation*
Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li and Ming Zhou

Tuesday, August 1st

10:30–11:45 Session 4E: Social Media 1

10:30–10:48 *Detect Rumors in Microblog Posts Using Propagation Structure via Kernel Learning*
Jing Ma, Wei Gao and Kam-Fai Wong

10:49–11:07 *EmoNet: Fine-Grained Emotion Detection with Gated Recurrent Neural Networks*
Muhammad Abdul-Mageed and Lyle Ungar

11:08–11:26 *Beyond Binary Labels: Political Ideology Prediction of Twitter Users*
Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins and Lyle Ungar

11:27–11:45 *Leveraging Behavioral and Social Information for Weakly Supervised Collective Classification of Political Discourse on Twitter*
Kristen Johnson, Di Jin and Dan Goldwasser

Tuesday, August 1st

13:49–14:39 Session 5A: Multidisciplinary 1

13:49–14:07 *A Nested Attention Neural Hybrid Model for Grammatical Error Correction*
Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong and Jianfeng Gao

14:08–14:26 *TextFlow: A Text Similarity Measure based on Continuous Sequences*
Yassine Mrabet, Halil Kilicoglu and Dina Demner-Fushman

14:27–14:39 *Friendships, Rivalries, and Trysts: Characterizing Relations between Ideas in Texts*
Chenhao Tan, Dallas Card and Noah A. Smith

Tuesday, August 1st

13:30–14:26 Session 5B: Language and Resources 1

13:30–13:48 *Polish evaluation dataset for compositional distributional semantics models*
Alina Wróblewska and Katarzyna Krasnowska-Kieraś

13:49–14:07 *Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction*
Christopher Bryant, Mariano Felice and Ted Briscoe

14:08–14:26 *Evaluation Metrics for Machine Reading Comprehension: Prerequisite Skills and Readability*
Saku Sugawara, Yusuke Kido, Hikaru Yokono and Akiko Aizawa

Tuesday, August 1st

13:30–14:26 Session 5C: Syntax 2 (NN)

13:30–13:48 *A Minimal Span-Based Neural Constituency Parser*
Mitchell Stern, Jacob Andreas and Dan Klein

13:49–14:07 *Semantic Dependency Parsing via Book Embedding*
Weiwei Sun, Junjie Cao and Xiaojun Wan

14:08–14:26 *Neural Word Segmentation with Rich Pretraining*
Jie Yang, Yue Zhang and Fei Dong

Tuesday, August 1st

13:30–14:07 Session 5D: Machine Translation 3 (NN)

13:30–13:48 *Neural Machine Translation via Binary Code Prediction*
Yusuke Oda, Philip Arthur, Graham Neubig, Koichiro Yoshino and Satoshi Nakamura

13:49–14:07 *What do Neural Machine Translation Models Learn about Morphology?*
Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad and James Glass

Tuesday, August 1st

13:30–14:07 Session 5E: Sentiment 2

13:30–13:48 *Context-Dependent Sentiment Analysis in User-Generated Videos*
Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh and Louis-Philippe Morency

13:49–14:07 *A Multidimensional Lexicon for Interpersonal Stancetaking*
Umashanthi Pavalanathan, Jim Fitzpatrick, Scott Kiesling and Jacob Eisenstein

Tuesday, August 1st

15:25–16:21 Session 6A: Information Extraction 4

15:25–15:43 *Tandem Anchoring: a Multiword Anchor Approach for Interactive Topic Modeling*
Jeffrey Lund, Connor Cook, Kevin Seppi and Jordan Boyd-Graber

15:44–16:02 *Apples to Apples: Learning Semantics of Common Entities Through a Novel Comprehension Task*
Omid Bakhshandeh and James Allen

16:03–16:21 *Going out on a limb: Joint Extraction of Entity Mentions and Relations without Dependency Trees*
Arzoo Katiyar and Claire Cardie

Tuesday, August 1st

15:25–16:40 Session 6B: Semantics 2 (NN)

15:25–15:43 *Naturalizing a Programming Language via Interactive Learning*
Sida I. Wang, Samuel Ginn, Percy Liang and Christopher D. Manning

15:44–16:02 *Semantic Word Clusters Using Signed Spectral Clustering*
Joao Sedoc, Jean Gallier, Dean Foster and Lyle Ungar

16:03–16:21 *An Interpretable Knowledge Transfer Model for Knowledge Base Completion*
Qizhe Xie, Xuezhe Ma, Zihang Dai and Eduard Hovy

16:22–16:40 *Learning a Neural Semantic Parser from User Feedback*
Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy and Luke Zettlemoyer

Tuesday, August 1st

15:25–17:00 Session 6C: Discourse 2 / Dialogue 3

15:25–15:43 *Joint Modeling of Content and Discourse Relations in Dialogues*
Kechen Qin, Lu Wang and Joseph Kim

15:44–16:02 *Argument Mining with Structured SVMs and RNNs*
Vlad Niculae, Joonsuk Park and Claire Cardie

16:03–16:21 *Neural Discourse Structure for Text Categorization*
Yangfeng Ji and Noah A. Smith

16:22–16:40 *Adversarial Connective-exploiting Networks for Implicit Discourse Relation Classification*
Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu and Eric Xing

16:41–17:00 *Don't understand a measure? Learn it: Structured Prediction for Coreference Resolution optimizing its measures*
Iryna Haponchyk and Alessandro Moschitti

Tuesday, August 1st

15:25–16:21 Session 6D: Machine Learning 2

15:25–15:43 *Bayesian Modeling of Lexical Resources for Low-Resource Settings*
Nicholas Andrews, Mark Dredze, Benjamin Van Durme and Jason Eisner

15:44–16:02 *Semi-Supervised QA with Generative Domain-Adaptive Nets*
Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov and William Cohen

16:03–16:21 *From Language to Programs: Bridging Reinforcement Learning and Maximum Marginal Likelihood*
Kelvin Guu, Panupong Pasupat, Evan Liu and Percy Liang

Tuesday, August 1st

15:25–17:00 Session 6E: Summarization 1

15:25–15:43 *Diversity driven attention model for query-based abstractive summarization*
Preksha Nema, Mitesh M. Khapra, Anirban Laha and Balaraman Ravindran

15:44–16:02 *Get To The Point: Summarization with Pointer-Generator Networks*
Abigail See, Peter J. Liu and Christopher D. Manning

16:03–16:21 *Supervised Learning of Automatic Pyramid for Optimization-Based Multi-Document Summarization*
Maxime Peyrard and Judith Eckle-Kohler

16:22–16:40 *Selective Encoding for Abstractive Sentence Summarization*
Qingyu Zhou, Nan Yang, Furu Wei and Ming Zhou

16:41–17:00 *PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents*
Corina Florescu and Cornelia Caragea

Wednesday, August 2nd

10:40–11:36 Session 7A: Outstanding Papers 1

10:40–10:58 *Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses*
Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio and Joelle Pineau

10:59–11:17 *A Transition-Based Directed Acyclic Graph Parser for UCCA*
Daniel Hershcovich, Omri Abend and Ari Rappoport

11:18–11:36 *Abstract Syntax Networks for Code Generation and Semantic Parsing*
Maxim Rabinovich, Mitchell Stern and Dan Klein

Wednesday, August 2nd

10:40–11:17 Session 7B: Outstanding Papers 2

10:40–10:58 *Visualizing and Understanding Neural Machine Translation*
Yanzhuo Ding, Yang Liu, Huanbo Luan and Maosong Sun

10:59–11:17 *Detecting annotation noise in automatically labelled data*
Ines Rehbein and Josef Ruppenhofer

Wednesday, August 2nd

15:00–16:34 Session 8A: Outstanding Papers 3

15:00–15:18 *Abstractive Document Summarization with a Graph-Based Attentional Neural Model*
Jiwei Tan, Xiaojun Wan and Jianguo Xiao

15:19–15:37 *Probabilistic Typology: Deep Generative Models of Vowel Inventories*
Ryan Cotterell and Jason Eisner

15:38–15:56 *Adversarial Multi-Criteria Learning for Chinese Word Segmentation*
Xinchi Chen, Zhan Shi, Xipeng Qiu and Xuanjing Huang

15:57–16:15 *Neural Joint Model for Transition-based Chinese Syntactic Analysis*
Shuhei Kurita, Daisuke Kawahara and Sadao Kurohashi

16:16–16:34 *Robust Incremental Neural Semantic Graph Parsing*
Jan Buys and Phil Blunsom

Wednesday, August 2nd

15:00–16:34 Session 8B: Outstanding Papers 4

15:00–15:18 *Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme*
Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou and Bo Xu

15:19–15:37 *A Local Detection Approach for Named Entity Recognition and Mention Detection*
Mingbin Xu, Hui Jiang and Sedtawut Watcharawittayakul

15:38–15:56 *Vancouver Welcomes You! Minimalist Location Metonymy Resolution*
Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham and Nigel Collier

15:57–16:15 *Unifying Text, Metadata, and User Network Representations with a Neural Network for Geolocation Prediction*
Yasuhide Miura, Motoki Taniguchi, Tomoki Taniguchi and Tomoko Ohkuma

16:16–16:34 *Multi-Task Video Captioning with Video and Entailment Generation*
Ramakanth Pasunuru and Mohit Bansal

Monday, July 31st

18:00–21:30 Session P1: Poster Session 1

Enriching Complex Networks with Word Embeddings for Detecting Mild Cognitive Impairment from Speech Transcripts
Leandro Santos, Edilson Anselmo Corrêa Júnior, Osvaldo Oliveira Jr, Diego Amancio, Letícia Mansur and Sandra Aluísio

Adversarial Adaptation of Synthetic or Stale Data
Young-Bum Kim, Karl Stratos and Dongchan Kim

Chat Detection in an Intelligent Assistant: Combining Task-oriented and Non-task-oriented Spoken Dialogue Systems
Satoshi Akasaki and Nobuhiro Kaji

A Neural Local Coherence Model
Dat Tien Nguyen and Shafiq Joty

Data-Driven Broad-Coverage Grammars for Opinionated Natural Language Generation (ONLG)
Tomer Cagan, Stefan L. Frank and Reut Tsarfaty

Learning to Ask: Neural Question Generation for Reading Comprehension
Xinya Du, Junru Shao and Claire Cardie

Joint Optimization of User-desired Content in Multi-document Summaries by Learning from User Feedback
Avinesh PVS and Christian M. Meyer

Flexible and Creative Chinese Poetry Generation Using Neural Memory
Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang and Andi Zhang

Learning to Generate Market Comments from Stock Prices
Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura and Yusuke Miyao

Monday, July 31st (continued)

Can Syntax Help? Improving an LSTM-based Sentence Compression Model for New Domains
Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song and Lejian Liao

Transductive Non-linear Learning for Chinese Hypernym Prediction
Chengyu Wang, Junchi Yan, Aoying Zhou and Xiaofeng He

A Constituent-Centric Neural Architecture for Reading Comprehension
Pengtao Xie and Eric Xing

Cross-lingual Distillation for Text Classification
Ruochen Xu and Yiming Yang

Understanding and Predicting Empathic Behavior in Counseling Therapy
Verónica Pérez-Rosas, Rada Mihalcea, Kenneth Resnicow, Satinder Singh and Lawrence An

Leveraging Knowledge Bases in LSTMs for Improving Machine Reading
Bishan Yang and Tom Mitchell

Prerequisite Relation Learning for Concepts in MOOCs
Liangming Pan, Chengjiang Li, Juanzi Li and Jie Tang

Unsupervised Text Segmentation Based on Native Language Characteristics
Shervin Malmasi, Mark Dras, Mark Johnson, Lan Du and Magdalena Wolska

Weakly Supervised Cross-Lingual Named Entity Recognition via Effective Annotation and Representation Projection
Jian Ni, Georgiana Dinu and Radu Florian

Context Sensitive Lemmatization Using Two Successive Bidirectional Gated Recurrent Networks
Abhisek Chakrabarty, Onkar Arun Pandit and Utpal Garain

Learning to Create and Reuse Words in Open-Vocabulary Neural Language Modeling
Kazuya Kawakami, Chris Dyer and Phil Blunsom

Bandit Structured Prediction for Neural Sequence-to-Sequence Learning
Julia Kreutzer, Artem Sokolov and Stefan Riezler

Prior Knowledge Integration for Neural Machine Translation using Posterior Regularization
Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu and Maosong Sun

Incorporating Word Reordering Knowledge into Attention-based Neural Machine Translation
Jinchao Zhang, Mingxuan Wang, Qun Liu and Jie Zhou

Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search
Chris Hokamp and Qun Liu

Combating Human Trafficking with Multimodal Deep Models
Edmund Tong, Amir Zadeh, Cara Jones and Louis-Philippe Morency

MalwareTextDB: A Database for Annotated Malware Articles
Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu and Chen Hui Ong

A Corpus of Annotated Revisions for Studying Argumentative Writing
Fan Zhang, Homa B. Hashemi, Rebecca Hwa and Diane Litman

Automatic Induction of Synsets from a Graph of Synonyms
Dmitry Ustalov, Alexander Panchenko and Chris Biemann

Neural Modeling of Multi-Predicate Interactions for Japanese Predicate Argument Structure Analysis
Hiroki Ouchi, Hiroyuki Shindo and Yuji Matsumoto

TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension
Mandar Joshi, Eunsol Choi, Daniel Weld and Luke Zettlemoyer

Learning Semantic Correspondences in Technical Documentation
Kyle Richardson and Jonas Kuhn

Bridge Text and Knowledge by Learning Multi-Prototype Entity Mention Embedding
Yixin Cao, Lifu Huang, Heng Ji, Xu Chen and Juanzi Li

Interactive Learning of Grounded Verb Semantics towards Human-Robot Communication
Lanbo She and Joyce Chai

Multimodal Word Distributions
Ben Athiwaratkun and Andrew Wilson

Enhanced LSTM for Natural Language Inference
Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang and Diana Inkpen

Linguistic analysis of differences in portrayal of movie characters
Anil Ramakrishna, Victor R. Martínez, Nikolaos Malandrakis, Karan Singla and Shrikanth Narayanan

Linguistically Regularized LSTM for Sentiment Classification
Qiao Qian, Minlie Huang, Jinhao Lei and Xiaoyan Zhu

Sarcasm SIGN: Interpreting Sarcasm with Sentiment Based Monolingual Machine Translation
Lotem Peled and Roi Reichart

Active Sentiment Domain Adaptation
Fangzhao Wu, Yongfeng Huang and Jun Yan

Volatility Prediction using Financial Disclosures Sentiments with Word Embedding-based IR Models
Navid Rekabsaz, Mihai Lupu, Artem Baklanov, Alexander Dür, Linda Andersson and Allan Hanbury

CANE: Context-Aware Network Embedding for Relation Modeling
Cunchao Tu, Han Liu, Zhiyuan Liu and Maosong Sun

Universal Dependencies Parsing for Colloquial Singaporean English
Hongmin Wang, Yue Zhang, GuangYong Leonard Chan, Jie Yang and Hai Leong Chieu

Generic Axiomatization of Families of Noncrossing Graphs in Dependency Parsing
Anssi Yli-Jyrä and Carlos Gómez-Rodríguez

Semi-supervised sequence tagging with bidirectional language models
Matthew Peters, Waleed Ammar, Chandra Bhagavatula and Russell Power

Tuesday, August 1st

19:00–22:00 Session P2: Poster Session 2

Learning Symmetric Collaborative Dialogue Agents with Dynamic Knowledge Graph Embeddings
He He, Anusha Balakrishnan, Mihail Eric and Percy Liang

Neural Belief Tracker: Data-Driven Dialogue State Tracking
Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson and Steve Young

Exploiting Argument Information to Improve Event Detection via Supervised Attention Mechanisms
Shulin Liu, Yubo Chen, Kang Liu and Jun Zhao

Topical Coherence in LDA-based Models through Induced Segmentation
Hesam Amoualian, Wei Lu, Eric Gaussier, Georgios Balikas, Massih R Amini and Marianne Clausel

Jointly Extracting Relations with Class Ties via Effective Deep Ranking
Hai Ye, Wenhan Chao, Zhunchen Luo and Zhoujun Li

Search-based Neural Structured Learning for Sequential Question Answering
Mohit Iyyer, Wen-tau Yih and Ming-Wei Chang

Gated-Attention Readers for Text Comprehension
Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen and Ruslan Salakhutdinov

Determining Gains Acquired from Word Embedding Quantitatively Using Discrete Distribution Clustering
Jianbo Ye, Yanran Li, Zhaohui Wu, James Z. Wang, Wenjie Li and Jia Li

Towards a Seamless Integration of Word Senses into Downstream NLP Applications
Mohammad Taher Pilehvar, Jose Camacho-Collados, Roberto Navigli and Nigel Collier

Reading Wikipedia to Answer Open-Domain Questions
Danqi Chen, Adam Fisch, Jason Weston and Antoine Bordes

Learning to Skim Text
Adams Wei Yu, Hongrae Lee and Quoc Le

An Algebra for Feature Extraction
Vivek Srikumar

Chunk-based Decoder for Neural Machine Translation
Shonosuke Ishiwatari, Jingtao Yao, Shujie Liu, Mu Li, Ming Zhou, Naoki Yoshinaga, Masaru Kitsuregawa and Weijia Jia

Doubly-Attentive Decoder for Multi-modal Neural Machine Translation
Iacer Calixto, Qun Liu and Nick Campbell

A Teacher-Student Framework for Zero-Resource Neural Machine Translation
Yun Chen, Yang Liu, Yong Cheng and Victor O.K. Li

Improved Neural Machine Translation with a Syntax-Aware Encoder and Decoder
Huadong Chen, Shujian Huang, David Chiang and Jiajun Chen

Cross-lingual Name Tagging and Linking for 282 Languages
Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight and Heng Ji

Adversarial Training for Unsupervised Bilingual Lexicon Induction
Meng Zhang, Yang Liu, Huanbo Luan and Maosong Sun

Estimating Code-Switching on Twitter with a Novel Generalized Word-Level Language Detection Technique
Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali and Chandra Shekhar Maddila

Using Global Constraints and Reranking to Improve Cognates Detection
Michael Bloodgood and Benjamin Strauss

One-Shot Neural Cross-Lingual Transfer for Paradigm Completion
Katharina Kann, Ryan Cotterell and Hinrich Schütze

Morphological Inflection Generation with Hard Monotonic Attention
Roee Aharoni and Yoav Goldberg

Tuesday, August 1st (continued)

From Characters to Words to in Between: Do We Capture Morphology?
Clara Vania and Adam Lopez

Riemannian Optimization for Skip-Gram Negative Sampling
Alexander Fonarev, Oleksii Grinchuk, Gleb Gusev, Pavel Serdyukov and Ivan Oseledets

Deep Multitask Learning for Semantic Dependency Parsing
Hao Peng, Sam Thomson and Noah A. Smith

Improved Word Representation Learning with Sememes
Yilin Niu, Ruobing Xie, Zhiyuan Liu and Maosong Sun

Learning Character-level Compositionality with Visual Features
Frederick Liu, Han Lu, Chieh Lo and Graham Neubig

A Progressive Learning Approach to Chinese SRL Using Heterogeneous Data
Qiaolin Xia, Lei Sha, Baobao Chang and Zhifang Sui

Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings
John Wieting and Kevin Gimpel

Ontology-Aware Token Embeddings for Prepositional Phrase Attachment
Pradeep Dasigi, Waleed Ammar, Chris Dyer and Eduard Hovy

Identifying 1950s American Jazz Musicians: Fine-Grained IsA Extraction via Modifier Composition
Ellie Pavlick and Marius Pasca

Parsing to 1-Endpoint-Crossing, Pagenumber-2 Graphs
Junjie Cao, Sheng Huang, Weiwei Sun and Xiaojun Wan

Semi-supervised Multitask Learning for Sequence Labeling
Marek Rei

Semantic Parsing of Pre-university Math Problems
Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai and Noriko H. Arai

Adversarial Multi-task Learning for Text Classification

Pengfei Liu Xipeng Qiu Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pfliu14,xpqiu,xjhuang}@fudan.edu.cn

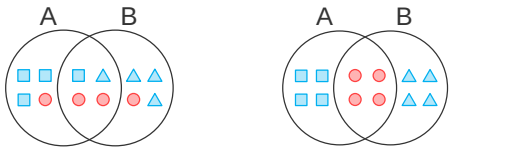
Abstract

Neural network models have shown their promising opportunities for multi-task learning, which focus on learning the shared layers to extract the common and task-invariant features. However, in most existing approaches, the extracted shared features are prone to be contaminated by task-specific features or the noise brought by other tasks. In this paper, we propose an adversarial multi-task learning framework, alleviating the shared and private latent feature spaces from interfering with each other. We conduct extensive experiments on 16 different text classification tasks, which demonstrates the benefits of our approach. Besides, we show that the shared knowledge learned by our proposed model can be regarded as off-the-shelf knowledge and easily transferred to new tasks. The datasets of all 16 tasks are publicly available at `http://nlp.fudan.edu.cn/data/`

1 Introduction

Multi-task learning is an effective approach to improve the performance of a single task with the help of other related tasks. Recently, neural-based models for multi-task learning have become very popular, ranging from computer vision (Misra et al., 2016; Zhang et al., 2014) to natural language processing (Collobert and Weston, 2008; Luong et al., 2015), since they provide a convenient way of combining information from multiple tasks.

However, most existing work on multi-task learning (Liu et al., 2016c,b) attempts to divide the features of different tasks into private and shared spaces, merely based on whether parameters of some components should be shared. As shown in Figure 1-(a), the general shared-private model introduces two feature spaces for any task: one is used to store task-dependent features, the other is used to capture shared features. The major limitation of this framework is that the shared feature space could contain some unnecessary task-specific features, while some sharable features could also be mixed in private space, suffering from feature redundancy.



(a) Shared-Private Model (b) Adversarial Shared-Private Model

Figure 1: Two sharing schemes for task A and task B. The overlap between two black circles denotes shared space. The blue triangles and boxes represent the task-specific features while the red circles denote the features which can be shared.

Taking the following two sentences as examples, which are extracted from two different sentiment classification tasks: Movie reviews and Baby products reviews.

*The **infantile** cart is simple and easy to use.*
*This kind of humour is **infantile** and boring.*

The word "`infantile`" indicates negative sentiment in Movie task while it is neutral in Baby task. However, the general shared-private model could place the task-specific word "`infantile`" in a shared space, leaving potential hazards for other tasks. Additionally, the capacity of shared space could also be wasted by some unnecessary features.

To address this problem, in this paper we propose an adversarial multi-task framework, in which the shared and private feature spaces are in-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1–10
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1001

herently disjoint by introducing orthogonality constraints. Specifically, we design a generic shared-private learning framework to model the text sequence. To prevent the shared and private latent feature spaces from interfering with each other, we introduce two strategies: adversarial training and orthogonality constraints. The adversarial training is used to ensure that the shared feature space simply contains common and task-invariant information, while the orthogonality constraint is used to eliminate redundant features from the private and shared spaces.

The contributions of this paper can be summarized as follows.

1. Proposed model divides the task-specific and shared space in a more precise way, rather than roughly sharing parameters.
2. We extend the original binary adversarial training to multi-class, which not only enables multiple tasks to be jointly trained, but allows us to utilize unlabeled data.
3. We can condense the shared knowledge among multiple tasks into an off-the-shelf neural layer, which can be easily transferred to new tasks.

2 Recurrent Models for Text Classification

There are many neural sentence models, which can be used for text modelling, involving recurrent neural networks (Sutskever et al., 2014; Chung et al., 2014; Liu et al., 2015a), convolutional neural networks (Collobert et al., 2011; Kalchbrenner et al., 2014), and recursive neural networks (Socher et al., 2013). Here we adopt recurrent neural network with long short-term memory (LSTM) due to their superior performance in various NLP tasks (Liu et al., 2016a; Lin et al., 2017).

Long Short-term Memory Long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (RNN) (Elman, 1990), and specifically addresses the issue of learning long-term dependencies. While there are numerous LSTM variants, here we use the LSTM architecture used by (Jozefowicz et al., 2015), which is similar to the architecture of (Graves, 2013) but without peep-hole connections.

We define the LSTM *units* at each time step t to be a collection of vectors in $\mathbb{R}^d$: an *input gate* $\mathbf{i}_t$, a *forget gate* $\mathbf{f}_t$, an *output gate* $\mathbf{o}_t$, a *memory cell* $\mathbf{c}_t$ and a hidden state $\mathbf{h}_t$. d is the number of the LSTM units. The elements of the gating vectors $\mathbf{i}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$ are in $[0, 1]$.

The LSTM is precisely specified as follows.

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W}_p \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b}_p \right), \quad (1)$$

$$\mathbf{c}_t = \tilde{\mathbf{c}}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (2)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (3)$$

where $\mathbf{x}_t \in \mathbb{R}^e$ is the input at the current time step; $\mathbf{W}_p \in \mathbb{R}^{4d \times (d+e)}$ and $\mathbf{b}_p \in \mathbb{R}^{4d}$ are parameters of affine transformation; σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication.

The update of each LSTM unit can be written precisely as follows:

$$\mathbf{h}_t = \mathbf{LSTM}(\mathbf{h}_{t-1}, \mathbf{x}_t, \theta_p). \quad (4)$$

Here, the function $\mathbf{LSTM}(\cdot, \cdot, \cdot, \cdot)$ is a shorthand for Eq. (1-3), and θ_p represents all the parameters of LSTM.

Text Classification with LSTM Given a text sequence $x = \{x_1, x_2, \cdots, x_T\}$, we first use a lookup layer to get the vector representation (embeddings) $\mathbf{x}_i$ of the each word x_i. The output at the last moment $\mathbf{h}_T$ can be regarded as the representation of the whole sequence, which has a fully connected layer followed by a softmax non-linear layer that predicts the probability distribution over classes.

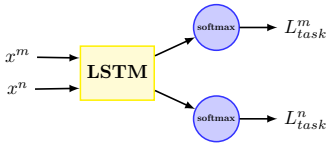
$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}\mathbf{h}_T + \mathbf{b}) \quad (5)$$

where $\hat{\mathbf{y}}$ is prediction probabilities, $\mathbf{W}$ is the weight which needs to be learned, $\mathbf{b}$ is a bias term.

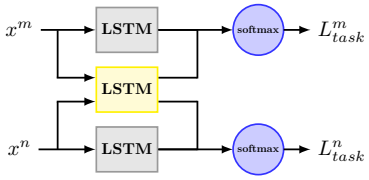
Given a corpus with N training samples (x_i, y_i), the parameters of the network are trained to minimise the cross-entropy of the predicted and true distributions.

$$L(\hat{y}, y) = -\sum_{i=1}^{N} \sum_{j=1}^{C} y_i^j \log(\hat{y}_i^j), \quad (6)$$

where y_i^j is the ground-truth label; $\hat{y}_i^j$ is prediction probabilities, and C is the class number.



(a) Fully Shared Model (FS-MTL)



(b) Shared-Private Model (SP-MTL)

Figure 2: Two architectures for learning multiple tasks. Yellow and gray boxes represent shared and private LSTM layers respectively.

3 Multi-task Learning for Text Classification

The goal of multi-task learning is to utilizes the correlation among these related tasks to improve classification by learning tasks in parallel. To facilitate this, we give some explanation for notations used in this paper. Formally, we refer to D_k as a dataset with N_k samples for task k. Specifically,

$$D_k = \{(x_i^k, y_i^k)\}_{i=1}^{N_k} \tag{7}$$

where x_i^k and y_i^k denote a sentence and corresponding label for task k.

3.1 Two Sharing Schemes for Sentence Modeling

The key factor of multi-task learning is the sharing scheme in latent feature space. In neural network based model, the latent features can be regarded as the states of hidden neurons. Specific to text classification, the latent features are the hidden states of LSTM at the end of a sentence. Therefore, the sharing schemes are different in how to group the shared features. Here, we first introduce two sharing schemes with multi-task learning: fully-shared scheme and shared-private scheme.

Fully-Shared Model (FS-MTL) In fully-shared model, we use a single shared LSTM layer to extract features for all the tasks. For example, given two tasks m and n, it takes the view that the features of task m can be totally shared by task n and vice versa. This model ignores the fact that some features are task-dependent. Figure 2a illustrates the fully-shared model.

Shared-Private Model (SP-MTL) As shown in Figure 2b, the shared-private model introduces two feature spaces for each task: one is used to store task-dependent features, the other is used to capture task-invariant features. Accordingly, we can see each task is assigned a private LSTM layer and shared LSTM layer. Formally, for any sentence in task k, we can compute its shared representation $\mathbf{s}_t^k$ and task-specific representation $\mathbf{h}_t^k$ as follows:

$$\mathbf{s}_t^k = \mathbf{LSTM}(x_t, \mathbf{s}_{t-1}^k, \theta_s), \tag{8}$$

$$\mathbf{h}_t^k = \mathbf{LSTM}(x_t, \mathbf{h}_{t-1}^m, \theta_k) \tag{9}$$

where $\mathbf{LSTM}(., \theta)$ is defined as Eq. (4).

The final features are concatenation of the features from private space and shared space.

3.2 Task-Specific Output Layer

For a sentence in task k, its feature $\mathbf{h}^{(k)}$, emitted by the deep muti-task architectures, is ultimately fed into the corresponding task-specific softmax layer for classification or other tasks.

The parameters of the network are trained to minimise the cross-entropy of the predicted and true distributions on all the tasks. The loss L_{task} can be computed as:

$$L_{Task} = \sum_{k=1}^{K} \alpha_k L(\hat{y}^{(k)}, y^{(k)}) \tag{10}$$

where α_k is the weights for each task k respectively. $L(\hat{y}, y)$ is defined as Eq. 6.

4 Incorporating Adversarial Training

Although the shared-private model separates the feature space into the shared and private spaces, there is no guarantee that sharable features can not exist in private feature space, or vice versa. Thus, some useful sharable features could be ignored in shared-private model, and the shared feature space is also vulnerable to contamination by some task-specific information.

Therefore, a simple principle can be applied into multi-task learning that a good shared feature space should contain more common information and no task-specific information. To address this problem, we introduce adversarial training into multi-task framework as shown in Figure 3 (ASP-MTL).

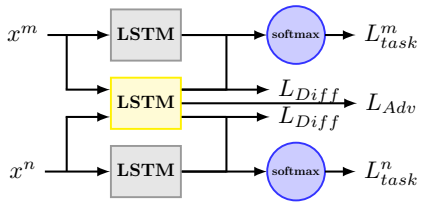


Figure 3: Adversarial shared-private model. Yellow and gray boxes represent shared and private LSTM layers respectively.

4.1 Adversarial Network

Adversarial networks have recently surfaced and are first used for generative model (Goodfellow et al., 2014). The goal is to learn a generative distribution $p_G(x)$ that matches the real data distribution $P_{data}(x)$ Specifically, GAN learns a generative network G and discriminative model D, in which G generates samples from the generator distribution $p_G(x)$. and D learns to determine whether a sample is from $p_G(x)$ or $P_{data}(x)$. This min-max game can be optimized by the following risk:

$$\phi = \min_G \max_D \Big(E_{x\sim P_{data}}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))] \Big) \quad (11)$$

While originally proposed for generating random samples, adversarial network can be used as a general tool to measure equivalence between distributions (Taigman et al., 2016). Formally, (Ajakan et al., 2014) linked the adversarial loss to the H-divergence between two distributions and successfully achieve unsupervised domain adaptation with adversarial network. Motivated by theory on domain adaptation (Ben-David et al., 2010, 2007; Bousmalis et al., 2016) that a transferable feature is one for which an algorithm cannot learn to identify the domain of origin of the input observation.

4.2 Task Adversarial Loss for MTL

Inspired by adversarial networks (Goodfellow et al., 2014), we proposed an adversarial shared-private model for multi-task learning, in which a shared recurrent neural layer is working adversarially towards a learnable multi-layer perceptron, preventing it from making an accurate prediction about the types of tasks. This adversarial training encourages shared space to be more pure and ensure the shared representation not be contaminated by task-specific features.

Task Discriminator Discriminator is used to map the shared representation of sentences into a probability distribution, estimating what kinds of tasks the encoded sentence comes from.

$$D(\mathbf{s}^k_T, \theta_D) = \text{softmax}(\mathbf{b} + \mathbf{U}\mathbf{s}^k_T) \quad (12)$$

where $\mathbf{U} \in \mathbb{R}^{d\times d}$ is a learnable parameter and $\mathbf{b} \in \mathbb{R}^d$ is a bias.

Adversarial Loss Different with most existing multi-task learning algorithm, we add an extra task adversarial loss L_{Adv} to prevent task-specific feature from creeping in to shared space. The task adversarial loss is used to train a model to produce shared features such that a classifier cannot reliably predict the task based on these features. The original loss of adversarial network is limited since it can only be used in binary situation. To overcome this, we extend it to multi-class form, which allow our model can be trained together with multiple tasks:

$$L_{Adv} = \min_{\theta_s} \left(\lambda \max_{\theta_D} \left(\sum_{k=1}^{K} \sum_{i=1}^{N_k} d^k_i \log[D(E(\mathbf{x}^k))] \right) \right) \quad (13)$$

where d^k_i denotes the ground-truth label indicating the type of the current task. Here, there is a min-max optimization and the basic idea is that, given a sentence, the shared LSTM generates a representation to mislead the task discriminator. At the same time, the discriminator tries its best to make a correct classification on the type of task. After the training phase, the shared feature extractor and task discriminator reach a point at which both cannot improve and the discriminator is unable to differentiate among all the tasks.

Semi-supervised Learning Multi-task Learning We notice that the L_{Adv} requires only the input sentence x and does not require the corresponding label y, which makes it possible to combine our model with semi-supervised learning. Finally, in this semi-supervised multi-task learning framework, our model can not only utilize the data from related tasks, but can employ abundant unlabeled corpora.

4.3 Orthogonality Constraints

We notice that there is a potential drawback of the above model. That is, the task-invariant features can appear both in shared space and private space.

Motivated by recently work(Jia et al., 2010; Salzmann et al., 2010; Bousmalis et al., 2016)

Dataset	Train	Dev.	Test	Unlab.	Avg. L	Vocab.
Books	1400	200	400	2000	159	62K
Elec.	1398	200	400	2000	101	30K
DVD	1400	200	400	2000	173	69K
Kitchen	1400	200	400	2000	89	28K
Apparel	1400	200	400	2000	57	21K
Camera	1397	200	400	2000	130	26K
Health	1400	200	400	2000	81	26K
Music	1400	200	400	2000	136	60K
Toys	1400	200	400	2000	90	28K
Video	1400	200	400	2000	156	57K
Baby	1300	200	400	2000	104	26K
Mag.	1370	200	400	2000	117	30K
Soft.	1315	200	400	475	129	26K
Sports	1400	200	400	2000	94	30K
IMDB	1400	200	400	2000	269	44K
MR	1400	200	400	2000	21	12K

Table 1: Statistics of the 16 datasets. The columns 2-5 denote the number of samples in training, development, test and unlabeled sets. The last two columns represent the average length and vocabulary size of corresponding dataset.

on shared-private latent space analysis, we introduce orthogonality constraints, which penalize redundant latent representations and encourages the shared and private extractors to encode different aspects of the inputs.

After exploring many optional methods, we find below loss is optimal, which is used by Bousmalis et al. (2016) and achieve a better performance:

$$L_{\text{diff}} = \sum_{k=1}^{K} \left\| \mathbf{S}^{k\top} \mathbf{H}^{k} \right\|_F^2 , \qquad (14)$$

where $\| \cdot \|_F^2$ is the squared Frobenius norm. $\mathbf{S}^k$ and $\mathbf{H}^k$ are two matrics, whose rows are the output of shared extractor $E_s(,;\theta_s)$ and task-specific extrator $E_k(,;\theta_k)$ of a input sentence.

4.4 Put It All Together

The final loss function of our model can be written as:

$$L = L_{Task} + \lambda L_{Adv} + \gamma L_{Diff} \qquad (15)$$

where λ and γ are hyper-parameter.

The networks are trained with backpropagation and this minimax optimization becomes possible via the use of a gradient reversal layer (Ganin and Lempitsky, 2015).

5 Experiment

5.1 Dataset

To make an extensive evaluation, we collect 16 different datasets from several popular review corpora.

The first 14 datasets are product reviews, which contain Amazon product reviews from different domains, such as Books, DVDs, Electronics, ect. The goal is to classify a product review as either positive or negative. These datasets are collected based on the raw data [1] provided by (Blitzer et al., 2007). Specifically, we extract the sentences and corresponding labels from the unprocessed original data [2]. The only preprocessing operation of these sentences is tokenized using the Stanford tokenizer [3].

The remaining two datasets are about movie reviews. The IMDB dataset[4] consists of movie reviews with binary classes (Maas et al., 2011). One key aspect of this dataset is that each movie review has several sentences. The MR dataset also consists of movie reviews from rotten tomato website with two classes [5](Pang and Lee, 2005).

All the datasets in each task are partitioned randomly into training set, development set and testing set with the proportion of 70%, 20% and 10% respectively. The detailed statistics about all the datasets are listed in Table 1.

5.2 Competitor Methods for Multi-task Learning

The multi-task frameworks proposed by previous works are various while not all can be applied to the tasks we focused. Nevertheless, we chose two most related neural models for multi-task learning and implement them as competitor methods.

- MT-CNN: This model is proposed by Collobert and Weston (2008) with convolutional layer, in which lookup-tables are shared partially while other layers are task-specific.

[1] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

[2] Blitzer et al. (2007) also provides two extra processed datasets with the format of Bag-of-Words, which are not proper for neural-based models.

[3] http://nlp.stanford.edu/software/tokenizer.shtml

[4] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/unprocessed.tar.gz

[5] https://www.cs.cornell.edu/people/pabo/movie-review-data/.

Task	Single Task				Multiple Tasks				
	LSTM	BiLSTM	sLSTM	Avg.	MT-DNN	MT-CNN	FS-MTL	SP-MTL	ASP-MTL
Books	20.5	19.0	18.0	19.2	$17.8_{(-1.4)}$	$15.5_{(-3.7)}$	$17.5_{(-1.7)}$	$18.8_{(-0.4)}$	$16.0_{(-3.2)}$
Electronics	19.5	21.5	23.3	21.4	$18.3_{(-3.1)}$	$16.8_{(-4.6)}$	$14.3_{(-7.1)}$	$15.3_{(-6.1)}$	$13.2_{(-8.2)}$
DVD	18.3	19.5	22.0	19.9	$15.8_{(-4.1)}$	$16.0_{(-3.9)}$	$16.5_{(-3.4)}$	$16.0_{(-3.9)}$	$14.5_{(-5.4)}$
Kitchen	22.0	18.8	19.5	20.1	$19.3_{(-0.8)}$	$16.8_{(-3.3)}$	$14.0_{(-6.1)}$	$14.8_{(-5.3)}$	$13.8_{(-6.3)}$
Apparel	16.8	14.0	16.3	15.7	$15.0_{(-0.7)}$	$16.3_{(+0.6)}$	$15.5_{(-0.2)}$	$13.5_{(-2.2)}$	$13.0_{(-2.7)}$
Camera	14.8	14.0	15.0	14.6	$13.8_{(-0.8)}$	$14.0_{(-0.6)}$	$13.5_{(-1.1)}$	$12.0_{(-2.6)}$	$10.8_{(-3.8)}$
Health	15.5	21.3	16.5	17.8	$14.3_{(-3.5)}$	$12.8_{(-5.0)}$	$12.0_{(-5.8)}$	$12.8_{(-5.0)}$	$11.8_{(-6.0)}$
Music	23.3	22.8	23.0	23.0	$15.3_{(-7.7)}$	$16.3_{(-6.7)}$	$18.8_{(-4.2)}$	$17.0_{(-6.0)}$	$17.5_{(-5.5)}$
Toys	16.8	15.3	16.8	16.3	$12.3_{(-4.0)}$	$10.8_{(-5.5)}$	$15.5_{(-0.8)}$	$14.8_{(-1.5)}$	$12.0_{(-4.3)}$
Video	18.5	16.3	16.3	17.0	$15.0_{(-2.0)}$	$18.5_{(+1.5)}$	$16.3_{(-0.7)}$	$16.8_{(-0.2)}$	$15.5_{(-1.5)}$
Baby	15.3	16.5	15.8	15.9	$12.0_{(-3.9)}$	$12.3_{(-3.6)}$	$12.0_{(-3.9)}$	$13.3_{(-2.6)}$	$11.8_{(-4.1)}$
Magazines	10.8	8.5	12.3	10.5	$10.5_{(+0.0)}$	$12.3_{(+1.8)}$	$7.5_{(-3.0)}$	$8.0_{(-2.5)}$	$7.8_{(-2.7)}$
Software	15.3	14.3	14.5	14.7	$14.3_{(-0.4)}$	$13.5_{(-1.2)}$	$13.8_{(-0.9)}$	$13.0_{(-1.7)}$	$12.8_{(-1.9)}$
Sports	18.3	16.0	17.5	17.3	$16.8_{(-0.5)}$	$16.0_{(-1.3)}$	$14.5_{(-2.8)}$	$12.8_{(-4.5)}$	$14.3_{(-3.0)}$
IMDB	18.3	15.0	18.5	17.3	$16.8_{(-0.5)}$	$13.8_{(-3.5)}$	$17.5_{(+0.2)}$	$15.3_{(-2.0)}$	$14.5_{(-2.8)}$
MR	27.3	25.3	28.0	26.9	$24.5_{(-2.4)}$	$25.5_{(-1.4)}$	$25.3_{(-1.6)}$	$24.0_{(-2.9)}$	$23.3_{(-3.6)}$
AVG	18.2	17.4	18.3	18.0	$15.7_{(-2.2)}$	$15.5_{(-2.5)}$	$15.3_{(-2.7)}$	$14.9_{(-3.1)}$	$13.9_{(-4.1)}$

Table 2: Error rates of our models on 16 datasets against typical baselines. The numbers in brackets represent the improvements relative to the average performance (Avg.) of three single task baselines.

- MT-DNN: The model is proposed by Liu et al. (2015b) with bag-of-words input and multi-layer perceptrons, in which a hidden layer is shared.

5.3 Hyperparameters

The word embeddings for all of the models are initialized with the 200d GloVe vectors ((Pennington et al., 2014)). The other parameters are initialized by randomly sampling from uniform distribution in $[-0.1, 0.1]$. The mini-batch size is set to 16.

For each task, we take the hyperparameters which achieve the best performance on the development set via an small grid search over combinations of the initial learning rate $[0.1, 0.01]$, $\lambda \in [0.01, 0.1]$, and $\gamma \in [0.01, 0.1]$. Finally, we chose the learning rate as 0.01, λ as 0.05 and γ as 0.01.

5.4 Performance Evaluation

Table 2 shows the error rates on 16 text classification tasks. The column of "Single Task" shows the results of vanilla LSTM, bidirectional LSTM (BiLSTM), stacked LSTM (sLSTM) and the average error rates of previous three models. The column of "Multiple Tasks" shows the results achieved by corresponding multi-task models. From this table, we can see that the performance of most tasks can be improved with a large margin with the help of multi-task learning, in which our model achieves the lowest error rates. More concretely, compared with SP-MTL, ASP-MTL achieves 4.1% average improvement surpassing SP-MTL with 1.0%, which indicates the importance of adversarial learning. It is noteworthy that for FS-MTL, the performances of some tasks are degraded, since this model puts all private and shared information into a unified space.

5.5 Shared Knowledge Transfer

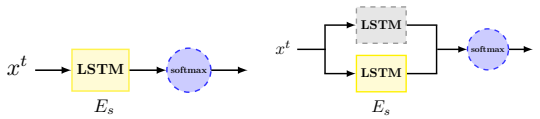
With the help of adversarial learning, the shared feature extractor E_s can generate more pure task-invariant representations, which can be considered as off-the-shelf knowledge and then be used for unseen new tasks.

To test the transferability of our learned shared extractor, we also design an experiment, in which we take turns choosing 15 tasks to train our model M_S with multi-task learning, then the learned shared layer are transferred to a second network M_T that is used for the remaining one task. The parameters of transferred layer are **kept frozen**, and the rest of parameters of the network M_T are randomly initialized.

More formally, we investigate two mechanisms towards the transferred shared extractor. As shown in Figure 4. The first one Single Channel (SC) model consists of one shared feature extractor E_s from M_S, then the extracted representation will be sent to an output layer. By contrast, the Bi-Channel (BC) model introduces an extra LSTM layer to encode more task-specific information. To evaluate the effectiveness of our introduced adversarial training framework, we also make a compar-

Source Tasks	Single Task				Transfer Models			
	LSTM	BiLSTM	sLSTM	Avg.	SP-MTL-SC	SP-MTL-BC	ASP-MTL-SC	ASP-MTL-BC
ϕ (Books)	20.5	19.0	18.0	19.2	$17.8_{(-1.4)}$	$16.3_{(-2.9)}$	$16.8_{(-2.4)}$	$16.3_{(-2.9)}$
ϕ (Electronics)	19.5	21.5	23.3	21.4	$15.3_{(-6.1)}$	$14.8_{(-6.6)}$	$17.8_{(-3.6)}$	$16.8_{(-4.6)}$
ϕ (DVD)	18.3	19.5	22.0	19.9	$14.8_{(-5.1)}$	$15.5_{(-4.4)}$	$14.5_{(-5.4)}$	$14.3_{(-5.6)}$
ϕ (Kitchen)	22.0	18.8	19.5	20.1	$15.0_{(-5.1)}$	$16.3_{(-3.8)}$	$16.3_{(-3.8)}$	$15.0_{(-5.1)}$
ϕ (Apparel)	16.8	14.0	16.3	15.7	$14.8_{(-0.9)}$	$12.0_{(-3.7)}$	$12.5_{(-3.2)}$	$13.8_{(-1.9)}$
ϕ (Camera)	14.8	14.0	15.0	14.6	$13.3_{(-1.3)}$	$12.5_{(-2.1)}$	$11.8_{(-2.8)}$	$10.3_{(-4.3)}$
ϕ (Health)	15.5	21.3	16.5	17.8	$14.5_{(-3.3)}$	$14.3_{(-3.5)}$	$12.3_{(-5.5)}$	$13.5_{(-4.3)}$
ϕ (Music)	23.3	22.8	23.0	23.0	$20.0_{(-3.0)}$	$17.8_{(-5.2)}$	$17.5_{(-5.5)}$	$18.3_{(-4.7)}$
ϕ (Toys)	16.8	15.3	16.8	16.3	$13.8_{(-2.5)}$	$12.5_{(-3.8)}$	$13.0_{(-3.3)}$	$11.8_{(-4.5)}$
ϕ (Video)	18.5	16.3	16.3	17.0	$14.3_{(-2.7)}$	$15.0_{(-2.0)}$	$14.8_{(-2.2)}$	$14.8_{(-2.2)}$
ϕ (Baby)	15.3	16.5	15.8	15.9	$16.5_{(+0.6)}$	$16.8_{(+0.9)}$	$13.5_{(-2.4)}$	$12.0_{(-3.9)}$
ϕ (Magazines)	10.8	8.5	12.3	10.5	$10.5_{(+0.0)}$	$10.3_{(-0.2)}$	$8.8_{(-1.7)}$	$9.5_{(-1.0)}$
ϕ (Software)	15.3	14.3	14.5	14.7	$13.0_{(-1.7)}$	$12.8_{(-1.9)}$	$14.5_{(-0.2)}$	$11.8_{(-2.9)}$
ϕ (Sports)	18.3	16.0	17.5	17.3	$16.3_{(-1.0)}$	$16.3_{(-1.0)}$	$13.3_{(-4.0)}$	$13.5_{(-3.8)}$
ϕ (IMDB)	18.3	15.0	18.5	17.3	$12.8_{(-4.5)}$	$12.8_{(-4.5)}$	$12.5_{(-4.8)}$	$13.3_{(-4.0)}$
ϕ (MR)	27.3	25.3	28.0	26.9	$26.0_{(-0.9)}$	$26.5_{(-0.4)}$	$24.8_{(-2.1)}$	$23.5_{(-3.4)}$
AVG	18.2	17.4	18.3	18.0	$15.6_{(-2.4)}$	$15.2_{(-2.8)}$	$14.7_{(-3.3)}$	$14.3_{(-3.7)}$

Table 3: Error rates of our models on 16 datasets against vanilla multi-task learning. ϕ (Books) means that we transfer the knowledge of the other 15 tasks to the target task Books.



(a) Single Channel (b) Bi-Channel

Figure 4: Two transfer strategies using a pre-trained shared LSTM layer. Yellow box denotes shared feature extractor E_s trained by 15 tasks.

ison with vanilla multi-task learning method.

Results and Analysis As shown in Table 3, we can see the shared layer from ASP-MTL achieves a better performance compared with SP-MTL. Besides, for the two kinds of transfer strategies, the Bi-Channel model performs better. The reason is that the task-specific layer introduced in the Bi-Channel model can store some private features. Overall, the results indicate that we can save the existing knowledge into a shared recurrent layer using adversarial multi-task learning, which is quite useful for a new task.

5.6 Visualization

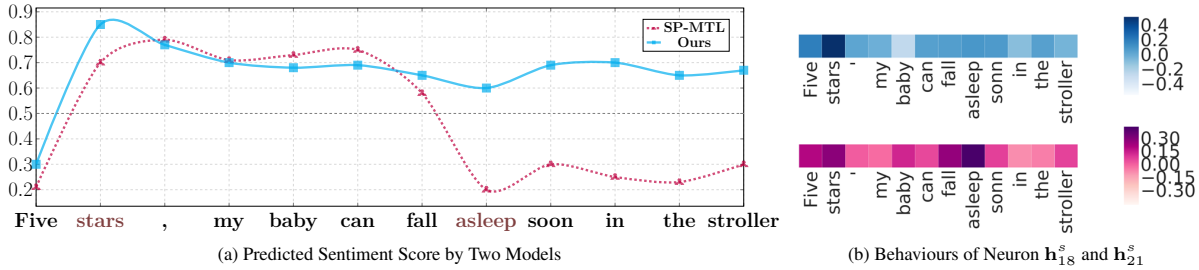
To get an intuitive understanding of how the introduced orthogonality constraints worked compared with vanilla shared-private model, we design an experiment to examine the behaviors of neurons from private layer and shared layer. More concretely, we refer to h_{tj} as the activation of the j-neuron at time step t, where $t \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, d\}$. By visualizing the hidden state $\mathbf{h}_j$ and analyzing the maximum activation, we can find what kinds of patterns the current neuron focuses on.

Figure 5 illustrates this phenomenon. Here, we randomly sample a sentence from the validation set of Baby task and analyze the changes of the predicted sentiment score at different time steps, which are obtained by SP-MTL and our proposed model. Additionally, to get more insights into how neurons in shared layer behave diversely towards different input word, we visualize the activation of two typical neurons. For the positive sentence "`Five stars, my baby can fall asleep soon in the stroller`", both models capture the informative pattern "`Five stars`" [6]. However, SP-MTL makes a wrong prediction due to misunderstanding of the word "`asleep`".

By contrast, our model makes a correct prediction and the reason can be inferred from the activation of Figure 5-(b), where the shared layer of SP-MTL is so sensitive that many features related to other tasks are included, such as "`asleep`", which misleads the final prediction. This indicates the importance of introducing adversarial learning to prevent the shared layer from being contaminated by task-specific features.

We also list some typical patterns captured by

[6]For this case, the vanilla LSTM also give a wrong answer due to ignoring the feature "`Five stars`".



(a) Predicted Sentiment Score by Two Models

(b) Behaviours of Neuron $\mathbf{h}^s_{18}$ and $\mathbf{h}^s_{21}$

Figure 5: (a) The change of the predicted sentiment score at different time steps. Y-axis represents the sentiment score, while X-axis represents the input words in chronological order. The darker grey horizontal line gives a border between the positive and negative sentiments. (b) The purple heat map describes the behaviour of neuron $\mathbf{h}^s_{18}$ from shared layer of SP-MTL, while the blue one is used to show the behaviour of neuron $\mathbf{h}^s_{21}$, which belongs to the shared layer of our model.

Model	Shared Layer	Task-Movie	Task-Baby
SP-MTL	good, great bad, love, simple, cut, slow, cheap, infantile	good, great, well-directed, pointless, cut, cheap, infantile	love, bad, cute, safety, mild, broken simple
ASP-MTL	good, great, love, bad poor	well-directed, pointless, cut, cheap, infantile	cute, safety, mild, broken simple

Table 4: Typical patterns captured by shared layer and task-specific layer of SP-MTL and ASP-MTL models on Movie and Baby tasks.

neurons from shared layer and task-specific layer in Table 4, and we have observed that: 1) for SP-MTL, if some patterns are captured by task-specific layer, they are likely to be placed into shared space. Clearly, suppose we have many tasks to be trained jointly, the shared layer bear much pressure and must sacrifice substantial amount of capacity to capture the patterns they actually do not need. Furthermore, some typical task-invariant features also go into task-specific layer. 2) for ASP-MTL, we find the features captured by shared and task-specific layer have a small amount of intersection, which allows these two kinds of layers can work effectively.

6 Related Work

There are two threads of related work. One thread is multi-task learning with neural network. Neural networks based multi-task learning has been proven effective in many NLP problems (Collobert and Weston, 2008; Glorot et al., 2011).

Liu et al. (2016c) first utilizes different LSTM layers to construct multi-task learning framwork for text classification. Liu et al. (2016b) proposes a generic multi-task framework, in which different tasks can share information by an external memory and communicate by a reading/writing mechanism. These work has potential limitation of just learning a shared space solely on sharing parameters, while our model introduce two strategies to learn the clear and non-redundant shared-private space.

Another thread of work is adversarial network. Adversarial networks have recently surfaced as a general tool measure equivalence between distributions and it has proven to be effective in a variety of tasks. Ajakan et al. (2014); Bousmalis et al. (2016) applied adverarial training to domain adaptation, aiming at transferring the knowledge of one source domain to target domain. Park and Im (2016) proposed a novel approach for multi-modal representation learning which uses adversarial back-propagation concept.

Different from these models, our model aims to find task-invariant sharable information for multiple related tasks using adversarial training strategy. Moreover, we extend binary adversarial training to multi-class, which enable multiple tasks to be jointly trained.

7 Conclusion

In this paper, we have proposed an adversarial multi-task learning framework, in which the task-specific and task-invariant features are learned non-redundantly, therefore capturing the shared-private separation of different tasks. We have demonstrated the effectiveness of our approach by applying our model to 16 different text classification tasks. We also perform extensive qualitative

analysis, deriving insights and indirectly explaining the quantitative improvements in the overall performance.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and thank Kaiyu Qian, Gang Niu for useful discussions. This work was partially funded by National Natural Science Foundation of China (No. 61532011 and 61672162), the National High Technology Research and Development Program of China (No. 2015AA015408), Shanghai Municipal Science and Technology Commission (No. 16JC1420401).

References

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. 2014. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446* .

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning* 79(1-2):151–175.

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. 2007. Analysis of representations for domain adaptation. *Advances in neural information processing systems* 19:137.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*. volume 7, pages 440–447.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*. pages 343–351.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The JMLR* 12:2493–2537.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. pages 1180–1189.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 513–520.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. pages 2672–2680.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

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

Yangqing Jia, Mathieu Salzmann, and Trevor Darrell. 2010. Factorized latent spaces with structured sparsity. In *Advances in Neural Information Processing Systems*. pages 982–990.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of The 32nd International Conference on Machine Learning*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* .

Pengfe Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion LSTMs for text semantic matching. In *Proceedings of ACL*.

PengFei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015a. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on EMNLP*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016b. Deep multi-task learning with shared memory. In *Proceedings of EMNLP*.

PengFei Liu, Xipeng Qiu, and Xuanjing Huang. 2016c. Recurrent neural network for text classification with multi-task learning. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015b. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL*.

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

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the ACL*. pages 142–150.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3994–4003.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 115–124.

Gwangbeen Park and Woobin Im. 2016. Image-text multi-modal representation learning by adversarial backpropagation. *arXiv preprint arXiv:1612.08354* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the EMNLP* 12:1532–1543.

Mathieu Salzmann, Carl Henrik Ek, Raquel Urtasun, and Trevor Darrell. 2010. Factorized orthogonal latent spaces. In *AISTATS*. pages 701–708.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in NIPS*. pages 3104–3112.

Yaniv Taigman, Adam Polyak, and Lior Wolf. 2016. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200* .

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. 2014. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*. Springer, pages 94–108.

Neural End-to-End Learning for Computational Argumentation Mining

Steffen Eger[†‡], **Johannes Daxenberger**[†], **Iryna Gurevych**[†‡]
[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universitt Darmstadt
[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research and Educational Information
http://www.ukp.tu-darmstadt.de

Abstract

We investigate neural techniques for end-to-end computational argumentation mining (AM). We frame AM both as a token-based dependency parsing and as a token-based sequence tagging problem, including a multi-task learning setup. Contrary to models that operate on the argument component level, we find that framing AM as dependency parsing leads to subpar performance results. In contrast, less complex (local) tagging models based on BiLSTMs perform robustly across classification scenarios, being able to catch long-range dependencies inherent to the AM problem. Moreover, we find that jointly learning 'natural' subtasks, in a multi-task learning setup, improves performance.

1 Introduction

Computational argumentation mining (AM) deals with finding argumentation structures in text. This involves several subtasks, such as: (a) separating argumentative units from non-argumentative units, also called 'component segmentation'; (b) classifying argument components into classes such as "Premise" or "Claim"; (c) finding relations between argument components; (d) classifying relations into classes such as "Support" or "Attack" (Persing and Ng, 2016; Stab and Gurevych, 2017).

Thus, AM would have to detect claims and premises (reasons) in texts such as the following, where premise P supports claim C:

> Since it killed many marine lives$_P$, tourism has threatened nature$_C$.

Argument structures in real texts are typically much more complex, cf. Figure 1.

While different research has addressed different subsets of the AM problem (see below), the ultimate goal is to solve all of them, starting from unannotated plain text. Two recent approaches to this end-to-end learning scenario are Persing and Ng (2016) and Stab and Gurevych (2017). Both solve the end-to-end task by first training independent models for each subtask and then defining an integer linear programming (ILP) model that encodes global constraints such as that each premise has a parent, etc. Besides their pipeline architecture the approaches also have in common that they heavily rely on hand-crafted features.

Hand-crafted features pose a problem because AM is to some degree an "arbitrary" problem in that the notion of "argument" critically relies on the underlying argumentation theory (Reed et al., 2008; Biran and Rambow, 2011; Habernal and Gurevych, 2015; Stab and Gurevych, 2017). Accordingly, datasets typically differ with respect to their annotation of (often rather complex) argument structure. Thus, feature sets would have to be manually adapted to and designed for each new sample of data, a challenging task. The same critique applies to the designing of ILP constraints. Moreover, from a machine learning perspective, pipeline approaches are problematic because they solve subtasks independently and thus lead to error propagation rather than exploiting interrelationships between variables. In contrast to this, we investigate *neural* techniques for end-to-end learning in computational AM, which do not require the hand-crafting of features or constraints. The models we survey also all capture some notion of "joint"—rather than "pipeline"—learning. We investigate several approaches.

First, we frame the end-to-end AM problem as a dependency parsing problem. Dependency parsing may be considered a natural choice for AM, because argument structures often form trees,

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 11–22
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1002

or closely resemble them (see §3). Hence, it is not surprising that 'discourse parsing' (Muller et al., 2012) has been suggested for AM (Peldszus and Stede, 2015). What distinguishes our approach from these previous ones is that we operate on the *token* level, rather than on the level of components, because we address the end-to-end framework and, thus, do not assume that non-argumentative units have already been sorted out and/or that the boundaries of argumentative units are given.

Second, we frame the problem as a sequence tagging problem. This is a natural choice especially for component identification (segmentation and classification), which is a typical entity recognition problem for which BIO tagging is a standard approach, pursued in AM, e.g., by Habernal and Gurevych (2016). The challenge in the end-to-end setting is to also include relations into the tagging scheme, which we realize by coding the distances between linked components into the tag label. Since related entities in AM are oftentimes several dozens of tokens apart from each other, neural sequence tagging models are in principle ideal candidates for such a framing because they can take into account *long-range dependencies*—something that is inherently difficult to capture with traditional feature-based tagging models such as conditional random fields (CRFs).

Third, we frame AM as a *multi-task* (tagging) problem (Caruana, 1997; Collobert and Weston, 2008). We experiment with subtasks of AM—e.g., component identification—as auxiliary tasks and investigate whether this improves performance on the AM problem. Adding such subtasks can be seen as analogous to de-coupling, e.g., component identification from the full AM problem.

Fourth, we evaluate the model of Miwa and Bansal (2016) that combines sequential (entity) and tree structure (relation) information and is in principle applicable to any problem where the aim is to extract entities and their relations. As such, this model makes fewer assumptions than our dependency parsing and tagging approaches.

The contributions of this paper are as follows. (1) We present the *first neural end-to-end* solutions to computational AM. (2) We show that several of them perform better than the state-of-the-art joint ILP model. (3) We show that a framing of AM as a token-based dependency parsing problem is ineffective—in contrast to what has been proposed for systems that operate on the coarser component level and that (4) a standard neural sequence tagging model that encodes distance information between components performs robustly in different environments. Finally, (5) we show that a multi-task learning setup where natural subtasks of the full AM problem are added as auxiliary tasks improves performance.[1]

2 Related Work

AM has applications in legal decision making (Palau and Moens, 2009; Moens et al., 2007), document summarization, and the analysis of scientific papers (Kirschner et al., 2015). Its importance for the educational domain has been highlighted by recent work on writing assistance (Zhang and Litman, 2016) and essay scoring (Persing and Ng, 2015; Somasundaran et al., 2016).

Most works on AM address subtasks of AM such as locating/classifying components (Florou et al., 2013; Moens et al., 2007; Rooney et al., 2012; Knight et al., 2003; Levy et al., 2014; Rinott et al., 2015). Relatively few works address the full AM problem of component *and* relation identification. Peldszus and Stede (2016) present a corpus of microtexts containing only argumentatively relevant text of controlled complexity. To our best knowledge, Stab and Gurevych (2017) created the only corpus of attested high quality which annotates the AM problem in its entire complexity: it contains token-level annotations of components, their types, as well as relations and their types.

3 Data

We use the dataset of persuasive essays (PE) from Stab and Gurevych (2017), which contains student essays written in response to controversial topics such as "competition or cooperation—which is better?"

	Train	Test
Essays	322	80
Paragraphs	1786	449
Tokens	118648	29538

Table 1: Corpus statistics

As Table 1 details, the corpus consists of 402 essays, 80 of which are reserved for testing. The an-

[1] Scripts that document how we ran our experiments are available from https://github.com/UKPLab/acl2017-neural_end2end_AM.

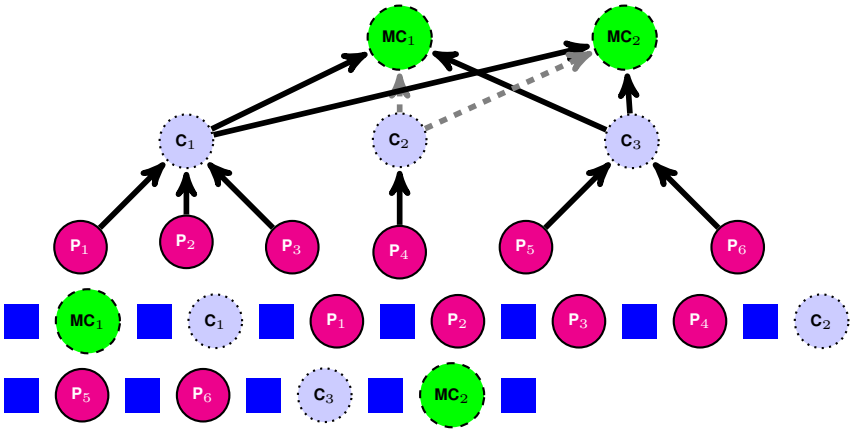


Figure 1: Bottom: Linear argumentation structure in a student essay. The essay is comprised of non-argumentative units (square) and argumentative units of different types: Premises (P), claims (C) and major claims (MC). Top: Relationsships between argumentative units. Solid arrows are support (for), dashed arrows are attack (against).

notation distinguishes between **major claims** (the central position of an author with respect to the essay's topic), **claims** (controversial statements that are either *for* or *against* the major claims), and **premises**, which give reasons for claims or other premises and either *support* or *attack* them. Overall, there are 751 major claims, 1506 claims, and 3832 premises. There are 5338 relations, most of which are supporting relations (>90%).

The corpus has a special structure, illustrated in Figure 1. First, major claims relate to no other components. Second, claims always relate to all other major claims.[2] Third, each premise relates to exactly one claim or premise. Thus, the argument structure in each essay is—almost—a tree. Since there may be several major claims, each claim potentially connects to multiple targets, violating the tree structure. This poses no problem, however, since we can "loss-lessly" re-link the claims to one of the major claims (e.g., the last major claim in a document) and create a special root node to which the major claims link. From this tree, the actual graph can be uniquely reconstructed.

There is another peculiarity of this data. Each essay is divided into paragraphs, of which there are 2235 in total. The argumentation structure is completely contained within a paragraph, except, possibly, for the relation from claims to major claims. Paragraphs have an average length of 66 tokens and are therefore much shorter than essays, which have an average length of 368 tokens. Thus, prediction on the paragraph level is easier than prediction on the essay level, because there are fewer components in a paragraph and hence fewer possibilities of source and target components in argument relations. The same is true for component classification: a paragraph can never contain premises only, for example, since premises link to other components.

[2] All MCs are considered as equivalent in meaning.

4 Models

This section describes our neural network framings for end-to-end AM.

Sequence Tagging is the problem of assigning each element in a stream of input tokens a label. In a neural context, the natural choice for tagging problems are recurrent neural nets (RNNs) in which a hidden vector representation $\mathbf{h}_t$ at time point t depends on the previous hidden vector representation $\mathbf{h}_{t-1}$ and the input $\mathbf{x}_t$. In this way, an infinite window ("long-range dependencies") around the current input token $\mathbf{x}_t$ can be taken into account when making an output prediction $\mathbf{y}_t$. We choose particular RNNs, namely, LSTMs (Hochreiter and Schmidhuber, 1997), which are popular for being able to address vanishing/exploding gradients problems. In addition to considering a left-to-right flow of information, bidirectional LSTMs (BL) also capture information to the right of the current input token.

The most recent generation of neural tagging models add label dependencies to BLs, so that successive output decisions are not made independently. This class of models is called BiLSTM-

CRF (BLC) (Huang et al., 2015). The model of Ma and Hovy (2016) adds convolutional neural nets (CNNs) on the character-level to BiLSTM-CRFs, leading to BiLSTM-CRF-CNN (BLCC) models. The character-level CNN may address problems of out-of-vocabulary words, that is, words not seen during training.

AM as Sequence Tagging: We frame AM as the following sequence tagging problem. Each input token has an associated label from $\mathcal{Y}$, where

$$\begin{aligned}\mathcal{Y} = \{(b,t,d,s) \,|\, & b \in \{\text{B},\text{I},\text{O}\}, t \in \{\text{P},\text{C},\text{MC},\perp\}, \\ & d \in \{\ldots,-2,-1,1,2,\ldots,\perp\}, \\ & s \in \{\text{Supp},\text{Att},\text{For},\text{Ag},\perp\}\}.\end{aligned} \quad (1)$$

In other words, $\mathcal{Y}$ consists of all four-tuples (b,t,d,s) where b is a BIO encoding indicating whether the current token is non-argumentative (O) or begins (B) or continues (I) a component; t indicates the *type* of the component (claim C, premise P, or major claim MC for our data). Moreover, d encodes the distance—measured in number of components—between the current component and the component it relates to. We encode the same d value for each token in a given component. Finally, s is the relation type ("stance") between two components and its value may be Support (Supp), Attack (Att), or For or Against (Ag). We also have a special symbol $\perp$ that indicates when a particular slot is not filled: e.g., a non-argumentative unit ($b = \text{O}$) has neither component type, nor relation, nor relation type. We refer to this framing as $\texttt{STag}_T$ (for "*S*imple *Tag*ging"), where T refers to the tagger used. For the example from §1, our coding would hence be:

Since (O,⊥,⊥,⊥)	it (B,P,1,Supp)	killed (I,P,1,Supp)	many (I,P,1,Supp)
marine (I,P,1,Supp)	lives (I,P,1,Supp)	, (O,⊥,⊥,⊥)	tourism (B,C,⊥,For)
has (I,C,⊥,For)	threatened (I,C,⊥,For)	nature (I,C,⊥,For)	. (O,⊥, ⊥, ⊥)

While the size of the label set $\mathcal{Y}$ is potentially infinite, we would expect it to be finite even in a potentially infinitely large data set, because humans also have only finite memory and are therefore expected to keep related components close in textual space. Indeed, as Figure 2 shows, in our PE essay data set about 30% of all relations between components have distance -1, that is, they follow the claim or premise that they attach to. Overall, around 2/3 of all relation distances d lie in $\{-2,-1,1\}$. However, the figure also illustrates that there are indeed long-range dependencies: distance values between -11 and $+10$ are observed in the data.

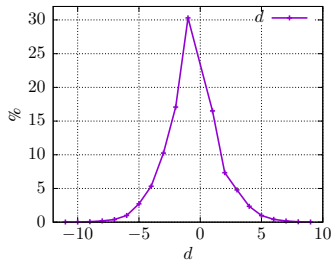


Figure 2: Distribution of distances d between components in PE dataset.

Multi-Task Learning Recently, there has been a lot of interest in so-called multi-task learning (MTL) scenarios, where several tasks are learned *jointly* (Søgaard and Goldberg, 2016; Peng and Dredze, 2016; Yang et al., 2016; Rusu et al., 2016; Héctor and Plank, 2017). It has been argued that such learning scenarios are closer to human learning because humans often transfer knowledge between several domains/tasks. In a neural context, MTL is typically implemented via weight sharing: several tasks are trained in the same network architecture, thereby sharing a substantial portion of network's parameters. This forces the network to learn *generalized* representations.

In the MTL framework of Søgaard and Goldberg (2016) the underlying model is a BiLSTM with several hidden layers. Then, given different tasks, each task k 'feeds' from one of the hidden layers in the network. In particular, the hidden states encoded in a specific layer are fed into a multiclass classifier f_k. The same work has demonstrated that this MTL protocol may be successful when there is a hierarchy between tasks and 'lower' tasks feed from lower layers.

AM as MTL: We use the same framework $\texttt{STag}_T$ for modeling AM as MTL. However, we in addition train auxiliary tasks in the network—each with a distinct label set $\mathcal{Y}'$.

Dependency Parsing methods can be classified into graph-based and transition-based approaches (Kiperwasser and Goldberg, 2016). *Transition-based* parsers encode the parsing problem as a sequence of configurations which may be modified by application of actions such as shift, reduce,

etc. The system starts with an initial configuration in which sentence elements are on a *buffer* and a *stack*, and a classifier successively decides which action to take next, leading to different configurations. The system terminates after a finite number of actions, and the parse tree is read off the terminal configuration. *Graph-based* parsers solve a structured prediction problem in which the goal is learning a scoring function over dependency trees such that correct trees are scored above all others.

Traditional dependency parsers used hand-crafted feature functions that look at "core" elements such as "word on top of the stack", "POS of word on top of the stack", and conjunctions of core features such as "word is X and POS is Y" (see McDonald et al. (2005)). Most neural parsers have not entirely abandoned feature engineering. Instead, they rely, for example, on encoding the core features of parsers as low-dimensional embedding vectors (Chen and Manning, 2014) but ignore feature combinations. Kiperwasser and Goldberg (2016) design a neural parser that uses only four features: the BiLSTM vector representations of the top 3 items on the stack and the first item on the buffer. In contrast, Dyer et al. (2015)'s neural parser associates each stack with a "stack LSTM" that encodes their contents. Actions are chosen based on the stack LSTM representations of the stacks, and no more feature engineering is necessary. Moreover, their parser has thus access to any part of the input, its history and stack contents.

AM as Dependency Parsing: To frame a problem as a dependency parsing problem, each instance of the problem must be encoded as a directed tree, where tokens have heads, which in turn are labeled. For end-to-end AM, we propose the framing illustrated in Figure 3. We highlight two design decisions, the remaining are analogous and/or can be read off the figure.

- The head of each non-argumentative text token is the document terminating token `END`, which is a punctuation mark in all our cases. The label of this link is O, the symbol for non-argumentative units.
- The head of each token in a premise is the *first* token of the claim or premise that it links to. The label of each of these links is $(b, \mathrm{P}, \mathrm{Supp})$ or $(b, \mathrm{P}, \mathrm{Att})$ depending on whether a premise "supports" or "attacks" a claim or premise; $b \in \{\mathrm{B}, \mathrm{I}\}$.

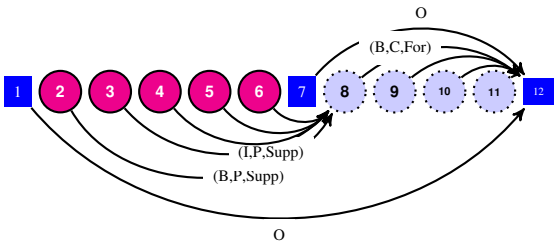


Figure 3: Dependency representation of sample sentence from §1. Links and selected labels.

LSTM-ER Miwa and Bansal (2016) present a neural end-to-end system for identifying both entities as well as relations between them. Their entity detection system is a BLC-type tagger and their relation detection system is a neural net that predicts a relation for each pair of detected entities. This relation module is a TreeLSTM model that makes use of dependency tree information. In addition to de-coupling entity and relation detection but jointly modeling them,[3] pretraining on entities and scheduled sampling (Bengio et al., 2015) is applied to prevent low performance at early training stages of entity detection and relation classification. To adapt LSTM-ER for the argument structure encoded in the PE dataset, we model three types of entities (premise, claim, major claim) and four types of relations (for, against, support, attack).

We use the **feature-based ILP model** from Stab and Gurevych (2017) as a comparison system. This system solves the subtasks of AM—component segmentation, component classification, relation detection and classification—independently. Afterwards, it defines an ILP model with various constraints to enforce valid argumentation structure. As features it uses structural, lexical, syntactic and context features, cf. Stab and Gurevych (2017) and Persing and Ng (2016).

Summarizing, we distinguish our framings in terms of *modularity* and in terms of their *constraints*. *Modularity*: Our dependency parsing framing and LSTM-ER are more modular than $\texttt{STag}_T$ because they de-couple relation information from entity information. However, (part of)

[3] By 'de-coupling', we mean that both tasks are treated separately rather than merging entity and relation information in the same tag label (output space). Still, a joint model like that of Miwa and Bansal (2016) de-couples the two tasks in such a way that many model parameters are shared across the tasks, similarly as in MTL.

this modularity can be regained by using $\texttt{STag}_T$ in an MTL setting. Moreover, since entity and relation information are considerably different, such a de-coupling may be advantageous. *Constraints*: LSTM-ER can, in principle, model any kind of—even many-to-many—relationships between detected entities. Thus, it is not guaranteed to produce trees, as we observe in AM datasets. $\texttt{STag}_T$ also does not need to produce trees, but it more severely restricts search space than does LSTM-ER: each token/component can only relate to one (and not several) other tokens/components. The same constraint is enforced by the dependency parsing framing. All of the tagging modelings, including LSTM-ER, are *local* models whereas our parsing framing is a *global* model: it globally enforces a tree structure on the token-level.

Further remarks: (1) part of the TreeLSTM modeling inherent to LSTM-ER is ineffective for our data because this modeling exploits dependency tree structures on the *sentence* level, while relationships between components are almost never on the sentence level. In our data, roughly 92% of all relationships are between components that appear in different sentences. Secondly, (2) that a model *enforces* a constraint does not necessarily mean that it is more suitable for a respective task. It has frequently been observed that models tend to produce output consistent with constraints in their training data in such situations (Zhang et al., 2017; Héctor and Plank, 2017); thus, they have *learned* the constraints.

5 Experiments

This section presents and discusses the empirical results for the AM framings outlined in §4. We relegate issues of *pre-trained word embeddings*, *hyperparameter optimization* and further *practical issues* to the supplementary material. Links to software used as well as some additional error analysis can also be found there.

Evaluation Metric We adopt the evaluation metric suggested in Persing and Ng (2016). This computes true positives TP, false positives FP, and false negatives FN, and from these calculates component and relation F_1 scores as $F_1 = \frac{2\text{TP}}{2\text{TP}+\text{FP}+\text{FN}}$. For space reasons, we refer to Persing and Ng (2016) for specifics, but to illustrate, for *components*, true positives are defined as the set of components in the gold standard for which there exists a predicted component with the same type that 'matches'. Persing and Ng (2016) define a notion of what we may term 'level α matching': for example, at the 100% level (exact match) predicted and gold components must have exactly the same spans, whereas at the 50% level they must only share at least 50% of their tokens (approximate match). We refer to these scores as C-F1 (100%) and C-F1 (50%), respectively. For *relations*, an analogous F1 score is determined, which we denote by R-F1 (100%) and R-F1 (50%). We note that R-F1 scores depend on C-F1 scores because correct relations must have correct arguments. We also define a 'global' F1 score, which is the F1-score of C-F1 and R-F1.

Most of our results are shown in Table 2.

(a) Dependency Parsing We show results for the two feature-based parsers MST (McDonald et al., 2005), Mate (Bohnet and Nivre, 2012) as well as the neural parsers by Dyer et al. (2015) (LSTM-Parser) and Kiperwasser and Goldberg (2016) (Kiperwasser). We train and test all parsers on the paragraph level, because training them on essay level was typically too memory-exhaustive.

MST mostly labels only non-argumentative units correctly, except for recognizing individual major claims, but never finds their exact spans (e.g., *"tourism can create negative impacts on"* while the gold major claim is *"international tourism can create negative impacts on the destination countries"*). Mate is slightly better and in particular recognizes several major claims correctly. Kiperwasser performs decently on the approximate match level, but not on exact level. Upon inspection, we find that the parser often predicts 'too large' component spans, e.g., by including following punctuation. The best parser by far is the LSTM-Parser. It is over 100% better than Kiperwasser on exact spans and still several percentage points on approximate spans.

How does performance change when we switch to the essay level? For the LSTM-Parser, the best performance on essay level is 32.84%/47.44% C-F1 (100%/50% level), and 9.11%/14.45% on R-F1, but performance result varied drastically between different parametrizations. Thus, the performance drop between paragraph and essay level is in any case immense.

Since the employed features of modern feature-based parsers are rather general—such as distance between words or word identities—we had expected them to perform much better. The mini-

	Paragraph level							Essay level						
	Acc.	C-F1		R-F1		F1		Acc.	C-F1		R-F1		F1	
		100%	50%	100%	50%	100%	50%		100%	50%	100%	50%	100%	50%
MST-Parser	31.23	0	6.90	0	1.29	0	2.17							
Mate	22.71	2.72	12.34	2.03	4.59	2.32	6.69							
Kiperwasser	52.80	26.65	61.57	15.57	34.25	19.65	44.01							
LSTM-Parser	55.68	58.86	68.20	35.63	40.87	44.38	51.11							
$\texttt{STag}_{\texttt{BLCC}}$	59.34	66.69	74.08	39.83	44.02	49.87	55.22	**60.46**	63.23	69.49	**34.82**	**39.68**	**44.90**	**50.51**
LSTM-ER	**61.67**	**70.83**	**77.19**	**45.52**	**50.05**	**55.42**	**60.72**	54.17	**66.21**	**73.02**	29.56	32.72	40.87	45.19
ILP	60.32	62.61	73.35	34.74	44.29	44.68	55.23							

Table 2: Performance of dependency parsers, $\texttt{STag}_{\texttt{BLCC}}$, LSTM-ER and ILP (from top to bottom). The ILP model operates on both levels. Best scores in each column in bold (signific. at $p < 0.01$; Two-sided Wilcoxon signed rank test, pairing F1 scores for documents). We also report token level accuracy.

mal feature set employed by Kiperwasser is apparently not sufficient for accurate AM but still a lot more powerful than the hand-crafted feature approaches. We hypothesize that the LSTM-Parser's good performance, relative to the other parsers, is due to its encoding of the *whole* stack history—rather than just the top elements on the stack as in Kiperwasser— which makes it aware of much larger 'contexts'. While the drop in performance from paragraph to essay level is expected, the LSTM-Parser's deterioration is much more severe than the other models' surveyed below. We believe that this is due to a mixture of the following: (1) 'capacity', i.e., model complexity, of the parsers—that is, risk of overfitting; and (2) few, but very long sequences on essay level—that is, little training data (trees), paired with a huge search space on each train/test instance, namely, the number of possible trees on n tokens. See also our discussions below, particularly, our stability analysis.

(b) Sequence Tagging For these experiments, we use the BLCC tagger from Ma and Hovy (2016) and refer to the resulting system as $\texttt{STag}_{\texttt{BLCC}}$. Again, we observe that paragraph level is considerably easier than essay level; e.g., for relations, there is ∼5% points increase from essay to paragraph level. Overall, $\texttt{STag}_{\texttt{BLCC}}$ is ∼13% better than the best parser for C-F1 and ∼11% better for R-F1 on the paragraph level. Our explanation is that taggers are simpler local models, and thus need less training data and are less prone to overfitting. Moreover, they can much better deal with the long sequences because they are largely invariant to length: e.g., it does in principle not matter, from a parameter estimation perspective, whether we train our taggers on two sequences of lengths n and m, respectively, or on one long sequence of length $n + m$.

(c) MTL As indicated, we use the MTL tagging framework from Søgaard and Goldberg (2016) for multi-task experiments. The underlying tagging framework is weaker than that of BLCC: there is no CNN which can take subword information into account and there are no dependencies between output labels: each tagging prediction is made independently of the other predictions. We refer to this system as $\texttt{STag}_{\texttt{BL}}$.

Accordingly, as Table 3 shows for the essay level (paragraph level omitted for space reasons), results are generally weaker: For exact match, C-F1 values are about ∼10% points below those of $\texttt{STag}_{\texttt{BLCC}}$, while approximate match performances are much closer. Hence, the independence assumptions of the BL tagger apparently lead to more 'local' errors such as exact argument span identification (cf. error analysis). An analogous trend holds for argument relations.

Additional Tasks: We find that when we train $\texttt{STag}_{\texttt{BL}}$ with only its main task—with label set $\mathcal{Y}$ as in Eq. (1)—the overall result is worst. In contrast, when we include the 'natural subtasks' "C" (label set $\mathcal{Y}_C$ consists of the projection on the coordinates (b, t) in $\mathcal{Y}$) and/or "R" (label set $\mathcal{Y}_R$ consists of the projection on the coordinates (d, s)), performance increases typically by a few percentage points. This indicates that complex sequence tagging may benefit when we train a "sub-labeler" in the same neural architecture, a finding that may be particularly relevant for morphological POS tagging (Müller et al., 2013). Unlike Søgaard and Goldberg (2016), we do not find that the optimal architecture is the one in which "lower" tasks (such as C or R) feed from lower layers. In fact, in one of the best parametrizations

the C task and the full task feed from the same layer in the deep BiLSTM. Moreover, we find that the C task is consistently more helpful as an auxiliary task than the R task.

	C-F1		R-F1		F1	
	100%	50%	100%	50%	100%	50%
$\mathcal{Y}$-3	49.59	65.37	26.28	37.00	34.35	47.25
$\mathcal{Y}$-3:$\mathcal{Y}_C$-1	54.71	66.84	28.44	37.35	37.40	47.92
$\mathcal{Y}$-3:$\mathcal{Y}_R$-1	51.32	66.49	26.92	37.18	35.31	47.69
$\mathcal{Y}$-3:$\mathcal{Y}_C$-3	**54.58**	67.66	**30.22**	**40.30**	**38.90**	**50.51**
$\mathcal{Y}$-3:$\mathcal{Y}_R$-3	53.31	66.71	26.65	35.86	35.53	46.64
$\mathcal{Y}$-3:$\mathcal{Y}_C$-1:$\mathcal{Y}_R$-2	52.95	**67.84**	27.90	39.71	36.54	50.09
$\mathcal{Y}$-3:$\mathcal{Y}_C$-3:$\mathcal{Y}_R$-3	54.55	67.60	28.30	38.26	37.26	48.86

Table 3: Performance of MTL sequence tagging approaches, essay level. Tasks separated by ":". Layers from which tasks feed are indicated by respective numbers.

On essay level, **(d) LSTM-ER** performs very well on component identification (+5% C-F1 compared to $\texttt{STag}_{\texttt{BLCC}}$), but rather poor on relation identification (-18% R-F1). Hence, its overall F1 on essay level is considerably below that of $\texttt{STag}_{\texttt{BLCC}}$. In contrast, LSTM-ER trained and tested on paragraph level substantially outperforms all other systems discussed, both for component as well as for relation identification.

We think that its generally excellent performance on components is due to LSTM-ER's de-coupling of component and relation tasks. Our findings indicate that a similar result can be achieved for $\texttt{STag}_T$ via MTL when components and relations are included as auxiliary tasks, cf. Table 3. For example, the improvement of LSTM-ER over $\texttt{STag}_{\texttt{BLCC}}$, for C-F1, roughly matches the increase for $\texttt{STag}_{\texttt{BL}}$ when including components and relations separately ($\mathcal{Y}$-3:$\mathcal{Y}_C$-3:$\mathcal{Y}_R$-3) over not including them as auxiliary tasks ($\mathcal{Y}$-3). Lastly, the better performance of LSTM-ER over $\texttt{STag}_{\texttt{BLCC}}$ for relations on paragraph level appears to be a consequence of its better performance on components. E.g., when both arguments are correctly predicted, $\texttt{STag}_{\texttt{BLCC}}$ has even higher chance of getting their relation correct than LSTM-ER (95.34% vs. 94.17%).

Why does LSTM-ER degrade so much on essay level for R-F1? As said, text sequences are much longer on essay level than on paragraph level—hence, there are on average many more entities on essay level. Thus, there are also many more possible relations between all entities discovered in a text—namely, there are $O(2^{m^2})$ possible relations between m discovered components. Due to its generality, LSTM-ER considers all these relations as plausible, while $\texttt{STag}_T$ does not (for any of choice of T): e.g., our coding explicitly constrains each premise to link to exactly *one* other component, rather than to $0, \ldots, m$ possible components, as LSTM-ER allows. In addition, our explicit coding of distance values d biases the learner T to reflect the distribution of distance values found in real essays—namely, that related components are typically close in terms of the number of components between them. In contrast, LSTM-ER only mildly prefers short-range dependencies over long-range dependencies, cf. Figure 4.

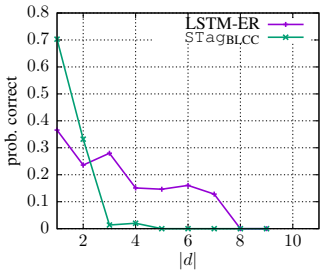


Figure 4: Probability of correct relation identification given true distance is $|d|$.

The **(e) ILP** has access to both paragraph and essay level information and thus has always more information than all neural systems compared to. Thus, it also knows in which paragraph in an essay it is. This is useful particularly for major claims, which always occur in first or last paragraphs in our data. Still, its performance is equal to or lower than that of LSTM-ER and $\texttt{STag}_{\texttt{BLCC}}$ when both are evaluated on paragraph level.

Stability Analysis

Table 4 shows averages and standard deviations of two selected models, namely, the $\texttt{STag}_{\texttt{BLCC}}$ tagging framework as well as the LSTM-Parser over several different runs (different random initializations as well as different hyperparameters as discussed in the supplementary material). These results detail that the taggers have lower standard deviations than the parsers. The difference is particularly striking on the essay level where the parsers often completely fail to learn, that is, their performance scores are close to 0%. As discussed above, we attribute this to the parsers' increased model capacity relative to the taggers, which makes them more prone to overfitting. Data scarcity is another very likely source of error in this context, as the parsers only observe 322 (though very rich) trees

in the training data, while the taggers are always roughly trained on 120K tokens. On paragraph level, they do observe more trees, namely, 1786.

	$\texttt{STag}_{\text{BLCC}}$	LSTM-Parser
Essay	60.62±3.54	9.40±13.57
Paragraph	64.74±1.97	56.24±2.87

Table 4: C-F1 (100%) in % for the two indicated systems; essay vs. paragraph level. Note that the mean performances are lower than the majority performances over the runs given in Table 2.

Error analysis

A systematic source of errors for all systems is detecting exact argument spans (segmentation). For instance, the ILP system predicts the following premise: *"As a practical epitome , students should be prepared to present in society after their graduation"*, while the gold premise omits the preceding discourse marker, and hence reads: *"students should be prepared to present in society after their graduation"*. On the one hand, it has been observed that even humans have problems exactly identifying such entity boundaries (Persing and Ng, 2016; Yang and Cardie, 2013). On the other hand, our results in Table 2 indicate that the neural taggers BLCC and BLC (in the LSTM-ER model) are much better at such exact identification than either the ILP model or the neural parsers. While the parsers' problems are most likely due to model complexity, we hypothesize that the ILP model's increased error rates stem from a weaker underlying tagging model (feature-based CRF vs. BiLSTM) and/or its features.[4] In fact, as Table 5 shows, the macro-F1 scores[5] on *only* the component segmentation tasks (BIO labeling) are substantially higher for both LSTM-ER and $\texttt{STag}_{\text{BLCC}}$ than for the ILP model. Noteworthy, the two neural systems even outperform the human upper bound (HUB) in this context, reported as 88.6% in Stab and Gurevych (2017).

6 Conclusion

We present the first study on neural end-to-end AM. We experimented with different *framings*, such as encoding AM as a dependency parsing problem, as a sequence tagging problem with particular label set, as a multi-task sequence tagging problem, and as a problem with both sequential and tree structure information. We show that (1) neural computational AM is as good or (substantially) better than a competing feature-based ILP formulation, while eliminating the need for manual feature engineering and costly ILP constraint designing. (2) BiLSTM taggers perform very well for component identification, as demonstrated for our $\texttt{STag}_T$ frameworks, for T = BLCC and T = BL, as well as for LSTM-ER (BLC tagger). (3) (Naively) coupling component and relation identification is not optimal, but both tasks should be treated separately, but modeled jointly. (4) Relation identification is more difficult: when there are few entities in a text ("short documents"), a more general framework such as that provided in LSTM-ER performs reasonably well. When there are many entities ("long documents"), a more restrained modeling is preferable. These are also our *policy recommendations*. Our work yields new state-of-the-art results in end-to-end AM on the PE dataset from Stab and Gurevych (2017).

Another possible framing, not considered here, is to frame AM as an encoder-decoder problem (Bahdanau et al., 2015; Vinyals et al., 2015). This is an even more general modeling than LSTM-ER. Its suitability for the end-to-end learning task is scope for future work, but its adequacy for component classification and relation identification has been investigated in Potash et al. (2016).

	$\texttt{STag}_{\text{BLCC}}$	LSTM-ER	ILP	HUB
Essay	90.04	90.57		
Paragraph	88.32	90.84	86.67	88.60

Table 5: F1 scores in % on BIO tagging task.

Acknowledgments

We thank Lucie Flekova, Judith Eckle-Kohler, Nils Reimers, and Christian Stab for valuable feedback and discussions. We also thank the anonymous reviewers for their suggestions. The second author was supported by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 01UG1416B (CEDIFOR).

[4]The BIO tagging task is independent and thus not affected by the ILP constraints in the model of Stab and Gurevych (2017). The same holds true for the model of Persing and Ng (2016).

[5]Denoted $Fscore_M$ in Sokolova and Lapalme (2009).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 1171–1179.

Or Biran and Owen Rambow. 2011. Identifying justifications in written dialogs. In *Fifth IEEE International Conference on Semantic Computing (ICSC)*. pages 162–168.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP-CoNLL '12, pages 1455–1465.

Rich Caruana. 1997. Multitask learning. *Mach. Learn.* 28(1):41–75. https://doi.org/10.1023/A:1007379606734.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '08, pages 160–167. https://doi.org/10.1145/1390156.1390177.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 334–343.

Eirini Florou, Stasinos Konstantopoulos, Antonis Koukourikos, and Pythagoras Karampiperis. 2013. Argument extraction for supporting public policy formulation. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*. Association for Computational Linguistics, Sofia, Bulgaria, pages 49–54.

Ivan Habernal and Iryna Gurevych. 2015. Exploiting debate portals for semi-supervised argumentation mining in user-generated web discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 2127–2137.

Ivan Habernal and Iryna Gurevych. 2016. Argumentation Mining in User-Generated Web Discourse. *Computational Linguistics* 43(1). In press. Preprint: http://arxiv.org/abs/1601.02403.

Martnez Alonso Héctor and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of EACL 2017 (long paper)*. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR* abs/1508.01991. http://arxiv.org/abs/1508.01991.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Christian Kirschner, Judith Eckle-Kohler, and Iryna Gurevych. 2015. Linking the thoughts: Analysis of argumentation structures in scientific publications. In *Proceedings of the 2nd Workshop on Argumentation Mining held in conjunction with the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL HLT 2015)*. pages 1–11.

Kevin Knight, Daniel Marcu, and Jill Burstein. 2003. Finding the write stuff: Automatic identification of discourse structure in student essays. *IEEE Intelligent Systems* 18:32–39.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*. pages 1489–1500.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1064–1074. http://www.aclweb.org/anthology/P16-1101.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology*

and Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '05, pages 523–530. https://doi.org/10.3115/1220575.1220641.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1105–1116. http://www.aclweb.org/anthology/P16-1105.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic detection of arguments in legal texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*. ACM, New York, NY, USA, ICAIL '07, pages 225–230. https://doi.org/10.1145/1276318.1276362.

Philippe Muller, Stergos D. Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*. pages 1883–1900.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 322–332. http://www.aclweb.org/anthology/D13-1032.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*. ACM, New York, NY, USA, ICAIL '09, pages 98–107. https://doi.org/10.1145/1568234.1568246.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 938–948. http://aclweb.org/anthology/D15-1110.

Andreas Peldszus and Manfred Stede. 2016. An annotated corpus of argumentative microtexts. In *Argumentation and Reasoned Action: Proceedings of the 1st European Conference on Argumentation*. Lisabon, pages 801–815.

Nanyun Peng and Mark Dredze. 2016. Multi-task multi-domain representation learning for sequence tagging. *CoRR* abs/1608.02689. http://arxiv.org/abs/1608.02689.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 543–552.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1384–1394. http://www.aclweb.org/anthology/N16-1164.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2016. Here's my point: Argumentation Mining with Pointer Networks. *Arxiv preprint https://arxiv.org/abs/1612.08994* .

Chris Reed, Raquel Mochales-Palau, Glenn Rowe, and Marie-Francine Moens. 2008. Language resources for studying argument. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*. Marrakech, Morocco, LREC '08, pages 2613–2618.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence - an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 440–450.

N. Rooney, H. Wang, and F. Browne. 2012. Applying kernel methods to argumentation mining. In *Twenty-Fifth International FLAIRS Conference*.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671* .

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 231–235. http://anthology.aclweb.org/P16-2038.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management* 45(4):427–437. https://doi.org/10.1016/j.ipm.2009.03.002.

Swapna Somasundaran, Brian Riordan, Binod Gyawali, and Su-Youn Yoon. 2016. Evaluating argumentative and narrative essays using graphs.

In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 1568–1578.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics* (in press), preprint: http://arxiv.org/abs/1604.07370).

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 2692–2700.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1640–1649. http://www.aclweb.org/anthology/P13-1161.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *CoRR* abs/1603.06270.

Fan Zhang and Diane J. Litman. 2016. Using context to predict the purpose of argumentative writing revisions. In *The Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1424–1430.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of EACL 2017 (long papers)*. Association for Computational Linguistics.

Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision

Chen Liang[*], **Jonathan Berant**[†], **Quoc Le, Kenneth D. Forbus, Ni Lao**
Northwestern University, Evanston, IL
Tel-Aviv University, Tel Aviv-Yafo, Israel
Google Inc., Mountain View, CA
{chenliang2013,forbus}@u.northwestern.edu, joberant@cs.tau.ac.il, {qvl,nlao}@google.com

Abstract

Harnessing the statistical power of neural networks to perform language understanding and symbolic reasoning is difficult, when it requires executing efficient discrete operations against a large knowledge-base. In this work, we introduce a *Neural Symbolic Machine* (NSM), which contains (a) a neural "programmer", i.e., a sequence-to-sequence model that maps language utterances to programs and utilizes a *key-variable memory* to handle compositionality (b) a symbolic "computer", i.e., a Lisp interpreter that performs program execution, and helps find good programs by pruning the search space. We apply REINFORCE to directly optimize the task reward of this structured prediction problem. To train with weak supervision and improve the stability of REINFORCE we augment it with an *iterative maximum-likelihood* training process. NSM outperforms the state-of-the-art on the WEBQUESTIONSSP dataset when trained from question-answer pairs only, without requiring any feature engineering or domain-specific knowledge.

1 Introduction

Deep neural networks have achieved impressive performance in supervised classification and structured prediction tasks such as speech recognition (Hinton et al., 2012), machine translation (Bahdanau et al., 2014; Wu et al., 2016) and more. However, training neural networks for semantic parsing (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2011) or program induction, where language is mapped to a symbolic representation that is executed by an executor, through weak supervision remains challenging. This is because the model must interact with a symbolic executor through non-differentiable operations to search over a large program space.

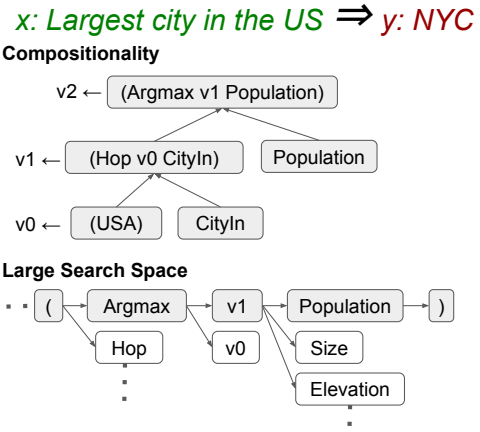


Figure 1: The main challenges of training a semantic parser from weak supervision: (a) *compositionality*: we use variables (v_0, v_1, v_2) to store execution results of intermediate generated programs. (b) *search*: we prune the search space and augment REINFORCE with pseudo-gold programs.

In semantic parsing, recent work handled this (Dong and Lapata, 2016; Jia and Liang, 2016) by training from manually annotated programs and avoiding program execution at training time. However, annotating programs is known to be expensive and scales poorly. In program induction, attempts to address this problem (Graves et al., 2014; Reed and de Freitas, 2016; Kaiser and Sutskever, 2015; Graves et al., 2016b; Andreas et al., 2016) either utilized low-level memory (Zaremba and Sutskever, 2015), or required memory to be differentiable (Neelakantan et al., 2015; Yin et al., 2015) so that the model can be trained with backpropagation. This makes it difficult to use the efficient discrete operations and memory of a traditional computer, and limited the application to synthetic or small knowledge bases.

In this paper, we propose to utilize the memory and discrete operations of a traditional com-

[*] Work done while the author was interning at Google
[†] Work done while the author was visiting Google

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 23–33
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1003

puter in a novel Manager-Programmer-Computer (MPC) framework for neural program induction, which integrates three components:

1. A **"manager"** that provides weak supervision (e.g., 'NYC' in Figure 1) through a reward indicating how well a task is accomplished. Unlike full supervision, weak supervision is easy to obtain at scale (Section 3.1).
2. A **"programmer"** that takes natural language as input and generates a program that is a sequence of tokens (Figure 2). The programmer learns from the reward and must overcome the hard search problem of finding correct programs (Section 2.2).
3. A **"computer"** that executes programs in a high level programming language. Its non-differentiable memory enables *abstract*, *scalable* and *precise* operations, but makes training more challenging (Section 2.3). To help the "programmer" prune the search space, it provides a friendly *neural computer interface*, which detects and eliminates invalid choices (Section 2.1).

Within this framework, we introduce the Neural Symbolic Machine (NSM) and apply it to semantic parsing. NSM contains a neural sequence-to-sequence (seq2seq) "programmer" (Sutskever et al., 2014) and a symbolic non-differentiable Lisp interpreter ("computer") that executes programs against a large knowledge-base (KB).

Our technical contribution in this work is threefold. First, to support language *compositionality*, we augment the standard seq2seq model with a *key-variable memory* to save and reuse intermediate execution results (Figure 1). This is a novel application of pointer networks (Vinyals et al., 2015) to compositional semantics.

Second, to alleviate the search problem of finding correct programs when training from question-answer pairs,we use the computer to execute partial programs and prune the programmer's search space by checking the syntax and semantics of generated programs. This generalizes the weakly supervised semantic parsing framework (Liang et al., 2011; Berant et al., 2013) by leveraging semantic denotations during structural search.

Third, to train from weak supervision and directly maximize the expected reward we turn to the REINFORCE (Williams, 1992) algorithm. Since learning from scratch is difficult for REINFORCE, we combine it with an *iterative maximum likelihood* (ML) training process, where beam search is used to find pseudo-gold programs, which are then used to augment the objective of REINFORCE.

On the WEBQUESTIONSSP dataset (Yih et al., 2016), NSM achieves new state-of-the-art results with weak supervision, significantly closing the gap between weak and full supervision for this task. Unlike prior works, it is trained end-to-end, and does not require feature engineering or domain-specific knowledge.

2 Neural Symbolic Machines

We now introduce NSM by first describing the "computer", a non-differentiable Lisp interpreter that executes programs against a large KB and provides code assistance (Section 2.1). We then propose a seq2seq model ("programmer") that supports compositionality using a key-variable memory to save and reuse intermediate results (Section 2.2). Finally, we describe a training procedure that is based on REINFORCE, but is augmented with pseudo-gold programs found by an iterative ML training procedure (Section 2.3).

Before diving into details, we define the *semantic parsing* task: given a knowledge base $\mathbb{K}$, and a question $x = (w_1, w_2, ..., w_m)$, produce a program or logical form z that when executed against $\mathbb{K}$ generates the right answer y. Let $\mathcal{E}$ denote a set of entities (e.g., ABELINCOLN),[1] and let $\mathcal{P}$ denote a set of properties (e.g., PLACEOFBIRTH). A knowledge base $\mathbb{K}$ is a set of assertions or triples $(e_1, p, e_2) \in \mathcal{E} \times \mathcal{P} \times \mathcal{E}$, such as (ABELINCOLN, PLACEOFBIRTH, HODGENVILLE).

2.1 Computer: Lisp Interpreter with Code Assistance

Semantic parsing typically requires using a set of operations to query the knowledge base and process the results. Operations learned with neural networks such as addition and sorting do not perfectly generalize to inputs that are larger than the ones observed in the training data (Graves et al., 2014; Reed and de Freitas, 2016). In contrast, operations implemented in high level programming languages are *abstract*, *scalable*, and *precise*, thus generalizes perfectly to inputs of arbitrary size. Based on this observation, we implement operations necessary for semantic parsing with an or-

[1] We also consider numbers (e.g., "1.33") and date-times (e.g., "1999-1-1") as entities.

dinary programming language instead of trying to learn them with a neural network.

We adopt a Lisp interpreter as the "computer". A program C is a list of expressions $(c_1...c_N)$, where each expression is either a special token "*Return*" indicating the end of the program, or a list of tokens enclosed by parentheses "$(FA_1...A_K)$". F is a function, which takes as input K arguments of specific types. Table 1 defines the semantics of each function and the types of its arguments (either a property p or a variable r). When a function is executed, it returns an entity list that is the expression's denotation in $\mathbb{K}$, and save it to a new variable.

By introducing variables that save the intermediate results of execution, the program naturally models *language compositionality* and describes from left to right a bottom-up derivation of the full meaning of the natural language input, which is convenient in a seq2seq model (Figure 1). This is reminiscent of the floating parser (Wang et al., 2015; Pasupat and Liang, 2015), where a derivation tree that is not grounded in the input is incrementally constructed.

The set of programs defined by our functions is equivalent to the subset of λ-calculus presented in (Yih et al., 2015). We did not use full Lisp programming language here, because constructs like control flow and loops are unnecessary for most current semantic parsing tasks, and it is simple to add more functions to the model when necessary.

To create a friendly *neural computer interface*, the interpreter provides code assistance to the programmer by producing a list of valid tokens at each step. First, a valid token should not cause a syntax error: e.g., if the previous token is "(", the next token must be a function name, and if the previous token is "*Hop*", the next token must be a variable. More importantly, a valid token should not cause a semantic (run-time) error: this is detected using the denotation saved in the variables. For example, if the previously generated tokens were "(*Hop* r", the next available token is restricted to properties $\{p \mid \exists e, e' : e \in r, (e, p, e') \in \mathbb{K}\}$ that are reachable from entities in r in the KB. These checks are enabled by the variables and can be derived from the definition of the functions in Table 1. The interpreter prunes the "programmer"'s search space by orders of magnitude, and enables learning from weak supervision on a large KB.

2.2 Programmer: Seq2seq Model with Key-Variable Memory

Given the "computer", the "programmer" needs to map natural language into a program, which is a sequence of tokens that reference operations and values in the "computer". We base our programmer on a standard seq2seq model with attention, but extend it with a key-variable memory that allows the model to learn to represent and refer to program variables (Figure 2).

Sequence-to-sequence models consist of two RNNs, an encoder and a decoder. We used a 1-layer GRU (Cho et al., 2014) for both the encoder and decoder. Given a sequence of words $w_1, w_2...w_m$, each word w_t is mapped to an embedding q_t (embedding details are in Section 3). Then, the encoder reads these embeddings and updates its hidden state step by step using $h_{t+1} = GRU(h_t, q_t, \theta_{Encoder})$, where $\theta_{Encoder}$ are the GRU parameters. The decoder updates its hidden states u_t by $u_{t+1} = GRU(u_t, c_{t-1}, \theta_{Decoder})$, where c_{t-1} is the embedding of last step's output token a_{t-1}, and $\theta_{Decoder}$ are the GRU parameters. The last hidden state of the encoder h_T is used as the decoder's initial state. We also adopt a dot-product attention similar to Dong and Lapata (2016). The tokens of the program $a_1, a_2...a_n$ are generated one by one using a softmax over the vocabulary of valid tokens at each step, as provided by the "computer" (Section 2.1).

To achieve compositionality, the decoder must learn to represent and refer to intermediate variables whose value was saved in the "computer" after execution. Therefore, we augment the model with a **key-variable memory**, where each entry has two components: a continuous embedding key v_i, and a corresponding variable token R_i referencing the value in the "computer" (see Figure 2). During encoding, we use an entity linker to link text spans (e.g., *"US"*) to KB entities. For each linked entity we add a memory entry where the key is the average of GRU hidden states over the entity span, and the variable token (R_1) is the name of a variable in the computer holding the linked entity (*m.USA*) as its value. During decoding, when a full expression is generated (i.e., the decoder generates ")"), it gets executed, and the result is stored as the value of a new variable in the "computer". This variable is keyed by the GRU hidden state at that step. When a new variable R_1 with key embedding v_1 is added into the key-variable memory,

$(\ \textit{Hop}\ r\ p\ ) \Rightarrow \{e_2 \mid e_1 \in r, (e_1, p, e_2) \in \mathbb{K}\}$
$(\ \textit{ArgMax}\ r\ p\ ) \Rightarrow \{e_1 \mid e_1 \in r, \exists e_2 \in \mathcal{E} : (e_1, p, e_2) \in \mathbb{K}, \forall e : (e_1, p, e) \in \mathbb{K}, e_2 \geq e\}$
$(\ \textit{ArgMin}\ r\ p\ ) \Rightarrow \{e_1 \mid e_1 \in r, \exists e_2 \in \mathcal{E} : (e_1, p, e_2) \in \mathbb{K}, \forall e : (e_1, p, e) \in \mathbb{K}, e_2 \leq e\}$
$(\ \textit{Filter}\ r_1\ r_2\ p\ ) \Rightarrow \{e_1 \mid e_1 \in r_1, \exists e_2 \in r_2 : (e_1, p, e_2) \in \mathbb{K}\}$

Table 1: Interpreter functions. r represents a variable, p a property in Freebase. $\geq$ and $\leq$ are defined on numbers and dates.

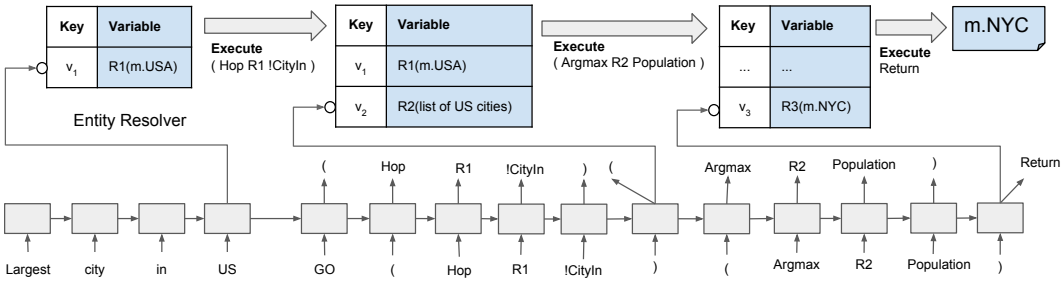


Figure 2: Semantic Parsing with NSM. The key embeddings of the key-variable memory are the output of the sequence model at certain encoding or decoding steps. For illustration purposes, we also show the values of the variables in parentheses, but the sequence model never sees these values, and only references them with the name of the variable ("R_1"). A special token "*GO*" indicates the start of decoding, and "*Return*" indicates the end of decoding.

the token R_1 is added into the decoder vocabulary with v_1 as its embedding. The final answer returned by the "programmer" is the value of the last computed variable.

Similar to pointer networks (Vinyals et al., 2015), the key embeddings for variables are dynamically generated for each example. During training, the model learns to represent variables by backpropagating gradients from a time step where a variable is selected by the decoder, through the key-variable memory, to an earlier time step when the key embedding was computed. Thus, the encoder/decoder learns to generate representations for variables such that they can be used at the right time to construct the correct program.

While the key embeddings are differentiable, the values referenced by the variables (lists of entities), stored in the "computer", are symbolic and non-differentiable. This distinguishes the key-variable memory from other memory-augmented neural networks that use continuous differentiable embeddings as the values of memory entries (Weston et al., 2014; Graves et al., 2016a).

2.3 Training NSM with Weak Supervision

NSM executes non-differentiable operations against a KB, and thus end-to-end backpropagation is not possible. Therefore, we base our training procedure on REINFORCE (Williams, 1992; Norouzi et al., 2016). When the reward signal is sparse and the search space is large, it is common to utilize some full supervision to pre-train REINFORCE (Silver et al., 2016). To train from weak supervision, we suggest an iterative ML procedure for finding pseudo-gold programs that will bootstrap REINFORCE.

REINFORCE We can formulate training as a reinforcement learning problem: given a question x, the state, action and reward at each time step $t \in \{0, 1, ..., T\}$ are (s_t, a_t, r_t). Since the environment is deterministic, the state is defined by the question x and the action sequence: $s_t = (x, a_{0:t-1})$, where $a_{0:t-1} = (a_0, ..., a_{t-1})$ is the history of actions at time t. A valid action at time t is $a_t \in A(s_t)$, where $A(s_t)$ is the set of valid tokens given by the "computer". Since each action corresponds to a token, the full history $a_{0:T}$ corresponds to a program. The reward $r_t = I[t = T] \cdot F_1(x, a_{0:T})$ is non-zero only at the last step of decoding, and is the F_1 score computed comparing the gold answer and the answer generated by executing the program $a_{0:T}$. Thus, the cumulative reward of a program $a_{0:T}$ is

$$R(x, a_{0:T}) = \sum_t r_t = F_1(x, a_{0:T}).$$

The agent's decision making procedure at each time is defined by a policy, $\pi_\theta(s, a) = P_\theta(a_t = a | x, a_{0:t-1})$, where θ are the model parameters. Since the environment is deterministic, the probability of generating a program $a_{0:T}$ is

$$P_\theta(a_{0:T}|x) = \prod_t P_\theta(a_t \mid x, a_{0:t-1}).$$

We can define our objective to be the expected cumulative reward and use policy gradient meth-

ods such as REINFORCE for training. The objective and gradient are:

$$J^{RL}(\theta) = \sum_x \mathbb{E}_{P_\theta(a_{0:T}|x)}[R(x, a_{0:T})],$$
$$\nabla_\theta J^{RL}(\theta) = \sum_x \sum_{a_{0:T}} P_\theta(a_{0:T} \mid x) \cdot [R(x, a_{0:T}) - B(x)] \cdot \nabla_\theta \log P_\theta(a_{0:T} \mid x),$$

where $B(x) = \sum_{a_{0:T}} P_\theta(a_{0:T} \mid x) R(x, a_{0:T})$ is a baseline that reduces the variance of the gradient estimation without introducing bias. Having a separate network to predict the baseline is an interesting future direction.

While REINFORCE assumes a stochastic policy, we use beam search for gradient estimation. Thus, in contrast with common practice of approximating the gradient by sampling from the model, we use the top-k action sequences (programs) in the beam with normalized probabilities. This allows training to focus on sequences with high probability, which are on the decision boundaries, and reduces the variance of the gradient.

Empirically (and in line with prior work), REINFORCE converged slowly and often got stuck in local optima (see Section 3). The difficulty of training resulted from the sparse reward signal in the large search space, which caused model probabilities for programs with non-zero reward to be very small at the beginning. If the beam size k is small, good programs fall off the beam, leading to zero gradients for all programs in the beam. If the beam size k is large, training is very slow, and the normalized probabilities of good programs when the model is untrained are still very small, leading to (1) near zero baselines, thus near zero gradients on "bad" programs (2) near zero gradients on good programs due to the low probability $P_\theta(a_{0:T} \mid x)$. To combat this, we present an alternative training strategy based on maximum-likelihood.

Iterative ML If we had gold programs, we could directly optimize their likelihood. Since we do not have gold programs, we can perform an iterative procedure (similar to hard Expectation-Maximization (EM)), where we search for good programs given fixed parameters, and then optimize the probability of the best program found so far. We do decoding on an example with a large beam size and declare $a^{best}_{0:T}(x)$ to be the pseudo-gold program, which achieved highest reward with shortest length among the programs decoded on x in all previous iterations. Then, we can optimize the ML objective:

$$J^{ML}(\theta) = \sum_x \log P_\theta(a^{best}_{0:T}(x) \mid x) \quad (1)$$

A question x is not included if we did not find any program with positive reward.

Training with iterative ML is fast because there is at most one program per example and the gradient is not weighted by model probability. while decoding with a large beam size is slow, we could train for multiple epochs after each decoding. This iterative process has a bootstrapping effect that a better model leads to a better program $a^{best}_{0:T}(x)$ through decoding, and a better program $a^{best}_{0:T}(x)$ leads to a better model through training.

Even with a large beam size, some programs are hard to find because of the large search space. A common solution to this problem is to use curriculum learning (Zaremba and Sutskever, 2015; Reed and de Freitas, 2016). The size of the search space is controlled by both the set of functions used in the program and the program length. We apply curriculum learning by gradually increasing both these quantities (see details in Section 3) when performing iterative ML.

Nevertheless, iterative ML uses only pseudo-gold programs and does not directly optimize the objective we truly care about. This has two adverse effects: (1) The best program $a^{best}_{0:T}(x)$ could be a spurious program that accidentally produces the correct answer (e.g., using the property PLACEOFBIRTH instead of PLACEOFDEATH when the two places are the same), and thus does not generalize to other questions. (2) Because training does not observe full negative programs, the model often fails to distinguish between tokens that are related to one another. For example, differentiating PARENTSOF vs. SIBLINGSOF vs. CHILDRENOF can be challenging. We now present learning where we combine iterative ML with REINFORCE.

Augmented REINFORCE To bootstrap REINFORCE, we can use iterative ML to find pseudo-gold programs, and then add these programs to the beam with a reasonably large probability. This is similar to methods from imitation learning (Ross et al., 2011; Jiang et al., 2012) that define a proposal distribution by linearly interpolating the model distribution and an oracle.

Algorithm 1 IML-REINFORCE

Input: question-answer pairs $\mathbb{D} = \{(x_i, y_i)\}$, mix ratio α, reward function $R(\cdot)$, training iterations N_{ML}, N_{RL}, and beam sizes B_{ML}, B_{RL}.
Procedure:
Initialize $C_x^* = \emptyset$ the best program so far for x
Initialize model θ randomly ▷ Iterative ML
for $n = 1$ to N_{ML} **do**
  for (x, y) in D **do**
    $\mathbb{C} \leftarrow$ Decode B_{ML} programs given x
    for j in $1...|\mathbb{C}|$ **do**
      if $R_{x,y}(C_j) > R_{x,y}(C_x^*)$ **then** $C_x^* \leftarrow C_j$
  $\theta \leftarrow$ ML training with $\mathbb{D}_{ML} = \{(x, C_x^*)\}$
Initialize model θ randomly ▷ REINFORCE
for $n = 1$ to N_{RL} **do**
  $\mathbb{D}_{RL} \leftarrow \emptyset$ is the RL training set
  for (x, y) in D **do**
    $\mathbb{C} \leftarrow$ Decode B_{RL} programs from x
    for j in $1...|\mathbb{C}|$ **do**
      if $R_{x,y}(C_j) > R_{x,y}(C_x^*)$ **then** $C_x^* \leftarrow C_j$
    $\mathbb{C} \leftarrow \mathbb{C} \cup \{C_x^*\}$
    for j in $1...|\mathbb{C}|$ **do**
      $\hat{p}_j \leftarrow (1-\alpha) \cdot \frac{p_j}{\sum_{j'} p_{j'}}$ where $p_j = P_\theta(C_j \mid x)$
      if $C_j = C_x^*$ **then** $\hat{p}_j \leftarrow \hat{p}_j + \alpha$
      $\mathbb{D}_{RL} \leftarrow \mathbb{D}_{RL} \cup \{(x, C_j, \hat{p}_j)\}$
  $\theta \leftarrow$ REINFORCE training with $\mathbb{D}_{RL}$

Algorithm 1 describes our overall training procedure. We first run iterative ML for N_{ML} iterations and record the best program found for every example x_i. Then, we run REINFORCE, where we normalize the probabilities of the programs in beam to sum to $(1-\alpha)$ and add α to the probability of the best found program $C^*(x_i)$. Consequently, the model always puts a reasonable amount of probability on a program with high reward during training. Note that we randomly initialized the parameters for REINFORCE, since initializing from the final ML parameters seems to get stuck in a local optimum and produced worse results.

On top of imitation learning, our approach is related to the common practice in reinforcement learning (Schaul et al., 2016) to replay rare successful experiences to reduce the training variance and improve training efficiency. This is also similar to recent developments (Wu et al., 2016) in machine translation, where ML and RL objectives are linearly combined, because anchoring the model to some high-reward outputs stabilizes training.

3 Experiments and Analysis

We now empirically show that NSM can learn a semantic parser from weak supervision over a large KB. We evaluate on WEBQUESTIONSSP, a challenging semantic parsing dataset with strong baselines. Experiments show that NSM achieves new state-of-the-art performance on WEBQUESTIONSSP with weak supervision, and significantly closes the gap between weak and full supervisions for this task.

3.1 The WEBQUESTIONSSP dataset

The WEBQUESTIONSSP dataset (Yih et al., 2016) contains full semantic parses for a subset of the questions from WEBQUESTIONS (Berant et al., 2013), because 18.5% of the original dataset were found to be "not answerable". It consists of 3,098 question-answer pairs for training and 1,639 for testing, which were collected using Google Suggest API, and the answers were originally obtained using Amazon Mechanical Turk workers. They were updated in (Yih et al., 2016) by annotators who were familiar with the design of Freebase and added semantic parses. We further separated out 620 questions from the training set as a validation set. For query pre-processing we used an in-house named entity linking system to find the entities in a question. The quality of the entity linker is similar to that of (Yih et al., 2015) at 94% of the gold root entities being included. Similar to Dong and Lapata (2016), we replaced named entity tokens with a special token "*ENT*". For example, the question "*who plays meg in family guy*" is changed to "*who plays ENT in ENT ENT*". This helps reduce overfitting, because instead of memorizing the correct program for a specific entity, the model has to focus on other context words in the sentence, which improves generalization.

Following (Yih et al., 2015) we used the last publicly available snapshot of Freebase (Bollacker et al., 2008). Since NSM training requires random access to Freebase during decoding, we pre-processed Freebase by removing predicates that are not related to world knowledge (starting with "*/common/*", "*/type/*", "*/freebase/*"),[2] and removing all text valued predicates, which are rarely the answer. Out of all 27K relations, 434 relations are removed during preprocessing. This results in a graph that fits in memory with 23K relations, 82M nodes, and 417M edges.

3.2 Model Details

For pre-trained word embeddings, we used the 300 dimension GloVe word embeddings trained on 840B tokens (Pennington et al., 2014). On the encoder side, we added a projection matrix to

[2] We kept "*/common/topic/notable_types*".

transform the embeddings into 50 dimensions. On the decoder side, we used the same GloVe embeddings to construct an embedding for each property using its Freebase id, and also added a projection matrix to transform this embedding to 50 dimensions. A Freebase id contains three parts: domain, type, and property. For example, the Freebase id for PARENTSOF is *"/people/person/parents"*. *"people"* is the domain, *"person"* is the type and *"parents"* is the property. The embedding is constructed by concatenating the average of word embeddings in the domain and type name to the average of word embeddings in the property name. For example, if the embedding dimension is 300, the embedding dimension for *"/people/person/parents"* will be 600. The first 300 dimensions will be the average of the embeddings for *"people"* and *"person"*, and the second 300 dimensions will be the embedding for *"parents"*.

The dimension of encoder hidden state, decoder hidden state and key embeddings are all 50. The embeddings for the functions and special tokens (e.g., "*UNK*", "*GO*") are randomly initialized by a truncated normal distribution with mean=0.0 and stddev=0.1. All the weight matrices are initialized with a uniform distribution in $[-\frac{\sqrt{3}}{d}, \frac{\sqrt{3}}{d}]$ where d is the input dimension. Dropout rate is set to 0.5, and we see a clear tendency for larger dropout rate to produce better performance, indicating overfitting is a major problem for learning.

3.3 Training Details

In iterative ML training, the decoder uses a beam of size $k = 100$ to update the pseudo-gold programs and the model is trained for 20 epochs after each decoding step. We use the Adam optimizer (Kingma and Ba, 2014) with initial learning rate 0.001. In our experiment, this process usually converges after a few (5-8) iterations.

For REINFORCE training, the best hyperparameters are chosen using the validation set. We use a beam of size $k = 5$ for decoding, and α is set to 0.1. Because the dataset is small and some relations are only used once in the whole training set, we train the model on the entire training set for 200 iterations with the best hyperparameters. Then we train the model with learning rate decay until convergence. Learning rate is decayed as $g_t = g_0 \times \beta^{\frac{\max(0, t-t_s)}{m}}$, where $g_0 = 0.001$, $\beta = 0.5$ $m = 1000$, and t_s is the number of training steps at the end of iteration 200.

Since decoding needs to query the knowledge base (KB) constantly, the speed bottleneck for training is decoding. We address this problem in our implementation by partitioning the dataset, and using multiple decoders in parallel to handle each partition. We use 100 decoders, which queries 50 KG servers, and one trainer. The neural network model is implemented in TensorFlow. Since the model is small, we didn't see a significant speedup by using GPU, so all the decoders and the trainer are using CPU only.

Inspired by the staged generation process in Yih et al. (2015), curriculum learning includes two steps. We first run iterative ML for 10 iterations with programs constrained to only use the "*Hop*" function and the maximum number of expressions is 2. Then, we run iterative ML again, but use both "*Hop*" and "*Filter*". The maximum number of expressions is 3, and the relations used by "*Hop*" are restricted to those that appeared in $a^{best}_{0:T}(q)$ in the first step.

3.4 Results and discussion

We evaluate performance using the offical evaluation script for WEBQUESTIONSSP. Because the answer to a question may contain multiple entities or values, precision, recall and F1 are computed based on the output of each individual question, and average F1 is reported as the main evaluation metric. Accuracy measures the proportion of questions that are answered exactly.

A comparison to STAGG, the previous state-of-the-art model (Yih et al., 2016, 2015), is shown in Table 2. Our model beats STAGG with weak supervision by a significant margin on all metrics, while relying on no feature engineering or handcrafted rules. When STAGG is trained with strong supervision it obtains an F1 of 71.7, and thus NSM closes half the gap between training with weak and full supervision.

Model	Prec.	Rec.	F1	Acc.
STAGG	67.3	73.1	66.8	58.8
NSM	70.8	76.0	**69.0**	59.5

Table 2: Results on the test set. Average F1 is the main evaluation metric and NSM outperforms STAGG with no domain-specific knowledge or feature engineering.

Four key ingredients lead to the final performance of NSM. The first one is the neural computer interface that provides code assistance by checking for syntax and semantic errors. We find

that semantic checks are very effective for open-domain KBs with a large number of properties. For our task, the average number of choices is reduced from 23K per step (all properties) to less than 100 (the average number of properties connected to an entity).

The second ingredient is augmented REINFORCE training. Table 3 compares augmented REINFORCE, REINFORCE, and iterative ML on the validation set. REINFORCE gets stuck in local optimum and performs poorly. Iterative ML training is not directly optimizing the F1 measure, and achieves sub-optimal results. In contrast, augmented REINFORCE is able to bootstrap using pseudo-gold programs found by iterative ML and achieves the best performance on both the training and validation set.

Settings	Train F1	Valid F1
Iterative ML	68.6	60.1
REINFORCE	55.1	47.8
Augmented REINFORCE	83.0	**67.2**

Table 3: Average F1 on the validation set for augmented REINFORCE, REINFORCE, and iterative ML.

The third ingredient is curriculum learning during iterative ML. We compare the performance of the best programs found with and without curriculum learning in Table 4. We find that the best programs found with curriculum learning are substantially better than those found without curriculum learning by a large margin on every metric.

Settings	Prec.	Rec.	F1	Acc.
No curriculum	79.1	91.1	78.5	67.2
Curriculum	88.6	96.1	89.5	79.8

Table 4: Evaluation of the programs with the highest F1 score in the beam ($a_{0:t}^{best}$) with and without curriculum learning.

The last important ingredient is reducing overfitting. Given the small size of the dataset, overfitting is a major problem for training neural network models. We show the contributions of different techniques for controlling overfitting in Table 5. Note that after all the techniques have been applied, the model is still overfitting with training F1@1=83.0% and validation F1@1=67.2%.

Among the programs generated by the model, a significant portion (36.7%) uses more than one expression. From Table 6, we can see that the performance doesn't decrease much as the compositional depth increases, indicating that the model is effective at capturing compositionality. We observe that programs with three expressions use a more limited set of properties, mainly focusing on answering a few types of questions such as "who plays meg in family guy", "what college did jeff corwin go to" and "which countries does russia border". In contrast, programs with two expressions use a more diverse set of properties, which could explain the lower performance compared to programs with three expressions.

Settings	Δ F1@1
−Pretrained word embeddings	−5.5
−Pretrained property embeddings	−2.7
−Dropout on GRU input and output	−2.4
−Dropout on softmax	−1.1
−Anonymize entity tokens	−2.0

Table 5: Contributions of different overfitting techniques on the validation set.

#Expressions	0	1	2	3
Percentage	0.4%	62.9%	29.8%	6.9%
F1	0.0	73.5	59.9	70.3

Table 6: Percentage and performance of model generated programs with different complexity (number of expressions).

Error analysis Error analysis on the validation set shows two main sources of errors:

1. **Search failure**: Programs with high reward are not found during search for pseudo-gold programs, either because the beam size is not large enough, or because the set of functions implemented by the interpreter is insufficient. The 89.5% F1 score in Table 4 indicates that at least 10% of the questions are of this kind.
2. **Ranking failure**: Programs with high reward exist in the beam, but are not ranked at the top during decoding. Because the training error is low, this is largely due to overfitting or spurious programs. The 67.2% F1 score in Table 3 indicates that about 20% of the questions are of this kind.

4 Related work

Among deep learning models for program induction, Reinforcement Learning Neural Turing Machines (RL-NTMs) (Zaremba and Sutskever, 2015) are the most similar to NSM, as a non-differentiable machine is controlled by a sequence

model. Therefore, both models rely on REINFORCE for training. The main difference between the two is the abstraction level of the programming language. RL-NTM uses lower level operations such as memory address manipulation and byte reading/writing, while NSM uses a high level programming language over a large knowledge base that includes operations such as following properties from entities, or sorting based on a property, which is more suitable for representing semantics. Earlier works such as OOPS (Schmidhuber, 2004) has desirable characteristics, for example, the ability to define new functions. These remain to be future improvements for NSM.

We formulate NSM training as an instance of reinforcement learning (Sutton and Barto, 1998) in order to directly optimize the task reward of the structured prediction problem (Norouzi et al., 2016; Li et al., 2016; Yu et al., 2017). Compared to imitation learning methods (Daume et al., 2009; Ross et al., 2011) that interpolate a model distribution with an oracle, NSM needs to solve a challenging search problem of training from weak supervisions in a large search space. Our solution employs two techniques (a) a symbolic "computer" helps find good programs by pruning the search space (b) an iterative ML training process, where beam search is used to find pseudo-gold programs. Wiseman and Rush (Wiseman and Rush, 2016) proposed a max-margin approach to train a sequence-to-sequence scorer. However, their training procedure is more involved, and we did not implement it in this work. MIXER (Ranzato et al., 2015) also proposed to combine ML training and REINFORCE, but they only considered tasks with full supervisions. Berant and Liang (Berant and Liang, 2015) applied imitation learning to semantic parsing, but still requires hand crafted grammars and features.

NSM is similar to Neural Programmer (Neelakantan et al., 2015) and Dynamic Neural Module Network (Andreas et al., 2016) in that they all solve the problem of semantic parsing from structured data, and generate programs using similar semantics. The main difference between these approaches is how an intermediate result (the memory) is represented. Neural Programmer and Dynamic-NMN chose to represent results as vectors of weights (row selectors and attention vectors), which enables backpropagation and search through all possible programs in parallel. However, their strategy is not applicable to a large KB such as Freebase, which contains about 100M entities, and more than 20k properties. Instead, NSM chooses a more scalable approach, where the "computer" saves intermediate results, and the neural network only refers to them with variable names (e.g., "R_1" for all cities in the US).

NSM is similar to the Path Ranking Algorithm (PRA) (Lao et al., 2011) in that semantics is encoded as a sequence of actions, and denotations are used to prune the search space during learning. NSM is more powerful than PRA by 1) allowing more complex semantics to be composed through the use of a key-variable memory; 2) controlling the search procedure with a trained neural network, while PRA only samples actions uniformly; 3) allowing input questions to express complex relations, and then dynamically generating action sequences. PRA can combine multiple semantic representations to produce the final prediction, which remains to be future work for NSM.

5 Conclusion

We propose the Manager-Programmer-Computer framework for neural program induction. It integrates neural networks with a symbolic *non-differentiable* computer to support *abstract*, *scalable* and *precise* operations through a friendly *neural computer interface*. Within this framework, we introduce the Neural Symbolic Machine, which integrates a neural sequence-to-sequence "programmer" with key-variable memory, and a symbolic Lisp interpreter with code assistance. Because the interpreter is non-differentiable and to directly optimize the task reward, we apply REINFORCE and use pseudo-gold programs found by an iterative ML training process to bootstrap training. NSM achieves new state-of-the-art results on a challenging semantic parsing dataset with weak supervision, and significantly closes the gap between weak and full supervision. It is trained end-to-end, and does not require any feature engineering or domain-specific knowledge.

Acknowledgements

We thank for discussions and help from Arvind Neelakantan, Mohammad Norouzi, Tom Kwiatkowski, Eugene Brevdo, Lukasz Kaizer, Thomas Strohmann, Yonghui Wu, Zhifeng Chen, Alexandre Lacoste, and John Blitzer. The second author is partially supported by the Israel Science Foundation, grant 942/16.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *CoRR* abs/1601.01705.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*. volume 2, page 6.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *TACL* 3:545–558.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *International Conference on Management of Data (SIGMOD)*. pages 1247–1250.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

H. Daume, J. Langford, and D. Marcu. 2009. Search-based structured prediction. *Machine Learning* 75:297–325.

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

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* .

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio G. Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, AdriÃ P. Badia, Karl M. Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. 2016a. Hybrid computing using a neural network with dynamic external memory. *Nature* advance online publication. https://doi.org/10.1038/nature20101.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016b. Hybrid computing using a neural network with dynamic external memory. *Nature* .

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29(6):82–97.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

J. Jiang, A. Teichert, J. Eisner, and H. Daume. 2012. Learned prioritization for trading off accuracy and speed. In *Advances in Neural Information Processing Systems (NIPS)*.

Łukasz Kaiser and Ilya Sutskever. 2015. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228* .

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 529–539.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* .

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

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2015. Neural programmer: Inducing latent programs with gradient descent. *CoRR* abs/1511.04834.

Mohammad Norouzi, Samy Bengio, zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances in Neural Information Processing Systems (NIPS)*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

Scott Reed and Nando de Freitas. 2016. Neural programmer-interpreters. In *ICLR*.

S. Ross, G. Gordon, and A. Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Artificial Intelligence and Statistics (AISTATS)*.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay. In *International Conference on Learning Representations*. Puerto Rico.

Jürgen Schmidhuber. 2004. Optimal ordered problem solver. *Machine Learning* 54(3):211–254.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916* .

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*. pages 229–256.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *CoRR* abs/1606.02960. http://arxiv.org/abs/1606.02960.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Association for Computational Linguistics (ACL)*.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Association for Computational Linguistics (ACL)*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables. *arXiv preprint arXiv:1512.00965* .

Adam Yu, Hongrae Lee, and Quoc Le. 2017. Learning to skim text. In *ACL*.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines. *arXiv preprint arXiv:1505.00521* .

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

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

Neural Relation Extraction with Multi-lingual Attention

Yankai Lin[1], **Zhiyuan Liu**[1]*, **Maosong Sun**[1,2]
[1] Department of Computer Science and Technology,
State Key Lab on Intelligent Technology and Systems,
National Lab for Information Science and Technology, Tsinghua University, Beijing, China
[2] Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China

Abstract

Relation extraction has been widely used for finding unknown relational facts from the plain text. Most existing methods focus on exploiting mono-lingual data for relation extraction, ignoring massive information from the texts in various languages. To address this issue, we introduce a multi-lingual neural relation extraction framework, which employs mono-lingual attention to utilize the information within mono-lingual texts and further proposes cross-lingual attention to consider the information consistency and complementarity among cross-lingual texts. Experimental results on real-world datasets show that our model can take advantage of multi-lingual texts and consistently achieve significant improvements on relation extraction as compared with baselines. The source code of this paper can be obtained from https://github.com/thunlp/MNRE

1 Introduction

People build many large-scale knowledge bases (KBs) to store structured knowledge about the real world, such as Wikidata[1] and DBpedia[2]. KBs are playing an important role in many AI and NLP applications such as information retrieval and question answering. The facts in KBs are typically organized in the form of triplets, e.g., (*New York*, `CityOf`, *United States*). Since existing KBs are far from complete and new facts are growing infinitely, meanwhile manual annotation of these knowledge is time-consuming and human-intensive, many works have been devoted to automated extraction of novel facts from various Web resources, where relation extraction (RE) from plain texts is one the most important knowledge sources.

Among various methods for relation extraction, distant supervision is the most promising approach (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012), which can automatically generate training instances via aligning KBs and texts to address the issue of lacking supervised data. As the development of deep learning, Zeng et al. (2015) introduce neural networks to extract relations with automatically learned features from training instances. To address the wrong labelling issue of distant supervision data, Lin et al. (2016) further employ sentence-level attention mechanism in neural relation extraction, and achieves the state-of-the-art performance.

However, most RE systems concentrate on extracting relational facts from mono-lingual data. In fact, people describe knowledge about the world using various languages. And people speaking different languages also share similar knowledge about the world due to the similarities of human experiences and human cognitive systems. For instance, though *New York* and *United States* are expressed as 纽约 and 美国 respectively in Chinese, both Americans and Chinese share the fact that "*New York* is a city of *USA*."

It is straightforward to build mono-lingual RE systems separately for each single language. But if so, it won't be able to take full advantage of diverse information hidden in the data of various languages. Multi-lingual data will benefit relation extraction for the following two reasons: **1. Consistency**. According to the distant supervision data in our experiments[3], we find that over half of Chinese

* Corresponding author: Zhiyuan Liu (liuzy@tsinghua.edu.cn).

[1] http://www.wikidata.org/

[2] http://wiki.dbpedia.org/

[3] The data is generated by aligning Wikidata with Chinese

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 34–43
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1004

Relation	City in
English	1. **New York is a city** in the northeastern **United States**.
Chinese	1. **纽约**位于美国纽约州东南部大西洋沿岸，**是美国第一大城市**及第一大港. (**New York** is in the United States New York and on the Atlantic coast of the southeast Atlantic, **is the largest city** and largest port **in the United States**.) 2. **纽约是美国**人口最多**的城市**. (**New York is** the most populous **city in the United States**)

Table 1: An example of Chinese sentences and English sentence about the same relational fact (*New York*, `CityOf`, *United States*). Important parts are highlighted with bold face.

and English sentences are longer than 20 words, in which only several words are related to the relational facts. Take Table 1 for example. The first Chinese sentence has over 20 words, in which only "纽约" (New York) and "是美国第一大城市" (is the biggest city in the United States) actually directly reflect the relational fact `CityOf`. It is thus non-trivial to locate and learn these relational patterns from complicated sentences for relation extraction. Fortunately, a relational fact is usually expressed with certain patterns in various languages, and the correspondence of these patterns among languages is substantially consistent. The pattern consistency among languages provides us augmented clues to enhance relational pattern learning for relation extraction.

2. Complementarity. From our experiment data, we also find that 42.2% relational facts in English data and 41.6% ones in Chinese data are unique. Moreover, for nearly half of relations, the number of sentences expressing relational facts of these relations varies a lot in different languages. It is thus straightforward that the texts in different languages can be complementary to each other, especially from those resource-rich languages to resource-poor languages, and improve the overall performance of relation extraction.

To take full consideration of these issues, we propose **M**ulti-lingual Attention-based **N**eural **R**elation **E**xtraction (MNRE). We first employ a convolutional neural network (CNN) to embed the relational patterns in sentences into real-valued vectors. Afterwards, to consider the complementarity of all informative sentences in various languages and capture the consistency of relational patterns, we apply mono-lingual attention to select the informative sentences within each language and propose cross-lingual attention to take advantages of pattern consistency and complementarity among languages. Finally, we classify relations according to the global vector aggregated from all sentence vectors weighted by mono-lingual attention and cross-lingual attention.

In experiments, we build training instances via distant supervision by aligning Wikidata with Chinese Baidu Baike and English Wikipedia articles, and evaluate the performance of relation extraction in both English and Chinese. The experimental results show that our framework achieves significant improvement for relation extraction as compared to all baseline methods including both mono-lingual and multi-lingual ones. It indicates that our framework can take full advantages of sentences in different languages and better capture sophisticated patterns expressing relations.

2 Related Work

Recent years KBs have been widely used on various AI and NLP applications. As an important approach to enrich KBs, relation extraction from plain text has attracted many research interests. Relation extraction typically classifies each entity pair into various relation types according to supporting sentences that the both entities appear, which needs human-labelled relation-specific training instances. Many works have been invested to relation extraction including kernel-based model (Zelenko et al., 2003), embedding-based model (Gormley et al., 2015), CNN-based models (Zeng et al., 2014; dos Santos et al., 2015), and RNN-based model (Socher et al., 2012).

Nevertheless, these RE systems are insufficient due to the lack of training data. To address this issue, Mintz et al. (2009) align plain text with Freebase to automatically generate training instances following the distant supervision assumption. To further alleviate the wrong labelling problem, Riedel et al. (2010) model distant supervision for relation extraction as a multi-instance single-label learning problem, and Hoffmann et al. (2011); Surdeanu et al. (2012) regard it as a multi-instance multi-label learning problem. Recently, Zeng et al. (2015) attempt to connect neural networks with distant supervision following the expressed-at-least-once assumption. Lin

Baidu Baike and English Wikipedia articles, which will be introduced in details in the section of experiments.

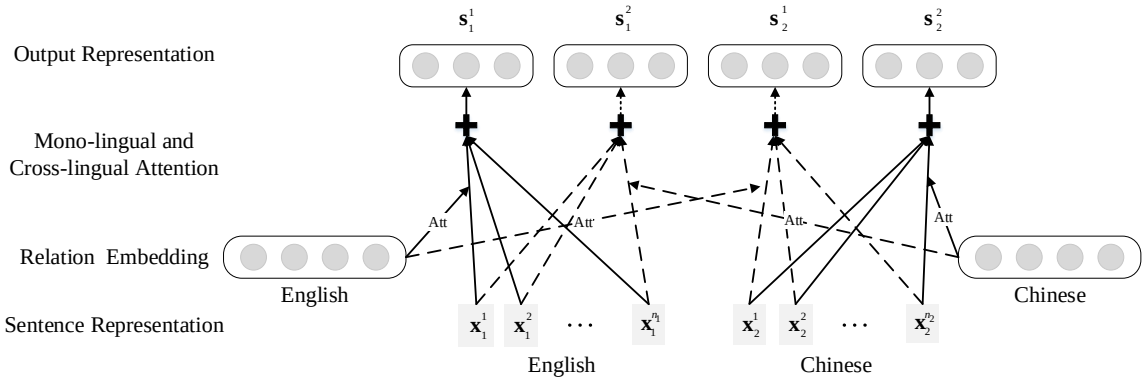


Figure 1: Overall architecture of our multi-lingual attention which contains two languages including English and Chinese. The solid lines indicates mono-lingual attention and the dashed lines indicates cross-lingual attention.

et al. (2016) further utilize sentence-level attention mechanism to consider all informative sentences jointly.

Most existing RE systems are absorbed in extracting relations from mono-lingual data, ignoring massive information lying in texts from multiple languages. In this area, Faruqui and Kumar (2015) present a language independent open domain relation extraction system, and Verga et al. (2015) further employ Universal Schema to combine OpenIE and link-prediction perspective for multi-lingual relation extraction. Both the works focus on multi-lingual transfer learning and learn a predictive model on a new language for existing KBs, by leveraging unified representation learning for cross-lingual entities. Different from these works, our framework aims to jointly model the texts in multiple languages to enhance relation extraction with distant supervision. To the best of our knowledge, this is the first effort to multi-lingual neural relation extraction.

The scope of multi-lingual analysis has been widely considered in many tasks besides relation extraction, such as sentiment analysis (Boiy and Moens, 2009), cross-lingual document summarization (Boudin et al., 2011), information retrieval in Web search (Dong et al., 2014) and so on.

3 Methodology

In this section, we describe our proposed MNRE framework in detail. The key motivation of MNRE is that, for each relational fact, the relation patterns in sentences of different languages should be substantially consistent, and MNRE can utilize the pattern consistency and complementarity among languages to achieve better results for relation extraction.

Formally, given two entities, their corresponding sentences in m different languages are defined as $T = \{S_1, S_2, \ldots, S_m\}$, where $S_j = \{x_j^1, x_j^2, \ldots, x_j^{n_j}\}$ corresponds to the sentence set in the jth language with n_j sentences. Our model measures a score $f(T, r)$ for each relation r, which is expected to be high when r is the valid one, otherwise low. The MNRE framework contains two main components:

1. Sentence Encoder. Given a sentence x and two target entities, we employ CNN to encode relation patterns in x into a distributed representation $\mathbf{x}$. The sentence encoder can also be implemented with GRU (Cho et al., 2014) or LSTM (Hochreiter and Schmidhuber, 1997). In experiments, we find CNN can achieve a better trade-off between computational efficiency and performance effectiveness. Thus, in this paper, we focus on CNN as the sentence encoder.

2. Multi-lingual Attention. With all sentences in various languages encoded into distributed vector representations, we apply mono-lingual and cross-lingual attentions to capture those informative sentences with accurate relation patterns. MNRE further aggregates these sentence vectors with weighted attentions into global representations for relation prediction.

We introduce the two components in detail as follows.

3.1 Sentence Encoder

The sentence encoder aims to transform a sentence x into its distributed representation $\mathbf{x}$ via CNN. First, it embeds the words in the input sentence

into dense real-valued vectors. Next, it employs convolutional, max-pooling and non-linear transformation layers to construct the distributed representation of the sentence, i.e., $\mathbf{x}$.

3.1.1 Input Representation

Following (Zeng et al., 2014), we transform each input word into the concatenation of two kinds of representations: (1) a word embedding which captures syntactic and semantic meanings of the word, and (2) a position embedding which specifies the position information of this word with respect to two target entities. In this way, we can represent the input sentence as a vector sequence $\mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots\}$ with $\mathbf{w}_i \in \mathbb{R}^d$, where $d = d^a + d^b \times 2$. (d^a and d^b are the dimensions of word embeddings and position embeddings respectively)

3.1.2 Convolution, Max-pooling and Non-linear Layers

After encoding the input sentence, we use a convolutional layer to extract the local features, max-pooling, and non-linear layers to merge all local features into a global representation.

First, the convolutional layer extracts local features by sliding a window of length l over the sentence and perform a convolution within each sliding window. Formally, the output of convolutional layer for the ith sliding window is computed as:

$$\mathbf{p}_i = \mathbf{W}\mathbf{w}_{i-l+1:i} + \mathbf{b}, \tag{1}$$

where $\mathbf{w}_{i-l+1:i}$ indicates the concatenation of l word embeddings within the i-th window, $\mathbf{W} \in \mathbb{R}^{d^c \times (l \times d)}$ is the convolution matrix and $\mathbf{b} \in \mathbb{R}^{d^c}$ is the bias vector. (d^c is the dimension of output embeddings of the convolution layer)

After that, we combines all local features via a max-pooling operation and apply a hyperbolic tangent function to obtain a fixed-sized sentence vector for the input sentence. Formally, the ith element of the output vector $\mathbf{x} \in \mathbb{R}^{d^c}$ is calculated as:

$$[\mathbf{x}]_j = \tanh\big(\max_i(\mathbf{p}_{ij})\big). \tag{2}$$

The final vector $\mathbf{x}$ is expected to efficiently encode relation patterns about target entities from the input sentence.

Here, instead of max pooling operation, we can use piecewise max pooling operation adopted by PCNN (Zeng et al., 2015) which is a variation of CNN to better capture the relation patterns in the input sentence.

3.2 Multi-lingual Attention

To exploit the information of the sentences from all languages, our model adopts two kinds of attention mechanisms for multi-lingual relation extraction, including: (1) the mono-lingual attention which selects the informative sentences within one language and (2) the cross-lingual attention which measures the pattern consistency among languages.

3.2.1 Mono-lingual Attention

To address the wrong-labelling issue in distant supervision, we follow the idea of sentence-level attention (Lin et al., 2016) and set mono-lingual attention for MNRE. It is intuitive that each human language has its own characteristics. Hence we adopt different mono-lingual attentions to de-emphasize those noisy sentences within each language.

More specifically, for the j-th language and the sentence set S_j, we aim to aggregate all sentence vectors into a real-valued vector $\mathbf{S}_j$ for relation prediction. The mono-lingual vector $\mathbf{S}_j$ is computed as a weighted sum of those sentence vectors $\mathbf{x}_j^i$:

$$\mathbf{S}_j = \sum_i \alpha_j^i \mathbf{x}_j^i, \tag{3}$$

where α_j^i is the attention score of each sentence vector $\mathbf{x}_j^i$, defined as:

$$\alpha_j^i = \frac{\exp(e_j^i)}{\sum_k \exp(e_j^k)}, \tag{4}$$

where e_j^i is referred as a query-based function which scores how well the input sentence x_j^i reflects its labelled relation r. There are many ways to obtain e_j^i, and here we simply compute e_i as the inner product:

$$e_j^i = \mathbf{x}_j^i \cdot \mathbf{r}_j. \tag{5}$$

Here $\mathbf{r}_j$ is the query vector of the relation r with respect to the j-th language.

3.2.2 Cross-lingual Attention

Besides mono-lingual attention, we propose cross-lingual attention for neural relation extraction to better take advantages of multi-lingual data.

The key idea of cross-lingual attention is to emphasize those sentences which have strong consistency among different languages. On the basis of mono-lingual attention, cross-lingual attention

is capable of further removing unlikely sentences and resulting in more concentrated and informative sentences, with the factor of consistent correspondence of relation patterns among different languages.

Cross-lingual attention works similar to mono-lingual attention. Suppose j indicates a language and k is a another language ($k \neq j$). Formally, the cross-lingual representation $\mathbf{S}_{jk}$ is defined as a weighted sum of those sentence vectors $\mathbf{x}_j^i$ in the jth language:

$$\mathbf{S}_{jk} = \sum_i \alpha_{jk}^i \mathbf{x}_j^i, \tag{6}$$

where α_{jk}^i is the cross-lingual attention score of each sentence vector $\mathbf{x}_j^i$ with respect to the kth language. The cross-lingual attention α_{jk}^i is defined as:

$$\alpha_{jk}^i = \frac{\exp(e_{jk}^i)}{\sum_k \exp(e_{jk}^k)}, \tag{7}$$

where e_{jk}^i is referred as a query-based function which scores the consistency between the input sentence x_j^i in the jth language and the relation patterns in the kth language for expressing the semantic meanings of the labelled relation r. Similar to the mono-lingual attention, we compute e_{jk}^i as follows:

$$e_{jk}^i = \mathbf{x}_j^i \cdot \mathbf{r}_k, \tag{8}$$

where $\mathbf{r}_k$ is the query vector of the relation r with respect to the kth language.

Note that, for convenience, we denote those mono-lingual attention vectors $\mathbf{S}_j$ as $\mathbf{S}_{jj}$ in the remainder of this paper.

3.3 Prediction

For each entity pair and its corresponding sentence set T in m languages, we can obtain $m \times m$ vectors $\{\mathbf{S}_{jk} | j, k \in \{1, \ldots, m\}\}$ from the neural networks with multi-lingual attention. Those vectors with $j = k$ are mono-lingual attention vectors, and those with $j \neq k$ are cross-lingual attention vectors.

We take all vectors $\{\mathbf{S}_{jk}\}$ together and define the overall score function $f(T, r)$ as follows:

$$f(T, r) = \sum_{j,k \in \{1,\ldots,m\}} \log p(r|\mathbf{S}_{jk}, \theta), \tag{9}$$

where $p(r|\mathbf{S}_{jk}, \theta)$ is the probability of predicting the relation r conditional on $\mathbf{S}_{jk}$, computed using a softmax layer as follows:

$$p(r|\mathbf{S}_{jk}, \theta) = \text{softmax}(\mathbf{M}\mathbf{S}_{jk} + \mathbf{d}), \tag{10}$$

where $\mathbf{d} \in \mathbb{R}^{n_r}$ is a bias vector, n_r is the number of relation types and $\mathbf{M} \in \mathbb{R}^{n_r \times R^c}$ is a global relation matrix initialized randomly.

To better consider the characteristics of each human language, we further introduce $\mathbf{R}_k$ as the specific relation matrix of the kth language. Here we simply define $\mathbf{R}_k$ as composed by $\mathbf{r}_k$ in Eq. (8). Hence, Eq. (10) can be extended to:

$$p(r|\mathbf{S}_{jk}, \theta) = \text{softmax}[(\mathbf{R}_k + \mathbf{M})\mathbf{S}_{jk} + \mathbf{d}], \tag{11}$$

where $\mathbf{M}$ encodes global patterns for predicting relations and $\mathbf{R}_k$ encodes those language-specific characteristics.

Note that, in the training phase, the vectors $\{\mathbf{S}_{jk}\}$ are constructed using Eq. (3) and (6) using the labelled relation. In the testing phase, since the relation is not known in advance, we will construct different vectors $\{\mathbf{S}_{jk}\}$ for each possible relation r to compute $f(T, r)$ for relation prediction.

3.4 Optimization

Here we introduce the learning and optimization details of our MNRE framework. We define the objective function as follows:

$$J(\theta) = \sum_{i=1}^{s} f(T_i, r_i), \tag{12}$$

where s indicates the number of all entity pairs with each corresponding to a sentence set in different languages, and θ indicates all parameters of our framework.

To solve the optimization problem, we adopt mini-batch stochastic gradient descent (SGD) to minimize the objective function. For learning, we iterate by randomly selecting a mini-batch from the training set until converge.

4 Experiments

We first introduce the datasets and evaluation metrics used in the experiments. Next, we use a validation set to determine the best model parameters and choose the best model via early stopping. Afterwards, we show the effectiveness of our framework of considering pattern complementarity and consistency for multi-lingual relation extraction by quantitative and qualitative analysis. Finally, we compare the effect of two kinds of relation matrices in Eq. (11) used for prediction.

4.1 Datasets and Evaluation Metrics

We generate a new multi-lingual relation extraction dataset to evaluate our MNRE framework. Without loss of generality, the experiments focus on relation extraction from two languages including English and Chinese. In this dataset, the Chinese instances are generated by aligning Chinese Baidu Baike with Wikidata, and the English instances are generated by aligning English Wikipedia articles with Wikidata. The relational facts of Wikidata in this dataset are divided into three parts for training, validation and testing respectively. There are 176 relations including a special relation NA indicating there is no relation between entities. And we set both validation and testing sets for Chinese and English parts contain the same facts. We list the statistics about the dataset in Table 2.

Dataset		#Rel	#Sent	#Fact
English	Train	176	1,022,239	47,638
	Valid		80,191	2,192
	Test		162,018	4,326
Chinese	Train	176	940,595	42,536
	Valid		82,699	2,192
	Test		167,224	4,326

Table 2: Statistics of the dataset.

We follow previous works (Mintz et al., 2009) and investigate the performance of RE systems using the held-out evaluation, by comparing the relational facts discovered by RE systems from the testing set with those facts in KB. The evaluation method assumes that if a RE system accurately finds more relational facts in KBs from the testing set, it will achieve better performance for relation extraction. The held-out evaluation provides an approximate measure of RE performance without time-consuming human evaluation. In experiments, we report the precision/recall curves as the evaluation metric.

4.2 Experimental Settings

We tune the parameters of our MNRE framework by grid searching using validation set. For training, we set the iteration number over all the training data as 15. The best models were selected by early stopping using the evaluation results on the validation set. In Table 3 we show the best setting of all parameters used in our experiments.

Hyper-parameter	value
Window size w	3
Sentence embedding size d^c	230
Word dimension d^a	50
Position dimension d^b	5
Batch size B	160
Learning rate λ	0.001
Dropout probability p	0.5

Table 3: Parameter settings.

4.3 Effectiveness of Consistency

To demonstrate the effectiveness of considering pattern consistency among languages, we empirically compare different methods through held-out evaluation. We select CNN proposed in (Zeng et al., 2014) as our sentence encoder and implement it by ourselves which achieves comparable results as the authors reported on their experimental dataset NYT10[4]. And we compare the performance of our framework with the [P]CNN model trained with only English data ([P]CNN-En), only Chinese data ([P]CNN-Zh), a joint model ([P]CNN+joint) which predicts using [P]CNN-En and [P]CNN-Zh jointly, and another joint model with shared embeddings ([P]CNN+share) which trains [P]CNN-En and [P]CNN-Zh with common relation embedding matrices.

From Fig. 2, we have the following observations:

(1) Both [P]CNN+joint and [P]CNN+share achieve better performances as compared to [P]CNN-En and [P]CNN-Zh. It indicates that utilizing Chinese and English sentences jointly is beneficial to extracting novel relational facts. The reason is that those relational facts that are discovered from multiple languages are more reliable to be true.

(2) CNN+share only has similar performance as compared to CNN+joint, even through a bit worse when recall ranges from 0.1 to 0.2. Besides, PCNN+share performs worse than PCNN+joint nearly over the entire range of recall. It demonstrates that a simple combination of multiple languages by sharing relation embedding matrices cannot further capture more implicit correlations among various languages.

(3) Our MNRE model achieves the highest precision over the entire range of recall as compared to other methods including [P]CNN+joint and [P]CNN+share models. By grid searching of

[4] http://iesl.cs.umass.edu/riedel/ecml/

CNN+Zh	CNN+En	MNRE	Sentence
—	Medium	Low	1. **Barzun** is a commune in the Pyrénées-Atlantiques department in the Nouvelle-Aquitaine region of south-western **France**.
—	Medium	High	2. **Barzun** was born in Créteil , France
Medium	—	Low	3. 作为从**法国**移民到美国来的顶尖知识分子，**巴尔赞**与莱昂内尔·特里林、德怀特·麦克唐纳等人一道，在冷战时期积极参与美国的公共知识生活 …(As a top intellectual immigrating from **France** to the United States, **Barzun**, together with Lionel Trilling and Dwight Macdonald, actively participated in public knowledge life in the United States during the cold war …)
Medium	—	High	4. **巴尔赞**于 1907 年出生于法国一个知识分子家庭，1920 年赴美。(**Barzun** was born in a **French** intellectual family in 1907 and went to America in 1920.)

Table 4: An example of our multi-lingual attention. Low, medium and high indicate the attention weights.

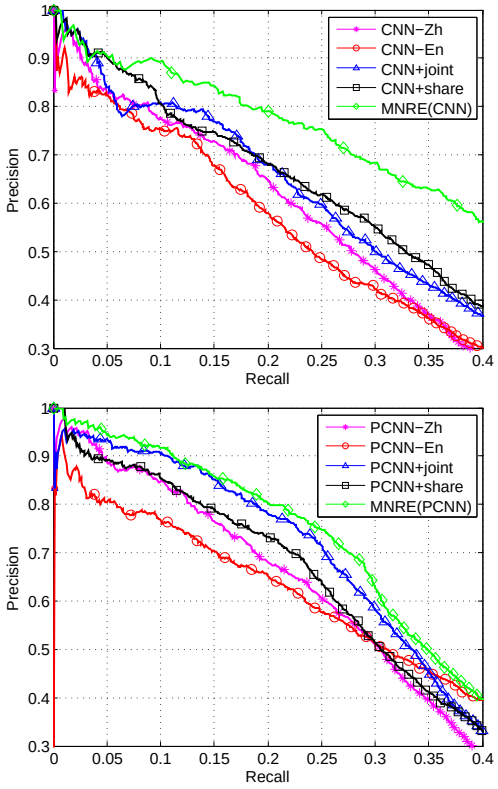


Figure 2: Top: Aggregated precision/recall curves of CNN-En, CNN-Zh, CNN+joint, CNN+share, and MNRE(CNN). Bottom: Aggregated precision/recall curves of PCNN-En, PCNN-Zh, PCNN+joint, PCNN+share, and MNRE(PCNN)

parameters for these baseline models, we can observe that both [P]CNN+joint and [P]CNN+share cannot achieve competitive results compared to MNRE even when increasing the size of the output layer. This indicates that no more useful information can be captured by simply increasing model size. On the contrary, our proposed MNRE model can successfully improve multi-lingual relation extraction by considering pattern consistency among languages.

We further give an example of cross-lingual attention in Table 4. It shows four sentences having the highest and lowest Chinese-to-English and English-to-Chinese attention weights respectively with respect to the relation `PlaceOfBirth` in MNRE. We highlight the entity pairs in bold face. For comparison, we also show their attention weights from CNN+Zh and CNN+En. From the table we find that, although all of the four sentences actually express the fact that Barzun was born in France, the first and third sentences contain much more noisy information that may confuse RE systems. By considering pattern consistency between sentences in two languages with cross-lingual attention, MNRE can identify the second and fourth sentences that unambiguously express the relation `PlaceOfBirth` with higher attention as compared to CNN+Zh and CNN+En.

4.4 Effectiveness of Complementarity

To demonstrate the effectiveness of considering pattern complementarity among languages, we empirically compare the following methods through held-out evaluation: MNRE for English (MNRE-En) and MNRE for Chinese (MNRE-Zh) which only use the mono-lingual vectors to predict relations, and [P]CNN-En and [P]CNN-Zh models.

Fig. 3 shows the aggregated precision/recall curves of the four models for both CNN and PCNN. From the figure, we find that:

(1) MNRE-En and MNRE-Zh outperform [P]CNN-En and [P]CNN-Zh almost in entire range of recall. It indicates that by jointly training with multi-lingual attention, both Chinese and English relation extractors are beneficial from those sentences from the other language.

(2) Although [P]CNN-En underperforms as compared to [P]CNN-Zh, MNRE-En is comparable to MNRE-Zh by jointly training through multi-lingual attention. It demonstrates that both Chi-

Relation	#Sent-En	#Sent-Zh	CNN-En	CNN-Zh	MNRE-En	MNRE-Zh
Contains	993	6984	17.95	69.87	73.72	75.00
HeadquartersLocation	1949	210	43.04	0.00	41.77	50.63
Father	1833	983	64.71	77.12	86.27	83.01
CountryOfCitizenship	25322	15805	95.22	93.23	98.41	98.21

Table 5: Detailed results (precision@1) of some specific relations. #Sent-En and #Sent-Zh indicate the numbers of English/Chinese sentences which are labelled with the relations.

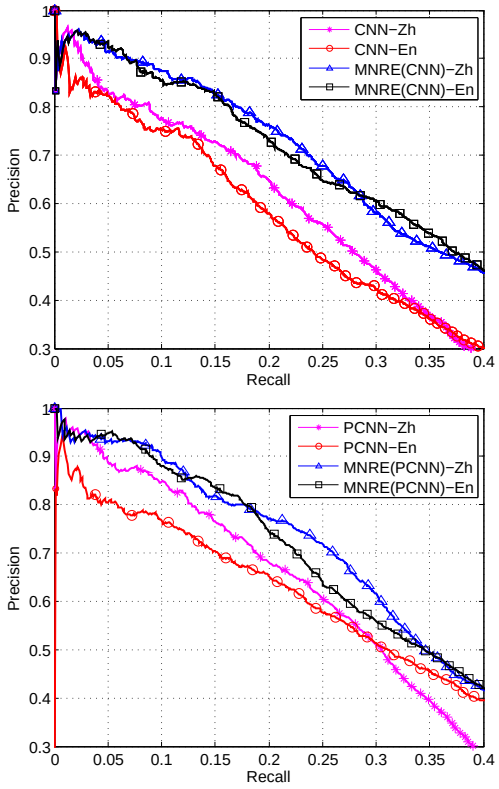


Figure 3: Top: Aggregate precision/recall curves of CNN-En, CNN-Zh, MNRE(CNN)-En and MNRE(CNN)-Zh. Bottom: Aggregate precision/recall curves of PCNN-En, PCNN-Zh, MNRE(PCNN)-En and MNRE(PCNN)-Zh.

nese and English relation extractors can take full advantages of texts in both languages via our propose multi-lingual attention scheme.

Table 5 shows the detailed results (in precision@1) of some specific relations of which the training instances are un-balanced on English and Chinese sides. From the table, we can see that:

(1) For the relation Contains of which the number of English training instances is only $1/7$ of Chinese ones, CNN-En gets much worse performance as compared to CNN-Zh due to the lack of training data. Nevertheless, by jointly training through multi-lingual attention, MNRE(CNN)-En is comparable to and slightly better than MNRE(CNN)-Zh.

(2) For the relation HeadquartersLocation of which the number of Chinese training instances is only $1/9$ of English ones, CNN-Zh even cannot predict any correct results. The reason is perhaps that, CNN-Zh of the relation is not sufficiently trained because there are only 210 Chinese training instances for this relation. Similarly, by jointly training through multi-lingual attention, MNRE(CNN)-En and MNRE(CNN)-Zh both achieve promising results.

(3) For the relations Father and CountryOfCitizenship of which the sentence number in English and Chinese are not so un-balanced, our MNRE can still improve the performance of relation extraction on both English and Chinese sides.

4.5 Comparison of Relation Matrix

For relation prediction, we use two kinds of relation matrices including: **M** that considers the global consistency of relations, and **R** that considers the specific characteristics of relations for each language. To measure the effect of the two relation matrices, we compare the performance of MNRE using the both matrices with those only using **M** (MNRE-M) and only using **R** (MNRE-R).

Fig. 4 shows the precision-recall curves for each method. From the figure, we observe that:t

(1) The performance of MNRE-M is much worse than both MNRE-R and MNRE. It indicates that we cannot just use global relation matrix for relation prediction. The reason is that each language has its own specific characteristics to express relation patterns, which cannot be well integrated into a single relation matrix.

(2) MNRE(CNN)-R has similar performance as compared to MNRE(CNN) when the recall is low. However, it has a sharp decline when the recall reaches 0.25. It suggests there also exists global consistency of relation patterns among languages which cannot be neglected. Hence, we should combine both **M** and **R** together for multi-lingual relation extraction, as proposed in our MNRE

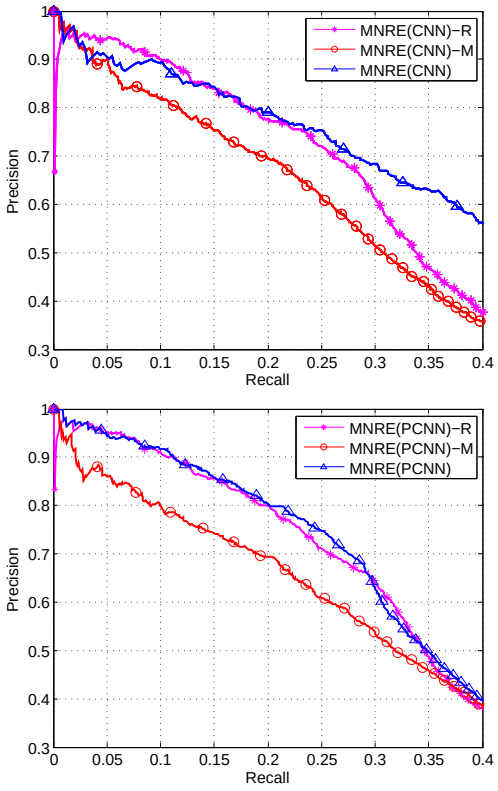


Figure 4: Top: Aggregated precion/recall curves of MNRE(CNN)-M, MNRE(CNN)-R and MNRE. Bottom: Aggregated precion/recall curves of MNRE(PCNN)-M, MNRE(PCNN)-R and MNRE(PCNN).

framework.

5 Conclusion

In this paper, we introduce a neural relation extraction framework with multi-lingual attention to take pattern consistency and complementarity among multiple languages into consideration. We evaluate our framework on multi-lingual relation extraction task, and the results show that our framework can effectively model relation patterns among languages and achieve state-of-the-art results.

We will explore the following directions as future work: (1) In this paper, we only consider sentence-level multi-lingual attention for relation extraction. In fact, we find that the word alignment information may be also helpful for capturing relation patterns. Hence, the word-level multi-lingual attention, which may discover implicit alignments between words in multiple languages, will further improve multi-lingual relation extraction. We will explore the effectiveness of word-level multi-lingual attention for relation extraction as our future work. (2) MNRE can be flexibly implemented in the scenario of multiple languages, and this paper focuses on two languages of English and Chinese. In future, we will extend MNRE to more languages and explore its significance.

Acknowledgments

This work is supported by the 973 Program (No. 2014CB340501), the National Natural Science Foundation of China (NSFC No. 61572273, 61532010), and the Key Technologies Research and Development Program of China (No. 2014BAK04B03). This work is also funded by the Natural Science Foundation of China (NSFC) and the German Research Foundation(DFG) in Project Crossmodal Learning, NSFC 61621136008 / DFC TRR-169.

References

Erik Boiy and Marie-Francine Moens. 2009. A machine learning approach to sentiment analysis in multilingual web texts. *Information retrieval* 12(5):526–558.

Florian Boudin, Stéphane Huet, and Juan-Manuel Torres-Moreno. 2011. A graph-based approach to cross-language multi-document summarization. *Polibits* (43):113–118.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Meiping Dong, Yong Cheng, Yang Liu, Jia Xu, Maosong Sun, Tatsuya Izuha, and Jie Hao. 2014. Query lattice for translation retrieval. In *Proceedings of COLING*. pages 2031–2041.

Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL*. volume 1, pages 626–634.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. *arXiv preprint arXiv:1503.06450* .

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of EMNLP*. pages 1774–1784.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* pages 1735–1780.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL-HLT*. pages 541–550.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*. volume 1, pages 2124–2133.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*. pages 1003–1011.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*. pages 148–163.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP-CoNLL*. pages 1201–1211.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*. pages 455–465.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2015. Multilingual relation extraction using compositional universal schema. *arXiv preprint arXiv:1511.06396* .

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *JMLR* 3(Feb):1083–1106.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*. pages 2335–2344.

Learning Structured Natural Language Representations for Semantic Parsing

Jianpeng Cheng[†] **Siva Reddy**[†] **Vijay Saraswat**[‡] and **Mirella Lapata**[†]
[†]School of Informatics, University of Edinburgh
[‡]IBM T.J. Watson Research
{jianpeng.cheng,siva.reddy}@ed.ac.uk, vsaraswa@us.ibm.com,
mlap@inf.ed.ac.uk

Abstract

We introduce a neural semantic parser which is interpretable and scalable. Our model converts natural language utterances to intermediate, domain-general natural language representations in the form of predicate-argument structures, which are induced with a transition system and subsequently mapped to target domains. The semantic parser is trained end-to-end using annotated logical forms or their denotations. We achieve the state of the art on SPADES and GRAPHQUESTIONS and obtain competitive results on GEOQUERY and WEBQUESTIONS. The induced predicate-argument structures shed light on the types of representations useful for semantic parsing and how these are different from linguistically motivated ones.[1]

1 Introduction

Semantic parsing is the task of mapping natural language utterances to machine interpretable meaning representations. Despite differences in the choice of meaning representation and model structure, most existing work conceptualizes semantic parsing following two main approaches. Under the first approach, an utterance is parsed and grounded to a meaning representation *directly* via learning a task-specific grammar (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Kwiatkowksi et al., 2010; Liang et al., 2011; Berant et al., 2013; Flanigan et al., 2014; Pasupat and Liang, 2015; Groschwitz et al., 2015). Under the second approach, the utterance is first parsed to an *intermediate* task-independent representation tied to a syntactic parser and then mapped to a grounded representation (Kwiatkowski et al., 2013; Reddy et al., 2016, 2014; Krishnamurthy and Mitchell, 2015; Gardner and Krishnamurthy, 2017). A merit of the two-stage approach is that it creates reusable intermediate interpretations, which potentially enables the handling of unseen words and knowledge transfer across domains (Bender et al., 2015).

The successful application of encoder-decoder models (Bahdanau et al., 2015; Sutskever et al., 2014) to a variety of NLP tasks has provided strong impetus to treat semantic parsing as a sequence transduction problem where an utterance is mapped to a target meaning representation in string format (Dong and Lapata, 2016; Jia and Liang, 2016; Kočiský et al., 2016). Such models still fall under the first approach, however, in contrast to previous work (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2011) they reduce the need for domain-specific assumptions, grammar learning, and more generally extensive feature engineering. But this modeling flexibility comes at a cost since it is no longer possible to interpret how meaning composition is performed. Such knowledge plays a critical role in understand modeling limitations so as to build better semantic parsers. Moreover, without any task-specific prior knowledge, the learning problem is fairly unconstrained, both in terms of the possible derivations to consider and in terms of the target output which can be ill-formed (e.g., with extra or missing brackets).

In this work, we propose a neural semantic parser that alleviates the aforementioned problems. Our model falls under the second class of approaches where utterances are first mapped to an intermediate representation containing natural language predicates. However, rather than using an external parser (Reddy et al., 2014, 2016) or manually specified CCG grammars (Kwiatkowski et al., 2013), we induce intermediate representations in the form of predicate-argument structures

[1]Our code is available at https://github.com/cheng6076/scanner.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 44–55
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1005

from data. This is achieved with a transition-based approach which by design yields recursive semantic structures, avoiding the problem of generating ill-formed meaning representations. Compared to most existing semantic parsers which employ a CKY style bottom-up parsing strategy (Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013; Berant et al., 2013; Berant and Liang, 2014), the transition-based approach we proposed does not require feature decomposition over structures and thereby enables the exploration of rich, non-local features. The output of the transition system is then grounded (e.g., to a knowledge base) with a neural mapping model under the assumption that grounded and ungrounded structures are isomorphic.[2] As a result, we obtain a neural model that jointly learns to parse natural language semantics and induce a lexicon that helps grounding.

The whole network is trained end-to-end on natural language utterances paired with annotated logical forms or their denotations. We conduct experiments on four datasets, including GEOQUERY (which has logical forms; Zelle and Mooney 1996), SPADES (Bisk et al., 2016), WEBQUESTIONS (Berant et al., 2013), and GRAPHQUESTIONS (Su et al., 2016) (which have denotations). Our semantic parser achieves the state of the art on SPADES and GRAPHQUESTIONS, while obtaining competitive results on GEOQUERY and WEBQUESTIONS. A side-product of our modeling framework is that the induced intermediate representations can contribute to rationalizing neural predictions (Lei et al., 2016). Specifically, they can shed light on the kinds of representations (especially predicates) useful for semantic parsing. Evaluation of the induced predicate-argument relations against syntax-based ones reveals that they are interpretable and meaningful compared to heuristic baselines, but they sometimes deviate from linguistic conventions.

2 Preliminaries

Problem Formulation Let $\mathcal{K}$ denote a knowledge base or more generally a reasoning system, and x an utterance paired with a grounded meaning representation G or its denotation y. Our problem is to learn a semantic parser that maps x to G via an intermediate ungrounded representation U. When G is executed against $\mathcal{K}$, it outputs denotation y.

[2]We discuss the merits and limitations of this assumption in Section 5

Predicate	Usage	Sub-categories
answer	denotation wrapper	—
type	entity type checking	*stateid*, *cityid*, *riverid*, etc.
all	querying for an entire set of entities	—
aggregation	one-argument meta predicates for sets	*count*, *largest*, *smallest*, etc.
logical connectors	two-argument meta predicates for sets	*intersect*, *union*, *exclude*

Table 1: List of domain-general predicates.

Grounded Meaning Representation We represent grounded meaning representations in FunQL (Kate et al., 2005) amongst many other alternatives such as lambda calculus (Zettlemoyer and Collins, 2005), λ-DCS (Liang, 2013) or graph queries (Holzschuher and Peinl, 2013; Harris et al., 2013). FunQL is a variable-free query language, where each predicate is treated as a function symbol that modifies an argument list. For example, the FunQL representation for the utterance *which states do not border texas* is:

answer(*exclude*(*state(all)*, *next_to*(*texas*)))

where *next_to* is a domain-specific binary predicate that takes one argument (i.e., the entity *texas*) and returns a *set* of entities (e.g., the states bordering Texas) as its denotation. *all* is a special predicate that returns a collection of entities. *exclude* is a predicate that returns the difference between two input sets.

An advantage of FunQL is that the resulting *s*-expression encodes semantic compositionality and derivation of the logical forms. This property makes FunQL logical forms convenient to be predicted with recurrent neural networks (Vinyals et al., 2015; Choe and Charniak, 2016; Dyer et al., 2016). However, FunQL is less expressive than lambda calculus, partially due to the elimination of variables. A more compact logical formulation which our method also applies to is λ-DCS (Liang, 2013). In the absence of anaphora and composite binary predicates, conversion algorithms exist between FunQL and λ-DCS. However, we leave this to future work.

Ungrounded Meaning Representation We also use FunQL to express ungrounded meaning representations. The latter consist primarily of natural language predicates and domain-general predicates. Assuming for simplicity that domain-general predicates share the same vocabulary

in ungrounded and grounded representations, the ungrounded representation for the example utterance is:

answer(*exclude*(*states(all)*, *border*(*texas*)))

where *states* and *border* are natural language predicates. In this work we consider five types of domain-general predicates illustrated in Table 1. Notice that domain-general predicates are often implicit, or represent extra-sentential knowledge. For example, the predicate *all* in the above utterance represents all states in the domain which are not mentioned in the utterance but are critical for working out the utterance denotation. Finally, note that for certain domain-general predicates, it also makes sense to extract natural language rationales (e.g., *not* is indicative for *exclude*). But we do not find this helpful in experiments.

In this work we constrain ungrounded representations to be structurally isomorphic to grounded ones. In order to derive the target logical forms, all we have to do is replacing predicates in the ungrounded representations with symbols in the knowledge base.

3 Modeling

In this section, we discuss our neural model which maps utterances to target logical forms. The semantic parsing task is decomposed in two stages: we first explain how an utterance is converted to an intermediate representation (Section 3.1), and then describe how it is grounded to a knowledge base (Section 3.2).

3.1 Generating Ungrounded Representations

At this stage, utterances are mapped to intermediate representations with a transition-based algorithm. In general, the transition system generates the representation by following a derivation tree (which contains a set of applied rules) and some canonical generation order (e.g., depth-first). For FunQL, a simple solution exists since the representation itself encodes the derivation. Consider again *answer*(*exclude*(*states(all)*, *border*(*texas*))) which is tree structured. Each predicate (e.g., *border*) can be visualized as a non-terminal node of the tree and each entity (e.g., *texas*) as a terminal. The predicate *all* is a special case which acts as a terminal directly. We can generate the tree with a top-down, depth first transition system reminiscent of recurrent neural network grammars (RNNGs; Dyer et al. 2016). Similar to RNNG, our algorithm uses a buffer to store input tokens in the utterance and a stack to store partially completed trees. A major difference in our semantic parsing scenario is that tokens in the buffer are not fetched in a sequential order or removed from the buffer. This is because the lexical alignment between an utterance and its semantic representation is hidden. Moreover, some predicates cannot be clearly anchored to a token span. Therefore, we allow the generation algorithm to pick tokens and combine logical forms in arbitrary orders, conditioning on the entire set of sentential features. Alternative solutions in the traditional semantic parsing literature include a floating chart parser (Pasupat and Liang, 2015) which allows to construct logical predicates out of thin air.

Our transition system defines three actions, namely NT, TER, and RED, explained below.

NT(X) generates a Non-Terminal predicate. This predicate is either a natural language expression such as *border*, or one of the domain-general predicates exemplified in Table 1 (e.g., *exclude*). The type of predicate is determined by the placeholder X and once generated, it is pushed onto the stack and represented as a non-terminal followed by an open bracket (e.g., *'border('*). The open bracket will be closed by a reduce operation.

TER(X) generates a TERminal entity or the special predicate *all*. Note that the terminal choice does not include variable (e.g., $0, $1), since FunQL is a variable-free language which sufficiently captures the semantics of the datasets we work with. The framework could be extended to generate directly acyclic graphs by incorporating variables with additional transition actions for handling variable mentions and co-reference.

RED stands for REDuce and is used for subtree completion. It recursively pops elements from the stack until an open non-terminal node is encountered. The non-terminal is popped as well, after which a composite term representing the entire subtree, e.g., *border(texas)*, is pushed back to the stack. If a RED action results in having no more open non-terminals left on the stack, the transition system terminates. Table 2 shows the transition actions used to generate our running example.

The model generates the ungrounded representation U conditioned on utterance x by recursively calling one of the above three actions. Note that U is defined by a sequence of actions (denoted

Sentence: *which states do not border texas*
Non-terminal symbols in buffer: *which*, *states*, *do*, *not*, *border*
Terminal symbols in buffer: *texas*

Stack	**Action**	NT choice	TER choice
	NT	*answer*	
answer (	NT	*exclude*	
answer (*exclude* (	NT	*states*	
answer (*exclude* (*states* (	TER		*all*
answer (*exclude* (*states* (*all*	RED		
answer (*exclude* (*states* (*all*)	NT	*border*	
answer (*exclude* (*states* (*all*) , *border* (	TER		*texas*
answer (*exclude* (*states* (*all*) , *border* (*texas*	RED		
answer (*exclude* (*states* (*all*) , *border* (*texas*)	RED		
answer (*exclude* (*states* (*all*) , *border* (*texas*))	RED		
answer (*exclude* (*states* (*all*) , *border* (*texas*)))			

Table 2: Actions taken by the transition system for generating the ungrounded meaning representation of the example utterance. Symbols in red indicate domain-general predicates.

by a) and a sequence of term choices (denoted by u) as shown in Table 2. The conditional probability $p(U|x)$ is factorized over time steps as:

$$p(U|x) = p(a, u|x) = \prod_{t=1}^{T} p(a_t|a_{<t}, x)p(u_t|a_{<t}, x)^{\mathbb{I}(a_t \neq \text{RED})} \quad (1)$$

where $\mathbb{I}$ is an indicator function.

To predict the actions of the transition system, we encode the input buffer with a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) and the output stack with a stack-LSTM (Dyer et al., 2015). At each time step, the model uses the representation of the transition system e_t to predict an action:

$$p(a_t|a_{<t}, x) \propto \exp(W_a \cdot e_t) \quad (2)$$

where e_t is the concatenation of the buffer representation b_t and the stack representation s_t. While the stack representation s_t is easy to retrieve as the top state of the stack-LSTM, obtaining the buffer representation b_t is more involved. This is because we do not have an explicit buffer representation due to the non-projectivity of semantic parsing. We therefore compute at each time step an adaptively weighted representation of b_t (Bahdanau et al., 2015) conditioned on the stack representation s_t. This buffer representation is then concatenated with the stack representation to form the system representation e_t.

When the predicted action is either NT or TER, an ungrounded term u_t (either a predicate or an entity) needs to be chosen from the candidate list depending on the specific placeholder X. To select a domain-general term, we use the same representation of the transition system e_t to compute a probability distribution over candidate terms:

$$p(u_t^{\text{GENERAL}}|a_{<t}, x) \propto \exp(W_p \cdot e_t) \quad (3)$$

To choose a natural language term, we directly compute a probability distribution of all natural language terms (in the buffer) conditioned on the stack representation s_t and select the most relevant term (Jia and Liang, 2016):

$$p(u_t^{\text{NL}}|a_{<t}, x) \propto \exp(s_t) \quad (4)$$

When the predicted action is RED, the completed subtree is composed into a single representation on the stack. For the choice of composition function, we use a single-layer neural network as in Dyer et al. (2015), which takes as input the concatenated representation of the predicate and argument of the subtree.

3.2 Generating Grounded Representations

Since we constrain the network to learn ungrounded structures that are isomorphic to the target meaning representation, converting ungrounded representations to grounded ones becomes a simple lexical mapping problem. For simplicity, hereafter we do not differentiate natural language and domain-general predicates.

To map an ungrounded term u_t to a grounded term g_t, we compute the conditional probability

of g_t given u_t with a bi-linear neural network:

$$p(g_t|u_t) \propto \exp \vec{u_t} \cdot W_{ug} \cdot \vec{g_t}^\top \quad (5)$$

where $\vec{u_t}$ is the contextual representation of the ungrounded term given by the bidirectional LSTM, $\vec{g_t}$ is the grounded term embedding, and W_{ug} is the weight matrix.

The above grounding step can be interpreted as learning a lexicon: the model exclusively relies on the intermediate representation U to predict the target meaning representation G without taking into account any additional features based on the utterance. In practice, U may provide sufficient contextual background for closed domain semantic parsing where an ungrounded predicate often maps to a single grounded predicate, but is a relatively impoverished representation for parsing large open-domain knowledge bases like Freebase. In this case, we additionally rely on a discriminative reranker which ranks the grounded representations derived from ungrounded representations (see Section 3.4).

3.3 Training Objective

When the target meaning representation is available, we directly compare it against our predictions and back-propagate. When only denotations are available, we compare surrogate meaning representations against our predictions (Reddy et al., 2014). Surrogate representations are those with the correct denotations. When there exist multiple surrogate representations,[3] we select one randomly and back-propagate. The global effect of the above update rule is close to maximizing the marginal likelihood of denotations, which differs from recent work on weakly-supervised semantic parsing based on reinforcement learning (Neelakantan et al., 2017).

Consider utterance x with ungrounded meaning representation U, and grounded meaning representation G. Both U and G are defined with a sequence of transition actions (same for U and G) and a sequence of terms (different for U and G). Recall that $a = [a_1, \cdots, a_n]$ denotes the transition action sequence defining U and G; let $u = [u_1, \cdots, u_k]$ denote the ungrounded terms (e.g., predicates), and $g = [g_1, \cdots, g_k]$ the grounded terms. We aim to maximize the likelihood of the grounded meaning representation $p(G|x)$ over all training examples. This likelihood can be decomposed into the likelihood of the grounded action sequence $p(a|x)$ and the grounded term sequence $p(g|x)$, which we optimize separately.

For the grounded action sequence (which by design is the same as the ungrounded action sequence and therefore the output of the transition system), we can directly maximize the log likelihood $\log p(a|x)$ for all examples:

$$\mathcal{L}_a = \sum_{x \in \mathcal{T}} \log p(a|x) = \sum_{x \in \mathcal{T}} \sum_{t=1}^{n} \log p(a_t|x) \quad (6)$$

where $\mathcal{T}$ denotes examples in the training data.

For the grounded term sequence g, since the intermediate ungrounded terms are latent, we maximize the expected log likelihood of the grounded terms $\sum_u [p(u|x) \log p(g|u, x)]$ for all examples, which is a lower bound of the log likelihood $\log p(g|x)$:

$$\begin{aligned} \mathcal{L}_g &= \sum_{x \in \mathcal{T}} \sum_{u} [p(u|x) \log p(g|u, x)] \\ &= \sum_{x \in \mathcal{T}} \sum_{u} \left[p(u|x) \sum_{t=1}^{k} \log p(g_t|u_t) \right] \end{aligned} \quad (7)$$

The final objective is the combination of $\mathcal{L}_a$ and $\mathcal{L}_g$, denoted as $\mathcal{L}_G = \mathcal{L}_a + \mathcal{L}_g$. We optimize this objective with the method described in Lei et al. (2016).

3.4 Reranker

As discussed above, for open domain semantic parsing, solely relying on the ungrounded representation would result in an impoverished model lacking sentential context useful for disambiguation decisions. For all Freebase experiments, we followed previous work (Berant et al., 2013; Berant and Liang, 2014; Reddy et al., 2014) in additionally training a discriminative ranker to re-rank grounded representations globally.

The discriminative ranker is a maximum-entropy model (Berant et al., 2013). The objective is to maximize the log likelihood of the correct answer y given x by summing over all grounded candidates G with denotation y (i.e.,$[\![G]\!]_{\mathcal{K}} = y$):

$$\mathcal{L}_y = \sum_{(x,y) \in \mathcal{T}} \log \sum_{[\![G]\!]_{\mathcal{K}} = y} p(G|x) \quad (8)$$

$$p(G|x) \propto \exp\{f(G, x)\} \quad (9)$$

where $f(G, x)$ is a feature function that maps pair (G, x) into a feature vector. We give details on the features we used in Section 4.2.

[3] The average Freebase surrogate representations obtained with highest denotation match (F1) is 1.4.

4 Experiments

In this section, we verify empirically that our semantic parser derives useful meaning representations. We give details on the evaluation datasets and baselines used for comparison. We also describe implementation details and the features used in the discriminative ranker.

4.1 Datasets

We evaluated our model on the following datasets which cover different domains, and use different types of training data, i.e., pairs of natural language utterances and grounded meanings or question-answer pairs.

GEOQUERY (Zelle and Mooney, 1996) contains 880 questions and database queries about US geography. The utterances are compositional, but the language is simple and vocabulary size small. The majority of questions include at most one entity. SPADES (Bisk et al., 2016) contains 93,319 questions derived from CLUEWEB09 (Gabrilovich et al., 2013) sentences. Specifically, the questions were created by randomly removing an entity, thus producing sentence-denotation pairs (Reddy et al., 2014). The sentences include two or more entities and although they are not very compositional, they constitute a large-scale dataset for neural network training. WEBQUESTIONS (Berant et al., 2013) contains 5,810 question-answer pairs. Similar to SPADES, it is based on Freebase and the questions are not very compositional. However, they are real questions asked by people on the Web. Finally, GRAPHQUESTIONS (Su et al., 2016) contains 5,166 question-answer pairs which were created by showing 500 Freebase graph queries to Amazon Mechanical Turk workers and asking them to paraphrase them into natural language.

4.2 Implementation Details

Amongst the four datasets described above, GEOQUERY has annotated logical forms which we directly use for training. For the other three datasets, we treat surrogate meaning representations which lead to the correct answer as gold standard. The surrogates were selected from a subset of candidate Freebase graphs, which were obtained by entity linking. Entity mentions in SPADES have been automatically annotated with Freebase entities (Gabrilovich et al., 2013). For WEBQUESTIONS and GRAPHQUESTIONS, we follow the procedure described in Reddy et al. (2016). We identify potential entity spans using seven handcrafted part-of-speech patterns and associate them with Freebase entities obtained from the Freebase/KG API.[4] We use a structured perceptron trained on the entities found in WEBQUESTIONS and GRAPHQUESTIONS to select the top 10 non-overlapping entity disambiguation possibilities. We treat each possibility as a candidate input utterance, and use the perceptron score as a feature in the discriminative reranker, thus leaving the final disambiguation to the semantic parser.

Apart from the entity score, the discriminative ranker uses the following basic features. The first feature is the likelihood score of a grounded representation aggregating all intermediate representations. The second set of features include the embedding similarity between the relation and the utterance, as well as the similarity between the relation and the question words. The last set of features includes the answer type as indicated by the last word in the Freebase relation (Xu et al., 2016).

We used the Adam optimizer for training with an initial learning rate of 0.001, two momentum parameters [0.99, 0.999], and batch size 1. The dimensions of the word embeddings, LSTM states, entity embeddings and relation embeddings are $[50, 100, 100, 100]$. The word embeddings were initialized with Glove embeddings (Pennington et al., 2014). All other embeddings were randomly initialized.

4.3 Results

Experimental results on the four datasets are summarized in Tables 3–6. We present comparisons of our system which we call SCANNER (as a shorthand for **S**ymboli**C** me**A**ni**N**g r**E**p**R**esentation) against a variety of models previously described in the literature.

GEOQUERY results are shown in Table 5. The first block contains symbolic systems, whereas neural models are presented in the second block. We report accuracy which is defined as the proportion of the utterance that are correctly parsed to their gold standard logical forms. All previous neural systems (Dong and Lapata, 2016; Jia and Liang, 2016) treat semantic parsing as a sequence transduction problem and use LSTMs to directly map utterances to logical forms. SCANNER yields performance improvements over these

[4] http://developers.google.com/freebase/

Models	F1
Berant et al. (2013)	35.7
Yao and Van Durme (2014)	33.0
Berant and Liang (2014)	39.9
Bast and Haussmann (2015)	49.4
Berant and Liang (2015)	49.7
Reddy et al. (2016)	50.3
Bordes et al. (2014)	39.2
Dong et al. (2015)	40.8
Yih et al. (2015)	52.5
Xu et al. (2016)	53.3
Neural Baseline	48.3
SCANNER	49.4

Table 3: WEBQUESTIONS results.

Models	F1
SEMPRE (Berant et al., 2013)	10.80
PARASEMPRE (Berant and Liang, 2014)	12.79
JACANA (Yao and Van Durme, 2014)	5.08
Neural Baseline	16.24
SCANNER	17.02

Table 4: GRAPHQUESTIONS results. Numbers for comparison systems are from Su et al. (2016).

Models	Accuracy
Zettlemoyer and Collins (2005)	79.3
Zettlemoyer and Collins (2007)	86.1
Kwiatkowksi et al. (2010)	87.9
Kwiatkowski et al. (2011)	88.6
Kwiatkowski et al. (2013)	88.0
Zhao and Huang (2015)	88.9
Liang et al. (2011)	91.1
Dong and Lapata (2016)	84.6
Jia and Liang (2016)	85.0
Jia and Liang (2016) with extra data	89.1
SCANNER	86.7

Table 5: GEOQUERY results.

Models	F1
Unsupervised CCG (Bisk et al., 2016)	24.8
Semi-supervised CCG (Bisk et al., 2016)	28.4
Neural baseline	28.6
Supervised CCG (Bisk et al., 2016)	30.9
Rule-based system (Bisk et al., 2016)	31.4
SCANNER	31.5

Table 6: SPADES results.

systems when using comparable data sources for training. Jia and Liang (2016) achieve better results with synthetic data that expands GEOQUERY; we could adopt their approach to improve model performance, however, we leave this to future work.

Table 6 reports SCANNER's performance on SPADES. For all Freebase related datasets we use average F1 (Berant et al., 2013) as our evaluation metric. Previous work on this dataset has used a semantic parsing framework similar to ours where natural language is converted to an intermediate syntactic representation and then grounded to Freebase. Specifically, Bisk et al. (2016) evaluate the effectiveness of four different CCG parsers on the semantic parsing task when varying the amount of supervision required. As can be seen, SCANNER outperforms all CCG variants (from unsupervised to fully supervised) without having access to any manually annotated derivations or lexicons. For fair comparison, we also built a neural baseline that encodes an utterance with a recurrent neural network and then predicts a grounded meaning representation directly (Ture and Jojic, 2016; Yih et al., 2016). Again, we observe that SCANNER outperforms this baseline.

Results on WEBQUESTIONS are summarized in Table 3. SCANNER obtains performance on par with the best symbolic systems (see the first block in the table). It is important to note that Bast and Haussmann (2015) develop a question answering system, which contrary to ours cannot produce meaning representations whereas Berant and Liang (2015) propose a sophisticated agenda-based parser which is trained borrowing ideas from imitation learning. SCANNER is conceptually similar to Reddy et al. (2016) who also learn a semantic parser via intermediate representations which they generate based on the output of a dependency parser. SCANNER performs competitively despite not having access to any linguistically-informed syntactic structures. The second block in Table 3 reports the results of several neural systems. Xu et al. (2016) represent the state of the art on WEBQUESTIONS. Their system uses Wikipedia to prune out erroneous candidate answers extracted from Freebase. Our model would also benefit from a similar post-processing step. As in previous experiments, SCANNER outperforms the neural baseline, too.

Finally, Table 4 presents our results on GRAPHQUESTIONS. We report F1 for SCANNER, the neural baseline model, and three symbolic systems presented in Su et al. (2016). SCANNER achieves a new state of the art on this dataset with a gain of 4.23 F1 points over the best previously reported model.

4.4 Analysis of Intermediate Representations

Since a central feature of our parser is that it learns intermediate representations with natural language predicates, we conducted additional experiments in order to inspect their quality. For GEOQUERY

Metrics	Accuracy
Exact match	79.3
Structure match	89.6
Token match	96.5

Table 7: GEOQUERY evaluation of ungrounded meaning representations. We report accuracy against a manually created gold standard.

which contains only 280 test examples, we manually annotated intermediate representations for the test instances and evaluated the learned representations against them. The experimental setup aims to shows how humans can participate in improving the semantic parser with feedback at the intermediate stage. In terms of evaluation, we use three metrics shown in Table 7. The first row shows the percentage of exact matches between the predicted representations and the human annotations. The second row refers to the percentage of structure matches, where the predicted representations have the same structure as the human annotations, but may not use the same lexical terms. Among structurally correct predictions, we additionally compute how many tokens are correct, as shown in the third row. As can be seen, the induced meaning representations overlap to a large extent with the human gold standard.

We also evaluated the intermediate representations created by SCANNER on the other three (Freebase) datasets. Since creating a manual gold standard for these large datasets is time-consuming, we compared the induced representations against the output of a syntactic parser. Specifically, we converted the questions to event-argument structures with EASYCCG (Lewis and Steedman, 2014), a high coverage and high accuracy CCG parser. EASYCCG extracts predicate-argument structures with a labeled F-score of 83.37%. For further comparison, we built a simple baseline which identifies predicates based on the output of the Stanford POS-tagger (Manning et al., 2014) following the ordering VBD $\gg$ VBN $\gg$ VB $\gg$ VBP $\gg$ VBZ $\gg$ MD.

As shown in Table 8, on SPADES and WEBQUESTIONS, the predicates learned by our model match the output of EASYCCG more closely than the heuristic baseline. But for GRAPHQUESTIONS which contains more compositional questions, the mismatch is higher. However, since the key idea of our model is to capture salient meaning for the task at hand rather than strictly obey syntax, we would not expect the

Dataset	SCANNER	Baseline
SPADES	51.2	45.5
–*conj* (1422)	56.1	66.4
–*control* (132)	28.3	40.5
–*pp* (3489)	46.2	23.1
–*subord* (76)	37.9	52.9
WEBQUESTIONS	42.1	25.5
GRAPHQUESTIONS	11.9	15.3

Table 8: Evaluation of predicates induced by SCANNER against EASYCCG. We report F1(%) across datasets. For SPADES, we also provide a breakdown for various utterance types.

predicates induced by our system to entirely agree with those produced by the syntactic parser. To further analyze how the learned predicates differ from syntax-based ones, we grouped utterances in SPADES into four types of linguistic constructions: coordination (*conj*), control and raising (*control*), prepositional phrase attachment (*pp*), and subordinate clauses (*subord*). Table 8 also shows the breakdown of matching scores per linguistic construction, with the number of utterances in each type. In Table 9, we provide examples of predicates identified by SCANNER, indicating whether they agree or not with the output of EASYCCG. As a reminder, the task in SPADES is to predict the entity masked by a *blank* symbol (_).

As can be seen in Table 8, the matching score is relatively high for utterances involving coordination and prepositional phrase attachments. The model will often identify informative predicates (e.g., nouns) which do not necessarily agree with linguistic intuition. For example, in the utterance *wilhelm_maybach and his son _ started maybach in 1909* (see Table 9), SCANNER identifies the predicate-argument structure *son(wilhelm_maybach)* rather than *started(wilhelm_maybach)*. We also observed that the model struggles with control and subordinate constructions. It has difficulty distinguishing control from raising predicates as exemplified in the utterance *ceo john_thain agreed to leave _* from Table 9, where it identifies the raising predicate *agreed*. For subordinate clauses, SCANNER tends to take shortcuts identifying as predicates words closest to the *blank* symbol.

5 Discussion

We presented a neural semantic parser which converts natural language utterances to grounded meaning representations via intermediate predicate-argument structures. Our model

conj	the boeing_company was founded in 1916 and is headquartered in __ , illinois . nstar was founded in 1886 and is based in boston , __ . the __ is owned and operated by zuffa_,_llc , headquarted in las_vegas , nevada . hugh attended __ and then shifted to uppingham_school in england .	__ was incorporated in 1947 and is based in new_york_city . the ifbb was formed in 1946 by president ben_weider and his brother __ . wilhelm_maybach and his son __ started maybach in 1909 . __ was founded in 1996 and is headquartered in chicago .
control	__ threatened to kidnap russ . __ has also been confirmed to play captain_haddock . hoffenberg decided to leave __ . __ is reportedly trying to get impregnated by djimon now . for right now , __ are inclined to trust obama to do just that .	__ agreed to purchase wachovia_corp . ceo john_thain agreed to leave __ . so nick decided to create __ . salva later went on to make the non clown-based horror __ . eddie dumped debbie to marry __ when carrie was 2 .
pp	__ is the home of the university_of_tennessee . chu is currently a physics professor at __ . youtube is based in __ , near san_francisco , california . mathematica is a product of __ .	jobs will retire from __ . the nab is a strong advocacy group in __ . this one starred robert_reed , known mostly as __ . __ is positively frightening as detective bud_white .
subord	the__ is a national testing board that is based in toronto . __ is a corporation that is wholly owned by the city_of_edmonton . unborn is a scary movie that stars __ . __ 's third wife was actress melina_mercouri , who died in 1994 . sure , there were __ who liked the shah .	founded the __ , which is now also a designated terrorist group . __ is an online bank that ebay owns . zoya_akhtar is a director , who has directed the upcoming movie __ . imelda_staunton , who plays __ , is genius . __ is the important president that american ever had . plus mitt_romney is the worst governor that __ has had .

Table 9: Informative predicates identified by SCANNER in various types of utterances. Yellow predicates were identified by both SCANNER and EASYCCG, red predicates by SCANNER alone, and green predicates by EASYCCG alone.

essentially jointly learns how to parse natural language semantics and the lexicons that help grounding. Compared to previous neural semantic parsers, our model is more interpretable as the intermediate structures are useful for inspecting what the model has learned and whether it matches linguistic intuition.

An assumption our model imposes is that ungrounded and grounded representations are structurally isomorphic. An advantage of this assumption is that tokens in the ungrounded and grounded representations are strictly aligned. This allows the neural network to focus on parsing and lexical mapping, sidestepping the challenging structure mapping problem which would result in a larger search space and higher variance. On the negative side, the structural isomorphism assumption restricts the expressiveness of the model, especially since one of the main benefits of adopting a two-stage parser is the potential of capturing domain-independent semantic information via the intermediate representation. While it would be challenging to handle drastically non-isomorphic structures in the current model, it is possible to perform local structure matching, i.e., when the mapping between natural language and domain-specific predicates is many-to-one or one-to-many. For instance, Freebase does not contain a relation representing *daughter*, using instead two relations representing *female* and *child*. Previous work (Kwiatkowski et al., 2013) models such cases by introducing collapsing (for many-to-one mapping) and expansion (for one-to-many mapping) operators. Within our current framework, these two types of structural mismatches can be handled with semi-Markov assumptions (Sarawagi and Cohen, 2005; Kong et al., 2016) in the parsing (i.e., predicate selection) and the grounding steps, respectively. Aside from relaxing strict isomorphism, we would also like to perform cross-domain semantic parsing where the first stage of the semantic parser is shared across domains.

Acknowledgments We would like to thank three anonymous reviewers, members of the Edinburgh ILCC and the IBM Watson, and Abulhair Saparov for feedback. The support of the European Research Council under award number 681760 "Translating Multiple Modalities into Text" is gratefully acknowledged.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly

learning to align and translate. In *Proceedings of ICLR 2015*. San Diego, California.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on Freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, pages 1431–1440.

Emily M Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics*. London, UK, pages 239–249.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, pages 1533–1544.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland, pages 1415–1425.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics* 3:545–558.

Yonatan Bisk, Siva Reddy, John Blitzer, Julia Hockenmaier, and Mark Steedman. 2016. Evaluating induced CCG parsers on grounded semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 2022–2027.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 615–620.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria, pages 423–433.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 2331–2336.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 33–43.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 260–269.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 334–343.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 199–209.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland, pages 1426–1436.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0) .

Matt Gardner and Jayant Krishnamurthy. 2017. Open-Vocabulary Semantic Parsing with both Distributional Statistics and Formal Knowledge. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. San Francisco, California, pages 3195–3201.

Jonas Groschwitz, Alexander Koller, and Christoph Teichmann. 2015. Graph parsing with s-graph grammars. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1481–1490.

Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. 2013. SPARQL 1.1 query language. *W3C recommendation* 21(10).

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

Florian Holzschuher and René Peinl. 2013. Performance of graph query languages: comparison of

cypher, gremlin and native access in Neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM, pages 195–204.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 12–22.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to Transform Natural to Formal Languages. In *Proceedings for the 20th National Conference on Artificial Intelligence*. Pittsburgh, Pennsylvania, pages 1062–1068.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *Proceedings of ICLR 2016*. San Juan, Puerto Rico.

Tomáš Kočiský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 1078–1087.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea, pages 754–765.

Jayant Krishnamurthy and Tom M. Mitchell. 2015. Learning a Compositional Semantics for Freebase with an Open Predicate Vocabulary. *Transactions of the Association for Computational Linguistics* 3:257–270.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA, pages 1223–1233.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-Fly Ontology Matching. In *Proceedings of Empirical Methods on Natural Language Processing*. pages 1545–1556.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, pages 1512–1523.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 107–117.

Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 990–1000.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408* .

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, pages 590–599.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland, pages 55–60.

Arvind Neelakantan, Quoc V Le, Martin Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. In *Proceedings of ICLR 2017*. Toulon, France.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1470–1480.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 1532–1543.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics* 2:377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics* 4:127–140.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17*, MIT Press, pages 1185–1192.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for

qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, pages 562–572.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, MIT Press, pages 3104–3112.

Ferhan Ture and Oliver Jojic. 2016. Simple and effective question answering with recurrent neural networks. *arXiv preprint arXiv:1606.05029* .

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*. MIT Press, pages 2773–2781.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. New York City, USA, pages 439–446.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on Freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 2326–2336.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland, pages 956–966.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1321–1331.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany, pages 201–206.

John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the 13th National Conference on Artificial Intelligence*. Portland, Oregon, pages 1050–1055.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic, pages 678–687.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of 21st Conference in Uncertainilty in Artificial Intelligence*. Edinburgh, Scotland, pages 658–666.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado, pages 1416–1421.

Morph-fitting: Fine-Tuning Word Vector Spaces with Simple Language-Specific Rules

Ivan Vulić[1] , **Nikola Mrkšić**[1], **Roi Reichart**[2]
Diarmuid Ó Séaghdha[3], **Steve Young**[1] , **Anna Korhonen**[1]
[1] University of Cambridge [2] Technion, Israel Institute of Technology [3] Apple Inc.
{iv250,nm480,sjy11,alk23}@cam.ac.uk
doseaghdha@apple.com roiri@ie.technion.ac.il

Abstract

Morphologically rich languages accentuate two properties of distributional vector space models: 1) the difficulty of inducing accurate representations for low-frequency word forms; and 2) insensitivity to distinct lexical relations that have similar distributional signatures. These effects are detrimental for language understanding systems, which may infer that *inexpensive* is a rephrasing for *expensive* or may not associate *acquire* with *acquires*. In this work, we propose a novel *morph-fitting* procedure which moves past the use of curated semantic lexicons for improving distributional vector spaces. Instead, our method injects morphological constraints generated using simple language-specific rules, pulling *inflectional* forms of the same word close together and pushing *derivational antonyms* far apart. In intrinsic evaluation over four languages, we show that our approach: **1)** improves low-frequency word estimates; and **2)** boosts the semantic quality of the entire word vector collection. Finally, we show that morph-fitted vectors yield large gains in the downstream task of *dialogue state tracking*, highlighting the importance of morphology for tackling long-tail phenomena in language understanding tasks.

1 Introduction

Word representation learning has become a research area of central importance in natural language processing (NLP), with its usefulness demonstrated across many application areas such as parsing (Chen and Manning, 2014; Johannsen et al., 2015), machine translation (Zou et al., 2013), and many others (Turian et al., 2010; Collobert et al., 2011). Most prominent word representation techniques are grounded in the *distributional hypothesis* (Harris, 1954), relying on word co-occurrence information in large textual corpora (Curran, 2004; Turney and Pantel, 2010; Mikolov et al., 2013; Mnih and Kavukcuoglu, 2013; Levy and Goldberg, 2014; Schwartz et al., 2015, i.a.).

Morphologically rich languages, in which "substantial grammatical information... is expressed at word level" (Tsarfaty et al., 2010), pose specific challenges for NLP. This is not always considered when techniques are evaluated on languages such as English or Chinese, which do not have rich morphology. In the case of distributional vector space models, morphological complexity brings two challenges to the fore:

1. Estimating Rare Words: A single lemma can have many different surface realisations. Naively treating each realisation as a separate word leads to sparsity problems and a failure to exploit their shared semantics. On the other hand, lemmatising the entire corpus can obfuscate the differences that exist between different word forms even though they share some aspects of meaning.

2. Embedded Semantics: Morphology can encode semantic relations such as antonymy (e.g. *literate* and *illiterate*, *expensive* and *inexpensive*) or (near-)synonymy (*north*, *northern*, *northerly*).

In this work, we tackle the two challenges jointly by introducing a *resource-light* vector space fine-tuning procedure termed *morph-fitting*. The proposed method does not require curated knowledge bases or gold lexicons. Instead, it makes use of the observation that morphology implicitly encodes semantic signals pertaining to synonymy (e.g., German word inflections *katalanisch, katalanischem, katalanischer* denote the same semantic concept in different grammatical roles), and antonymy (e.g., *mature* vs. *immature*), capitalising on the

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 56–68
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1006

en_expensive	de_teure	it_costoso	en_slow	de_langsam	it_lento	en_book	de_buch	it_libro
costly	teuren	dispendioso	fast	allmählich	lentissimo	books	sachbuch	romanzo
costlier	kostspielige	remunerativo	slowness	rasch	lenta	memoir	buches	racconto
cheaper	aufwändige	redditizio	slower	gemächlich	inesorabile	novel	romandebüt	volumetto
prohibitively	kostenintensive	rischioso	slowed	schnell	rapidissimo	storybooks	büchlein	saggio
pricey	aufwendige	costosa	slowing	explosionsartig	graduale	blurb	pamphlet	ecclesiaste
expensiveness	teures	costosa	slowing	langsamer	lenti	booked	bücher	libri
costly	teuren	costose	slowed	langsames	lente	rebook	büch	libra
costlier	teurem	costosi	slowness	langsame	lenta	booking	büche	librare
ruinously	teurer	dispendioso	slows	langsamem	veloce	rebooked	büches	libre
unaffordable	teurerer	dispendiose	idle	langsamen	rapido	books	büchen	librano

Table 1: The nearest neighbours of three example words (*expensive*, *slow* and *book*) in English, German and Italian before (top) and after (bottom) morph-fitting.

proliferation of word forms in morphologically rich languages. Formalised as an instance of the post-processing *semantic specialisation* paradigm (Faruqui et al., 2015; Mrkšić et al., 2016), morph-fitting is steered by a set of linguistic constraints derived from simple language-specific rules which describe (a subset of) morphological processes in a language. The constraints emphasise similarity on one side (e.g., by extracting *morphological* synonyms), and antonymy on the other (by extracting *morphological* antonyms), see Fig. 1 and Tab. 2.

The key idea of the fine-tuning process is to pull synonymous examples described by the constraints closer together in the transformed vector space, while at the same time pushing antonymous examples away from each other. The explicit post-hoc injection of morphological constraints enables: **a)** the estimation of more accurate vectors for low-frequency words which are linked to their high-frequency forms by the constructed constraints;[1] this tackles the data sparsity problem; and **b)** specialising the distributional space to distinguish between similarity and relatedness (Kiela et al., 2015), thus supporting language understanding applications such as *dialogue state tracking* (DST).[2]

As a post-processor, morph-fitting allows the integration of morphological rules with any distributional vector space in any language: it treats an input distributional word vector space as a black box and fine-tunes it so that the transformed space reflects the knowledge coded in the input morphological constraints (e.g., Italian words *rispettoso* and *irrispetosa* should be far apart in the transformed vector space, see Fig. 1). Tab. 1 illustrates the effects of morph-fitting by qualitative examples in three languages: the vast majority of nearest neighbours are "morphological" synonyms.

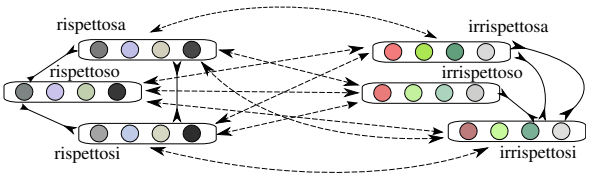


Figure 1: *Morph-fitting* in Italian. Representations for *rispettoso*, *rispettosa*, *rispettosi* (EN: *respectful*), are pulled closer together in the vector space (solid lines; ATTRACT constraints). At the same time, the model pushes them away from their antonyms (dashed lines; REPEL constraints) *irrispettoso*, *irrispettosa*, *irrispettosi* (EN: *disrespectful*), obtained through morphological affix transformation captured by language-specific rules (e.g., adding the prefix *ir-* typically negates the base word in Italian)

We demonstrate the efficacy of morph-fitting in four languages (English, German, Italian, Russian), yielding large and consistent improvements on benchmarking word similarity evaluation sets such as SimLex-999 (Hill et al., 2015), its multilingual extension (Leviant and Reichart, 2015), and SimVerb-3500 (Gerz et al., 2016). The improvements are reported for all four languages, and with a variety of input distributional spaces, verifying the robustness of the approach.

We then show that incorporating morph-fitted vectors into a state-of-the-art neural-network DST model results in improved tracking performance, especially for morphologically rich languages. We report an improvement of 4% on Italian, and 6% on German when using *morph-fitted* vectors instead of the distributional ones, setting a new state-of-the-art DST performance for the two datasets.[3]

[1] For instance, the vector for the word *katalanischem* which occurs only 9 times in the German Wikipedia will be pulled closer to the more reliable vectors for *katalanisch* and *katalanischer*, with frequencies of 2097 and 1383 respectively.

[2] Representation models that do not distinguish between synonyms and antonyms may have grave implications in downstream language understanding applications such as spoken dialogue systems: a user looking for *'an affordable Chinese restaurant in west Cambridge'* does not want a recommendation for *'an expensive Thai place in east Oxford'*.

[3] There are no readily available DST datasets for Russian.

2 Morph-fitting: Methodology

Preliminaries In this work, we focus on four languages with varying levels of morphological complexity: English (EN), German (DE), Italian (IT), and Russian (RU). These correspond to languages in the Multilingual SimLex-999 dataset. Vocabularies W_{en}, W_{de}, W_{it}, W_{ru} are compiled by retaining all word forms from the four Wikipedias with word frequency over 10, see Tab. 3. We then extract sets of linguistic constraints from these (large) vocabularies using a set of simple language-specific *if-then-else* rules, see Tab. 2.[4] These constraints (Sect. 2.2) are used as input for the vector space post-processing ATTRACT-REPEL algorithm (outlined in Sect. 2.1).

2.1 The ATTRACT-REPEL Model

The ATTRACT-REPEL model, proposed by Mrkšić et al. (2017b), is an extension of the PARAGRAM procedure proposed by Wieting et al. (2015). It provides a generic framework for incorporating *similarity* (e.g. *successful* and *accomplished*) and *antonymy* constraints (e.g. *nimble* and *clumsy*) into pre-trained word vectors. Given the initial vector space and collections of ATTRACT and REPEL constraints A and R, the model gradually modifies the space to bring the designated word vectors closer together or further apart. The method's cost function consists of three terms. The first term pulls the ATTRACT examples $(x_l, x_r) \in A$ closer together. If B_A denotes the current mini-batch of ATTRACT examples, this term can be expressed as:

$$A(\mathcal{B}_A) = \sum_{(x_l,x_r)\in\mathcal{B}_A} \left(ReLU\left(\delta_{att} + \mathbf{x}_l\mathbf{t}_l - \mathbf{x}_l\mathbf{x}_r\right)\right. \\ \left. + \; ReLU\left(\delta_{att} + \mathbf{x}_r\mathbf{t}_r - \mathbf{x}_l\mathbf{x}_r\right)\right)$$

where δ_{att} is the similarity margin which determines how much closer synonymous vectors should be to each other than to each of their respective negative examples. $ReLU(x) = \max(0, x)$ is the standard rectified linear unit (Nair and Hinton, 2010). The 'negative' example $\mathbf{t}_i$ for each word x_i in any ATTRACT pair is the word vector *closest* to $\mathbf{x}_i$ among the examples in the current mini-batch (distinct from its target synonym and $\mathbf{x}_i$ itself). This means that this term forces synonymous words from the in-batch ATTRACT constraints to be closer to one another than to any other word in the current mini-batch.

The second term pushes antonyms away from each other. If $(x_l, x_r) \in B_R$ is the current mini-batch of REPEL constraints, this term can be expressed as follows:

$$R(\mathcal{B}_R) = \sum_{(x_l,x_r)\in\mathcal{B}_R} \left(ReLU\left(\delta_{rpl} + \mathbf{x}_l\mathbf{x}_r - \mathbf{x}_l\mathbf{t}_r\right)\right. \\ \left. + \; ReLU\left(\delta_{rpl} + \mathbf{x}_l\mathbf{x}_r - \mathbf{x}_r\mathbf{t}_r\right)\right)$$

In this case, each word's 'negative' example is the (in-batch) word vector furthest away from it (and distinct from the word's target antonym). The intuition is that we want antonymous words from the input REPEL constraints to be *further away* from each other than from any other word in the current mini-batch; δ_{rpl} is now the *repel* margin.

The final term of the cost function serves to retain the abundance of semantic information encoded in the starting distributional space. If $\mathbf{x}_i^{init}$ is the initial distributional vector and $V(\mathcal{B})$ is the set of all vectors present in the given mini-batch, this term (per mini-batch) is expressed as follows:

$$R(\mathcal{B}_A, \mathcal{B}_R) = \sum_{\mathbf{x}_i\in V(\mathcal{B}_A\cup\mathcal{B}_R)} \lambda_{reg} \left\|\mathbf{x}_i^{init} - \mathbf{x}_i\right\|_2$$

where λ_{reg} is the L2 regularisation constant.[5] This term effectively *pulls* word vectors towards their initial (distributional) values, ensuring that relations encoded in initial vectors persist as long as they do not contradict the newly injected ones.

English	German	Italian
(discuss, discussed)	(schottisch, schottischem)	(golfo, golfi)
(laugh, laughing)	(damalige, damaligen)	(minato, minata)
(pacifist, pacifists)	(kombiniere, kombinierte)	(mettere, metto)
(evacuate, evacuated)	(schweigt, schweigst)	(crescono, cresci)
(evaluate, evaluates)	(hacken, gehackt)	(crediti, credite)
(dressed, undressed)	(stabil, unstabil)	(abitata, inabitato)
(similar, dissimilar)	(geformtes, ungeformt)	(realtà, irrealtà)
(formality, informality)	(relevant, irrelevant)	(attuato, inattuato)

Table 2: Example synonymous (inflectional; top) and antonymous (derivational; bottom) constraints.

2.2 Language-Specific Rules and Constraints

Semantic Specialisation with Constraints The fine-tuning ATTRACT-REPEL procedure is entirely driven by the input ATTRACT and REPEL sets of

[4] A native speaker can easily come up with these sets of morphological rules (or at least with a reasonable subset of them) without any linguistic training. What is more, the rules for DE, IT, and RU were created by non-native, non-fluent speakers with a limited knowledge of the three languages, exemplifying the simplicity and portability of the approach.

[5] We use hyperparameter values $\delta_{att} = 0.6$, $\delta_{rpl} = 0.0$, $\lambda_{reg} = 10^{-9}$ from prior work without fine-tuning. We train all models for 10 epochs with AdaGrad (Duchi et al., 2011).

	$\|W\|$	$\|A\|$	$\|R\|$
English	1,368,891	231,448	45,964
German	1,216,161	648,344	54,644
Italian	541,779	278,974	21,400
Russian	950,783	408,400	32,174

Table 3: Vocabulary sizes and counts of ATTRACT (A) and REPEL (R) constraints.

constraints. These can be extracted from a variety of semantic databases such as WordNet (Fellbaum, 1998), the Paraphrase Database (Ganitkevitch et al., 2013; Pavlick et al., 2015), or BabelNet (Navigli and Ponzetto, 2012; Ehrmann et al., 2014) as done in prior work (Faruqui et al., 2015; Wieting et al., 2015; Mrkšić et al., 2016, i.a.). In this work, we investigate another option: extracting constraints *without* curated knowledge bases in a spectrum of languages by exploiting inherent language-specific properties related to linguistic morphology. This relaxation ensures a wider portability of ATTRACT-REPEL to languages and domains without readily available or adequate resources.

Extracting ATTRACT Pairs The core difference between *inflectional* and *derivational morphology* can be summarised in a few lines as follows: the former refers to a set of processes through which the word form expresses meaningful syntactic information, e.g., verb tense, without any change to the semantics of the word. On the other hand, the latter refers to the formation of new words with semantic shifts in meaning (Schone and Jurafsky, 2001; Haspelmath and Sims, 2013; Lazaridou et al., 2013; Zeller et al., 2013; Cotterell and Schütze, 2017).

For the ATTRACT constraints, we focus on *inflectional* rather than on *derivational morphology* rules as the former preserve the full meaning of a word, modifying it only to reflect grammatical roles such as verb tense or case markers (e.g., *(en_read, en_reads)* or *(de_katalanisch, de_katalanischer)*). This choice is guided by our intent to fine-tune the original vector space in order to improve the embedded semantic relations.

We define two rules for English, widely recognised as morphologically simple (Avramidis and Koehn, 2008; Cotterell et al., 2016b). These are: **(R1)** *if* $w_1, w_2 \in W_{en}$, where $w_2 = w_1$ + *ing/ed/s*, *then* add (w_1, w_2) and (w_2, w_1) to the set of ATTRACT constraints A. This rule yields pairs such as *(look, looks), (look, looking), (look, looked)*.

If $w[:-1]$ is a function which strips the last character from word w, the second rule is: **(R2)** *if* w_1 ends with the letter *e and* $w_1 \in W_{en}$ *and* $w_2 \in W_{en}$, where $w_2 = w_1[:-1]$ + *ing/ed*, *then* add (w_1, w_2) and (w_2, w_1) to A. This creates pairs such as *(create, creating)* and *(create, created)*. Naturally, introducing more sophisticated rules is possible in order to cover for other special cases and morphological irregularities (e.g., *sweep / swept*), but in all our EN experiments, A is based on the two simple EN rules R1 and R2.

The other three languages, with more complicated morphology, yield a larger number of rules. In Italian, we rely on the sets of rules spanning: (1) regular formation of plural (*libro / libri*); (2) regular verb conjugation (*aspettare / aspettiamo*); (3) regular formation of past participle (*aspettare / aspettato*); and (4) rules regarding grammatical gender (*bianco / bianca*). Besides these, another set of rules is used for German and Russian: (5) regular declension (e.g., *asiatisch / asiatischem*).

Extracting REPEL Pairs As another source of implicit semantic signals, W also contains words which represent *derivational antonyms*: e.g., two words that denote concepts with opposite meanings, generated through a derivational process. We use a standard set of EN "antonymy" prefixes: AP_{en} = *{dis, il, un, in, im, ir, mis, non, anti}* (Fromkin et al., 2013). *If* $w_1, w_2 \in W_{en}$, where w_2 is generated by adding a prefix from AP_{en} to w_1, *then* (w_1, w_2) and (w_2, w_1) are added to the set of REPEL constraints R. This rule generates pairs such as *(advantage, disadvantage)* and *(regular, irregular)*. An additional rule replaces the suffix *-ful* with *-less*, extracting antonyms such as *(careful, careless)*.

Following the same principle, we use AP_{de} = *{un, nicht, anti, ir, in, miss}*, AP_{it} = *{in, ir, im, anti}*, and AP_{ru} = *{*не, анти*}*. For instance, this generates an IT pair *(rispettoso, irrispettoso)* (see Fig. 1). For DE, we use another rule targeting suffix replacement: *-voll* is replaced by *-los*.

We further expand the set of REPEL constraints by transitively combining antonymy pairs from the previous step with inflectional ATTRACT pairs. This step yields additional constraints such as *(rispettosa, irrispettosi)* (see Fig. 1). The final A and R constraint counts are given in Tab. 3. The full sets of rules are available as supplemental material.

3 Experimental Setup

Training Data and Setup For each of the four languages we train the skip-gram with negative sampling (SGNS) model (Mikolov et al., 2013)

on the latest Wikipedia dump of each language. We induce 300-dimensional word vectors, with the frequency cut-off set to 10. The vocabulary sizes $|W|$ for each language are provided in Tab. 3.[6] We label these collections of vectors SGNS-LARGE.

Other Starting Distributional Vectors We also analyse the impact of *morph-fitting* on other collections of well-known EN word vectors. These vectors have varying vocabulary coverage and are trained with different architectures. We test standard distributional models: Common-Crawl GloVe (Pennington et al., 2014), SGNS vectors (Mikolov et al., 2013) with various contexts (*BOW* = bag-of-words; *DEPS* = dependency contexts), and training data (*PW* = Polyglot Wikipedia from Al-Rfou et al. (2013); *8B* = 8 billion token `word2vec` corpus), following (Levy and Goldberg, 2014) and (Schwartz et al., 2015). We also test the symmetric-pattern based vectors of Schwartz et al. (2016) (*SymPat-Emb*), count-based PMI-weighted vectors reduced by SVD (Baroni et al., 2014) (*Count-SVD*), a model which replaces the context modelling function from CBOW with bidirectional LSTMs (Melamud et al., 2016) (*Context2Vec*), and two sets of EN vectors trained by injecting multilingual information: *BiSkip* (Luong et al., 2015) and *MultiCCA* (Faruqui and Dyer, 2014).

We also experiment with standard well-known distributional spaces in other languages (IT and DE), available from prior work (Dinu et al., 2015; Luong et al., 2015; Vulić and Korhonen, 2016a).

Morph-fixed Vectors A baseline which utilises an equal amount of knowledge as morph-fitting, termed *morph-fixing*, fixes the vector of each word to the distributional vector of its most frequent inflectional synonym, tying the vectors of low-frequency words to their more frequent inflections. For each word w_1, we construct a set of $M+1$ words $W_{w_1} = \{w_1, w'_1, \ldots, w'_M\}$ consisting of the word w_1 itself and all M words which co-occur with w_1 in the ATTRACT constraints. We then choose the word w'_{max} from the set W_{w_1} with the maximum frequency in the training data, and fix all other word vectors in W_{w_1} to its word vector. The morph-fixed vectors (MFIX) serve as our primary baseline, as they outperformed another straightforward baseline based on *stemming* across all of our intrinsic and extrinsic experiments.

Morph-fitting Variants We analyse two variants of morph-fitting: (1) using ATTRACT constraints only (MFIT-A), and (2) using both ATTRACT and REPEL constraints (MFIT-AR).

4 Intrinsic Evaluation: Word Similarity

Evaluation Setup and Datasets The first set of experiments intrinsically evaluates *morph-fitted* vector spaces on word similarity benchmarks, using Spearman's rank correlation as the evaluation metric. First, we use the SimLex-999 dataset, as well as SimVerb-3500, a recent EN verb pair similarity dataset providing similarity ratings for 3,500 verb pairs.[7] SimLex-999 was translated to DE, IT, and RU by Leviant and Reichart (2015), and they crowdsourced similarity scores from native speakers. We use this dataset for our multilingual evaluation.[8]

Morph-fitting EN Word Vectors As the first experiment, we morph-fit a wide spectrum of EN distributional vectors induced by various architectures (see Sect. 3). The results on SimLex and SimVerb are summarised in Tab. 4. The results with EN SGNS-LARGE vectors are shown in Fig. 3a. Morph-fitted vectors bring consistent improvement across all experiments, regardless of the quality of the initial distributional space. This finding confirms that the method is robust: its effectiveness does not depend on the architecture used to construct the initial space. To illustrate the improvements, note that the best score on SimVerb for a model trained on running text is achieved by *Context2vec* ($\rho = 0.388$); injecting morphological constraints into this vector space results in a gain of 7.1 ρ points.

Experiments on Other Languages We next extend our experiments to other languages, testing both morph-fitting variants. The results are summarised in Tab. 5, while Fig. 3a-3d show results for the morph-fitted SGNS-LARGE vectors. These scores confirm the effectiveness and robustness of morph-fitting across languages, suggesting that the idea of fitting to morphological constraints is indeed language-agnostic, given the set of language-specific rule-based constraints. Fig. 3 also demon-

[6] Other SGNS parameters were set to standard values (Baroni et al., 2014; Vulić and Korhonen, 2016b): 15 epochs, 15 negative samples, global learning rate: .025, subsampling rate: $1e-4$. Similar trends in results persist with $d = 100, 500$.

[7] Unlike other gold standard resources such as WordSim-353 (Finkelstein et al., 2002) or MEN (Bruni et al., 2014), SimLex and SimVerb provided explicit guidelines to discern between semantic similarity and association, so that related but non-similar words (e.g. *cup* and *coffee*) have a low rating.

[8] Since Leviant and Reichart (2015) re-scored the original EN SimLex, we use their EN SimLex version for consistency.

Vectors	Evaluation	
	SimLex-999	SimVerb-3500
1. SG-BOW2-PW (300) (Mikolov et al., 2013)	.339 → **.439**	.277 → **.381**
2. GloVe-6B (300) (Pennington et al., 2014)	.324 → **.438**	.286 → **.405**
3. Count-SVD (500) (Baroni et al., 2014)	.267 → **.360**	.199 → **.301**
4. SG-DEPS-PW (300) (Levy and Goldberg, 2014)	.376 → **.434**	.313 → **.418**
5. SG-DEPS-8B (500) (Bansal et al., 2014)	.373 → **.441**	.356 → **.473**
6. MultiCCA-EN (512) (Faruqui and Dyer, 2014)	.314 → **.391**	.296 → **.354**
7. BiSkip-EN (256) (Luong et al., 2015)	.276 → **.356**	.260 → **.333**
8. SG-BOW2-8B (500) (Schwartz et al., 2015)	.373 → **.440**	.348 → **.441**
9. SymPat-Emb (500) (Schwartz et al., 2016)	.381 → **.442**	.284 → **.373**
10. Context2Vec (600) (Melamud et al., 2016)	.371 → **.440**	.388 → **.459**

Table 4: The impact of morph-fitting (MFIT-AR used) on a representative set of EN vector space models. All results show the Spearman's ρ correlation before and after morph-fitting. The numbers in parentheses refer to the vector dimensionality.

Vectors	Distrib.	MFIT-A	MFIT-AR
EN: GloVe-6B (300)	.324	.376	**.438**
EN: SG-BOW2-PW (300)	.339	.385	**.439**
DE: SG-DEPS-PW (300) (Vulić and Korhonen, 2016a)	.267	.318	**.325**
DE: BiSkip-DE (256) (Luong et al., 2015)	.354	.414	**.421**
IT: SG-DEPS-PW (300) (Vulić and Korhonen, 2016a)	.237	.351	**.391**
IT: CBOW5-Wacky (300) (Dinu et al., 2015)	.363	.417	**.446**

Table 5: Results on multilingual SimLex-999 (EN, DE, and IT) with two morph-fitting variants.

strates that the morph-fitted vector spaces consistently outperform the morph-fixed ones.

The comparison between MFIT-A and MFIT-AR indicates that both sets of constraints are important for the fine-tuning process. MFIT-A yields consistent gains over the initial spaces, and (consistent) further improvements are achieved by also incorporating the antonymous REPEL constraints. This demonstrates that both types of constraints are useful for semantic specialisation.

Comparison to Other Specialisation Methods We also tried using other post-processing specialisation models from the literature in lieu of ATTRACT-REPEL using the same set of "morphological" synonymy and antonymy constraints. We compare ATTRACT-REPEL to the retrofitting model

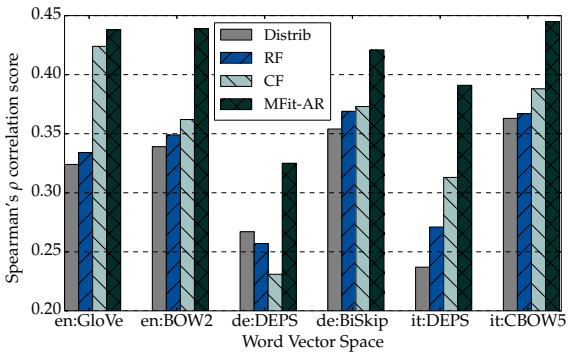


Figure 2: A comparison of morph-fitting (the MFIT-AR variant) with two other standard specialisation approaches using the same set of morphological constraints: Retrofitting (RF) (Faruqui et al., 2015) and Counter-fitting (CF) (Mrkšić et al., 2016). Spearman's ρ correlation scores on the multilingual SimLex-999 dataset for the same six distributional spaces from Tab. 5.

of (Faruqui et al., 2015) and counter-fitting (Mrkšić et al., 2017a). The two baselines were trained for 20 iterations using suggested settings. The results for EN, DE, and IT are summarised in Fig. 2. They clearly indicate that MFIT-AR outperforms the two other post-processors for each language. We hypothesise that the difference in performance mainly stems from context-sensitive vector space updates performed by ATTRACT-REPEL. Conversely, the other two models perform pairwise updates which do not consider what effect each update has on the example pair's relation to other word vectors (for a detailed comparison, see (Mrkšić et al., 2017b)).

Besides their lower performance, the two other specialisation models have additional disadvantages compared to the proposed morph-fitting model. First, retrofitting is able to incorporate only synonymy/ATTRACT pairs, while our results demonstrate the usefulness of both types of constraints, both for intrinsic evaluation (Tab. 5) and downstream tasks (see later Fig. 3). Second, counter-fitting is computationally intractable with SGNS-LARGE vectors, as its regularisation term involves the computation of all pairwise distances between words in the vocabulary.

Further Discussion The simplicity of the used language-specific rules does come at a cost of occasionally generating incorrect linguistic constraints such as *(tent, intent)*, *(prove, improve)* or *(press, impress)*. In future work, we will study how to fur-

ther refine extracted sets of constraints. We also plan to conduct experiments with gold standard morphological lexicons on languages for which such resources exist (Sylak-Glassman et al., 2015; Cotterell et al., 2016b), and investigate approaches which learn morphological inflections and derivations in different languages automatically as another potential source of morphological constraints (Soricut and Och, 2015; Cotterell et al., 2016a; Faruqui et al., 2016; Kann et al., 2017; Aharoni and Goldberg, 2017, i.a.).

5 Downstream Task: Dialogue State Tracking (DST)

Goal-oriented dialogue systems provide conversational interfaces for tasks such as booking flights or finding restaurants. In *slot-based* systems, application domains are specified using *ontologies* that define the search constraints which users can express. An ontology consists of a number of *slots* and their assorted *slot values*. In a *restaurant search* domain, sets of slot-values could include PRICE = [*cheap, expensive*] or FOOD = [*Thai, Indian, ...*].

The DST model is the first component of modern dialogue pipelines (Young, 2010). It serves to capture the intents expressed by the user at each dialogue turn and update the *belief state*. This probability distribution over the possible dialogue states (defined by the domain ontology) is the system's internal estimate of the user's goals. It is used by the downstream *dialogue manager* component to choose the subsequent system response (Su et al., 2016). The following example shows the true dialogue state in a multi-turn dialogue:

User: What's good in the southern part of town?
`inform(area=south)`
System: Vedanta is the top-rated Indian place.
User: How about something cheaper?
`inform(area=south, price=cheap)`
System: Seven Days is very popular. Great hot pot.
User: What's the address?
`inform(area=south, price=cheap); request(address)`
System: Seven Days is at 66 Regent Street.

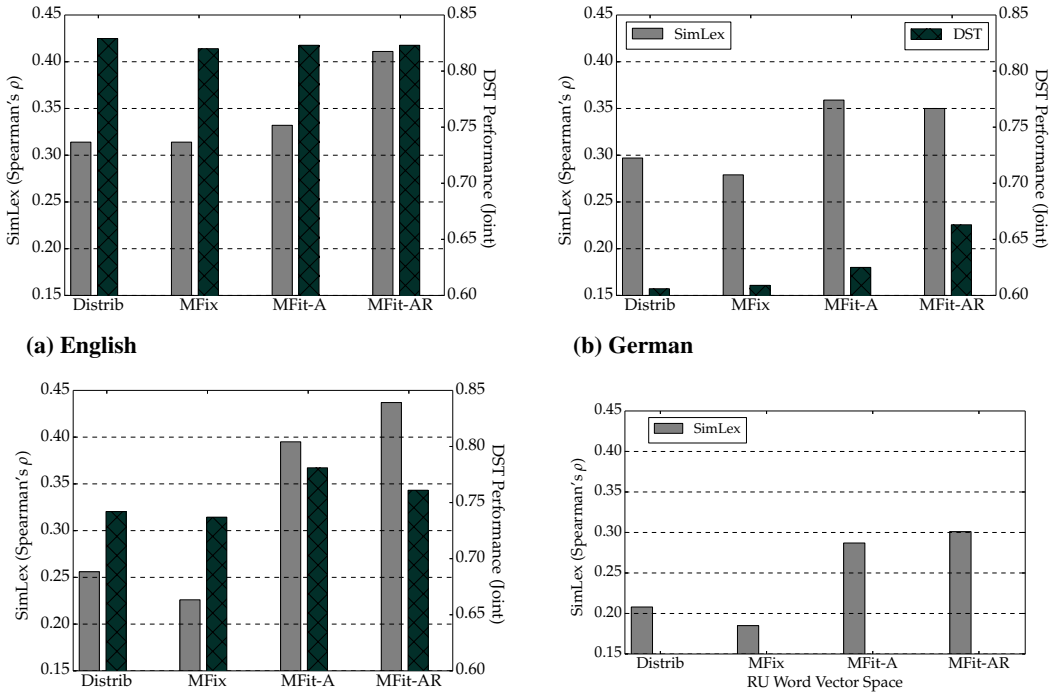
The Dialogue State Tracking Challenge (DSTC) shared task series formalised the evaluation and provided labelled DST datasets (Henderson et al., 2014a,b; Williams et al., 2016). While a plethora of DST models are available based on, e.g., handcrafted rules (Wang et al., 2014) or conditional random fields (Lee and Eskenazi, 2013), the recent DST methodology has seen a shift towards neural-network architectures (Henderson et al., 2014c,d; Zilka and Jurcicek, 2015; Mrkšić et al., 2015; Perez and Liu, 2017; Liu and Perez, 2017; Vodolán et al., 2017; Mrkšić et al., 2017a, i.a.).

Model: Neural Belief Tracker To detect intents in user utterances, most existing models rely on either (or both): **1)** Spoken Language Understanding models which require large amounts of annotated training data; or **2)** hand-crafted, domain-specific lexicons which try to capture lexical and morphological variation. The Neural Belief Tracker (NBT) is a novel DST model which overcomes both issues by reasoning purely over pre-trained word vectors (Mrkšić et al., 2017a). The NBT learns to compose these vectors into intermediate utterance and context representations. These are then used to decide which of the ontology-defined intents (goals) have been expressed by the user. The NBT model keeps word vectors *fixed* during training, so that unseen, yet related words can be mapped to the right intent at test time (e.g. *northern* to *north*).

Data: Multilingual WOZ 2.0 Dataset Our DST evaluation is based on the WOZ dataset, released by Wen et al. (2017). In this Wizard-of-Oz setup, two Amazon Mechanical Turk workers assumed the role of the user and the system asking/providing information about restaurants in Cambridge (operating over the same ontology and database used for DSTC2 (Henderson et al., 2014a)). Users typed instead of speaking, removing the need to deal with noisy speech recognition. In DSTC datasets, users would quickly adapt to the system's inability to deal with complex queries. Conversely, the WOZ setup allowed them to use sophisticated language. The WOZ 2.0 release expanded the dataset to 1,200 dialogues (Mrkšić et al., 2017a). In this work, we use translations of this dataset to Italian and German, released by Mrkšić et al. (2017b).

Evaluation Setup The principal metric we use to measure DST performance is the *joint goal accuracy*, which represents the proportion of test set dialogue turns where all user goals expressed up to that point of the dialogue were decoded correctly (Henderson et al., 2014a). The NBT models for EN, DE and IT are trained using four variants of the SGNS-LARGE vectors: **1)** the initial distributional vectors; **2)** *morph-fixed* vectors; **3)** and **4)** the two variants of *morph-fitted* vectors (see Sect. 3).

As shown by Mrkšić et al. (2017b), *semantic specialisation* of the employed word vectors ben-



(a) English **(b) German**

(c) Italian **(d) Russian**

Figure 3: An overview of the results (Spearman's ρ correlation) for four languages on SimLex-999 (grey bars, left y axis) and the downstream DST performance (dark bars, right y axis) using SGNS-LARGE vectors ($d = 300$), see Tab. 3 and Sect. 3. The left y axis measures the intrinsic word similarity performance, while the right y axis provides the scale for the DST performance (there are no DST datasets for Russian).

efits DST performance across all three languages. However, large gains on SimLex-999 do not always induce correspondingly large gains in downstream performance. In our experiments, we investigate the extent to which *morph-fitting* improves DST performance, and whether these gains exhibit stronger correlation with intrinsic performance.

Results and Discussion The dark bars (against the right axes) in Fig. 3 show the DST performance of NBT models making use of the four vector collections. IT and DE benefit from both kinds of *morph-fitting*: IT performance increases from $74.1 \rightarrow 78.1$ (MFIT-A) and DE performance rises even more: $60.6 \rightarrow 66.3$ (MFIT-AR), setting a new state-of-the-art score for both datasets. The *morph-fixed* vectors do not enhance DST performance, probably because fixing word vectors to their highest frequency inflectional form eliminates useful semantic content encoded in the original vectors. On the other hand, morph-fitting makes use of this information, supplementing it with semantic relations between different morphological forms. These conclusions are in line with the SimLex gains, where morph-fitting outperforms both distributional and *morph-fixed* vectors.

English performance shows little variation across the four word vector collections investigated here. This corroborates our intuition that, as a morphologically simpler language, English stands to gain less from fine-tuning the morphological variation for downstream applications. This result again points at the discrepancy between intrinsic and extrinsic evaluation: the considerable gains in SimLex performance do not necessarily induce similar gains in downstream performance. Additional discrepancies between SimLex and downstream DST performance are detected for German and Italian. While we observe a slight drop in SimLex performance with the DE MFIT-AR vectors compared to the MFIT-A ones, their relative performance is reversed in the DST task. On the other hand, we see the opposite trend in Italian, where the MFIT-A vectors score lower than the MFIT-AR vectors on SimLex, but higher on the DST task. In summary, we believe these results show that SimLex is not a perfect proxy for downstream performance in language understanding tasks. Regardless, its performance does correlate with downstream performance to a large extent, providing a useful indicator for the usefulness of specific word vector

spaces for extrinsic tasks such as DST.

6 Related Work

Semantic Specialisation A standard approach to incorporating external information into vector spaces is to pull the representations of similar words closer together. Some models integrate such constraints into the training procedure, modifying the prior or the regularisation (Yu and Dredze, 2014; Xu et al., 2014; Bian et al., 2014; Kiela et al., 2015), or using a variant of the SGNS-style objective (Liu et al., 2015; Osborne et al., 2016). Another class of models, popularly termed *retrofitting*, injects lexical knowledge from available semantic databases (e.g., WordNet, PPDB) into pre-trained word vectors (Faruqui et al., 2015; Jauhar et al., 2015; Wieting et al., 2015; Nguyen et al., 2016; Mrkšić et al., 2016). Morph-fitting falls into the latter category. However, instead of resorting to curated knowledge bases, and experimenting solely with English, we show that the *morphological richness* of any language can be exploited as a source of inexpensive supervision for fine-tuning vector spaces, at the same time specialising them to better reflect true semantic similarity, and learning more accurate representations for low-frequency words.

Word Vectors and Morphology The use of morphological resources to improve the representations of morphemes and words is an active area of research. The majority of proposed architectures encode morphological information, provided either as gold standard morphological resources (Sylak-Glassman et al., 2015) such as CELEX (Baayen et al., 1995) or as an external analyser such as Morfessor (Creutz and Lagus, 2007), along with distributional information jointly at *training* time in the language modelling (LM) objective (Luong et al., 2013; Botha and Blunsom, 2014; Qiu et al., 2014; Cotterell and Schütze, 2015; Bhatia et al., 2016, i.a.). The key idea is to learn a morphological composition function (Lazaridou et al., 2013; Cotterell and Schütze, 2017) which synthesises the representation of a word given the representations of its constituent morphemes. Contrary to our work, these models typically coalesce all lexical relations.

Another class of models, operating at the character level, shares a similar methodology: such models compose token-level representations from subcomponent embeddings (subwords, morphemes, or characters) (dos Santos and Zadrozny, 2014; Ling et al., 2015; Cao and Rei, 2016; Kim et al., 2016; Wieting et al., 2016; Verwimp et al., 2017, i.a.).

In contrast to prior work, our model *decouples* the use of morphological information, now provided in the form of inflectional and derivational rules transformed into constraints, from the actual training. This pipelined approach results in a simpler, more portable model. In spirit, our work is similar to Cotterell et al. (2016b), who formulate the idea of post-training specialisation in a generative Bayesian framework. Their work uses gold morphological lexicons; we show that competitive performance can be achieved using a non-exhaustive set of simple rules. Our framework facilitates the inclusion of *antonyms* at no extra cost and naturally extends to constraints from other sources (e.g., WordNet) in future work. Another practical difference is that we focus on similarity and evaluate morph-fitting in a well-defined downstream task where the artefacts of the distributional hypothesis are known to prompt statistical system failures.

7 Conclusion and Future Work

We have presented a novel *morph-fitting* method which injects morphological knowledge in the form of linguistic constraints into word vector spaces. The method makes use of implicit semantic signals encoded in inflectional and derivational rules which describe the morphological processes in a language. The results in intrinsic word similarity tasks show that *morph-fitting* improves vector spaces induced by distributional models across four languages. Finally, we have shown that the use of *morph-fitted* vectors boosts the performance of downstream language understanding models which rely on word representations as features, especially for morphologically rich languages such as German.

Future work will focus on other potential sources of morphological knowledge, porting the framework to other morphologically rich languages and downstream tasks, and on further refinements of the post-processing specialisation algorithm and the constraint selection.

Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909). RR is supported by the Intel-ICRI grant: Hybrid Models for Minimally Supervised Information Extraction from Conversations. The authors are grateful to the anonymous reviewers for their helpful suggestions.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of ACL*. https://arxiv.org/abs/1611.01487.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*. pages 183–192. http://www.aclweb.org/anthology/W13-3520.

Eleftherios Avramidis and Philipp Koehn. 2008. Enriching morphologically poor languages for statistical machine translation. In *Proceedings of ACL*. pages 763–770. http://www.aclweb.org/anthology/P/P08/P08-1087.

Harald R. Baayen, Richard Piepenbrock, and Hedderik van Rijn. 1995. The CELEX lexical data base on CD-ROM .

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of ACL*. pages 809–815. http://www.aclweb.org/anthology/P14-2131.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*. pages 238–247. http://www.aclweb.org/anthology/P14-1023.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of EMNLP*. pages 490–500. https://aclweb.org/anthology/D16-1047.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Proceedings of ECML-PKDD*. pages 132–148. https://doi.org/10.1007/978-3-662-44848-9_9.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of ICML*. pages 1899–1907. http://jmlr.org/proceedings/papers/v32/botha14.html.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research* 49:1–47. https://doi.org/10.1613/jair.4135.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. pages 18–26. http://aclweb.org/anthology/W/W16/W16-1603.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*. pages 740–750. http://www.aclweb.org/anthology/D14-1082.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537. http://dl.acm.org/citation.cfm?id=1953048.2078186.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016a. The sigmorphon 2016 shared task - morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. pages 10–22. http://anthology.aclweb.org/W16-2002.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of NAACL-HLT*. pages 1287–1292. http://www.aclweb.org/anthology/N15-1140.

Ryan Cotterell and Hinrich Schütze. 2017. Joint semantic synthesis and morphological analysis of the derived word. *Transactions of the ACL* https://arxiv.org/abs/1701.00946.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016b. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of ACL*. pages 1651–1660. http://www.aclweb.org/anthology/P16-1156.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *TSLP* 4(1):3:1–3:34. http://doi.acm.org/10.1145/1217098.1217101.

James Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, School of Informatics, University of Edinburgh. http://hdl.handle.net/1842/563.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of ICLR (Workshop Papers)*. http://arxiv.org/abs/1412.6568.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of ICML*. pages 1818–1826. http://jmlr.org/proceedings/papers/v32/santos14.html.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159. http://dl.acm.org/citation.cfm?id=2021068.

Maud Ehrmann, Francesco Cecconi, Daniele Vannella, John Philip Mccrae, Philipp Cimiano, and Roberto Navigli. 2014. Representing multilingual data as linked data: The case of BabelNet 2.0. In *Proceedings of LREC*. pages 401–408. http://www.lrec-conf.org/proceedings/lrec2014/summaries/810.html.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*. pages 1606–1615. http://www.aclweb.org/anthology/N15-1184.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*. pages 462–471. http://www.aclweb.org/anthology/E14-1049.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of NAACL-HLT*. pages 634–643. http://www.aclweb.org/anthology/N16-1077.

Christiane Fellbaum. 1998. *WordNet*. https://mitpress.mit.edu/books/wordnet.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems* 20(1):116–131. https://doi.org/10.1145/503104.503110.

Victoria Fromkin, Robert Rodman, and Nina Hyams. 2013. *An Introduction to Language, 10th Edition*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*. pages 758–764. http://www.aclweb.org/anthology/N13-1092.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of EMNLP*. pages 2173–2182. https://aclweb.org/anthology/D16-1235.

Zellig S. Harris. 1954. Distributional structure. *Word* 10(23):146–162.

Martin Haspelmath and Andrea Sims. 2013. *Understanding morphology*.

Matthew Henderson, Blaise Thomson, and Jason D. Wiliams. 2014a. The Second Dialog State Tracking Challenge. In *Proceedings of SIGDIAL*. pages 263–272. http://aclweb.org/anthology/W/W14/W14-4337.pdf.

Matthew Henderson, Blaise Thomson, and Jason D. Wiliams. 2014b. The Third Dialog State Tracking Challenge. In *Proceedings of IEEE SLT*. pages 324–329. https://doi.org/10.1109/SLT.2014.7078595.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014c. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Proceedings of IEEE SLT*. pages 360–365.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014d. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of SIGDIAL*. pages 292–299. http://aclweb.org/anthology/W/W14/W14-4340.pdf.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* 41(4):665–695. https://doi.org/10.1162/COLI_a_00237.

Sujay Kumar Jauhar, Chris Dyer, and Eduard H. Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of NAACL*. pages 683–693. http://www.aclweb.org/anthology/N15-1070.

Anders Johannsen, Héctor Martínez Alonso, and Anders Søgaard. 2015. Any-language frame-semantic parsing. In *Proceedings of EMNLP*. pages 2062–2066. http://aclweb.org/anthology/D15-1245.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. Neural multi-source morphological reinflection. In *Proceedings of EACL*. pages 514–524. http://www.aclweb.org/anthology/E17-1049.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of EMNLP*. pages 2044–2048. http://aclweb.org/anthology/D15-1242.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI*. pages 2741–2749.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositionally derived representations of morphologically complex words in distributional semantics. In *Proceedings of ACL*. pages 1517–1526. http://www.aclweb.org/anthology/P13-1149.

Sungjin Lee and Maxine Eskenazi. 2013. Recipe for building robust spoken dialog state trackers: Dialog State Tracking Challenge system description. In *Proceedings of SIGDIAL*. pages 414–422. http://aclweb.org/anthology/W/W13/W13-4066.pdf.

Ira Leviant and Roi Reichart. 2015. Separated by an un-common language: Towards judgment language informed vector space modeling. *CoRR* abs/1508.00106. http://arxiv.org/abs/1508.00106.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*. pages 302–308. http://www.aclweb.org/anthology/P14-2050.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form:

Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP*. pages 1520–1530. http://aclweb.org/anthology/D15-1176.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of EACL*. pages 1–10. http://www.aclweb.org/anthology/E17-1001.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of ACL*. pages 1501–1511. http://www.aclweb.org/anthology/P15-1145.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. pages 151–159. http://www.aclweb.org/anthology/W15-1521.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of CoNLL*. pages 104–113. http://www.aclweb.org/anthology/W13-3512.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. Context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of CoNLL*. pages 51–61. http://aclweb.org/anthology/K/K16/K16-1006.pdf.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*. pages 3111–3119.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of NIPS*. pages 2265–2273.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of ACL*. pages 794–799. http://aclweb.org/anthology/P/P15/P15-2130.pdf.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Tsung-Hsien Wen, and Steve Young. 2017a. Neural Belief Tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*. http://arxiv.org/abs/1606.03777.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT*. http://aclweb.org/anthology/N/N16/N16-1018.pdf.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017b. Semantic Specialisation of Distributional Word Vector Spaces using Monolingual and Cross-Lingual Constraints. arXiv.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*. pages 807–814. http://www.icml2010.org/papers/432.pdf.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193:217–250. https://doi.org/10.1016/j.artint.2012.07.001.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of ACL*. pages 454–459. http://anthology.aclweb.org/P16-2074.

Dominique Osborne, Shashi Narayan, and Shay Cohen. 2016. Encoding prior knowledge with eigenword embeddings. *Transactions of the ACL* 4:417–430. https://arxiv.org/abs/1509.01007.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of ACL*. pages 425–430. http://www.aclweb.org/anthology/P15-2070.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using Memory Network. In *Proceedings of EACL*. pages 305–314. http://www.aclweb.org/anthology/E17-1029.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING*. pages 141–150. http://www.aclweb.org/anthology/C14-1015.

Patrick Schone and Daniel Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proceedings of NAACL*. http://aclweb.org/anthology/N/N01/N01-1024.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of CoNLL*. pages 258–267. http://www.aclweb.org/anthology/K15-1026.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2016. Symmetric patterns and coordinations: Fast and enhanced representations of verbs and adjectives.

In *Proceedings of NAACL-HLT*. pages 499–505. http://www.aclweb.org/anthology/N16-1060.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of NAACL-HLT*. pages 1627–1637. http://www.aclweb.org/anthology/N15-1186.

Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. ???? Continuously learning neural dialogue management.

Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of ACL*. pages 2431–2441. http://www.aclweb.org/anthology/P16-1230.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *Proceedings of ACL*. pages 674–680. http://www.aclweb.org/anthology/P15-2111.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kuebler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (SPMRL) What, how and whither. In *Proceedings of the NAACL Workshop on Statistical Parsing of Morphologically-Rich Languages*. pages 1–12. http://www.aclweb.org/anthology/W10-1401.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*. pages 384–394. http://www.aclweb.org/anthology/P10-1040.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: vector space models of semantics. *Journal of Artifical Intelligence Research* 37(1):141–188. https://doi.org/10.1613/jair.2934.

Lyan Verwimp, Joris Pelemans, Hugo Van hamme, and Patrick Wambacq. 2017. Character-word LSTM language models. In *Proceedings of EACL*. pages 417–427. http://www.aclweb.org/anthology/E17-1040.

Miroslav Vodolán, Rudolf Kadlec, and Jan Kleindienst. 2017. Hybrid dialog state tracker with ASR features. In *Proceedings of EACL*. pages 205–210. http://www.aclweb.org/anthology/E17-2033.

Ivan Vulić and Anna Korhonen. 2016a. Is "universal syntax" universally useful for learning distributed word representations? In *Proceedings of ACL*. pages 518–524. http://anthology.aclweb.org/P16-2084.

Ivan Vulić and Anna Korhonen. 2016b. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of ACL*. pages 247–257. http://www.aclweb.org/anthology/P16-1024.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI*. pages 1112–1119.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*. http://www.aclweb.org/anthology/E17-1042.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL* 3:345–358.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of EMNLP*. pages 1504–1515. https://aclweb.org/anthology/D16-1157.

Jason D. Williams, Antoine Raux, and Matthew Henderson. 2016. The Dialog State Tracking Challenge series: A review. *Dialogue & Discourse* 7(3):4–33. http://dad.uni-bielefeld.de/index.php/dad/article/view/3685.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of CIKM*. pages 1219–1228. https://doi.org/10.1145/2661829.2662038.

Steve Young. 2010. Cognitive User Interfaces. *IEEE Signal Processing Magazine* .

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of ACL*. pages 545–550. http://www.aclweb.org/anthology/P14-2089.

Britta Zeller, Jan Šnajder, and Sebastian Padó. 2013. DErivBase: Inducing and evaluating a derivational morphology resource for German. In *Proceedings of ACL*. pages 1201–1211. http://www.aclweb.org/anthology/P13-1118.

Lukas Zilka and Filip Jurcicek. 2015. Incremental LSTM-based dialog state tracker. In *Proceedings of ASRU*.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*. pages 1393–1398. http://www.aclweb.org/anthology/D13-1141.

Skip-Gram – Zipf + Uniform = Vector Additivity

Alex Gittens
Dept. of Computer Science
Rensselaer Polytechnic Institute
gittea@rpi.edu

Dimitris Achlioptas
Dept. of Computer Science
UC Santa Cruz
optas@soe.ucsc.edu

Michael W. Mahoney
ICSI and Dept. of Statistics
UC Berkeley
mmahoney@stat.berkeley.edu

Abstract

In recent years word-embedding models have gained great popularity due to their remarkable performance on several tasks, including word analogy questions and caption generation. An unexpected "side-effect" of such models is that their vectors often exhibit compositionality, i.e., *adding* two word-vectors results in a vector that is only a small angle away from the vector of a word representing the semantic composite of the original words, e.g., "man" + "royal" = "king".

This work provides a theoretical justification for the presence of additive compositionality in word vectors learned using the Skip-Gram model. In particular, it shows that additive compositionality holds in an even stricter sense (small distance rather than small angle) under certain assumptions on the process generating the corpus. As a corollary, it explains the success of vector calculus in solving word analogies. When these assumptions do not hold, this work describes the correct non-linear composition operator.

Finally, this work establishes a connection between the Skip-Gram model and the Sufficient Dimensionality Reduction (SDR) framework of Globerson and Tishby: the parameters of SDR models can be obtained from those of Skip-Gram models simply by adding information on symbol frequencies. This shows that Skip-Gram embeddings are optimal in the sense of Globerson and Tishby and, further, implies that the heuristics commonly used to approximately fit Skip-Gram models can be used to fit SDR models.

1 Introduction

The strategy of representing words as vectors has a long history in computational linguistics and machine learning. The general idea is to find a map from words to vectors such that word-similarity and vector-similarity are in correspondence. Whilst vector-similarity can be readily quantified in terms of distances and angles, quantifying word-similarity is a more ambiguous task. A key insight in that regard is to posit that the meaning of a word is captured by "the company it keeps" (Firth, 1957) and, therefore, that two words that keep company with similar words are likely to be similar themselves.

In the simplest case, one seeks vectors whose inner products approximate the co-occurrence frequencies. In more sophisticated methods co-occurrences are reweighed to suppress the effect of more frequent words (Rohde et al., 2006) and/or to emphasize pairs of words whose co-occurrence frequency maximally deviates from the independence assumption (Church and Hanks, 1990).

An alternative to seeking word-embeddings that reflect co-occurrence statistics is to extract the vectorial representation of words from non-linear statistical language models, specifically neural networks. (Bengio et al., 2003) already proposed (i) associating with each vocabulary word a *feature vector*, (ii) expressing the *probability function* of word sequences in terms of the feature vectors of the words in the sequence, and (iii) learning *simultaneously* the vectors and the parameters of the probability function. This approach came into prominence recently through works of Mikolov et al. (see below) whose main departure from (Bengio et al., 2003) was to follow the suggestion of (Mnih and Hinton, 2007) and trade-away the expressive capacity of general neural-network models for the scalability (to very large

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 69–76
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1007

corpora) afforded by (the more restricted class of) log-linear models.

An unexpected side effect of deriving word-embeddings via neural networks is that the word-vectors produced appear to enjoy (approximate) *additive compositionality*: adding two word-vectors often results in a vector whose nearest word-vector belongs to the word capturing the composition of the added words, e.g., "man" + "royal" = "king" (Mikolov et al., 2013c). This unexpected property allows one to use these vectors to answer word-analogy questions *algebraically*, e.g., answering the question "Man is to king as woman is to ___" by returning the word whose word-vector is nearest to the vector

$$\mathbf{v}(\text{king}) - \mathbf{v}(\text{man}) + \mathbf{v}(\text{woman}).$$

In this work we focus on explaining the source of this phenomenon for the most prominent such model, namely the Skip-Gram model introduced in (Mikolov et al., 2013a). The Skip-Gram model learns vector representations of words based on their patterns of co-occurrence in the training corpus as follows: it assigns to each word c in the vocabulary V, a "context" and a "target" vector, respectively $\mathbf{u}_c$ and $\mathbf{v}_c$, which are to be used in order to predict the words that appear around each occurrence of c within a window of Δ tokens. Specifically, the log probability of any target word w to occur at any position within distance Δ of a context word c is taken to be proportional to the inner product between u_c and v_w, i.e., letting $n = |V|$,

$$p(w|c) = \frac{\mathrm{e}^{\mathbf{u}_c^T \mathbf{v}_w}}{\sum_{i=1}^{n} \mathrm{e}^{\mathbf{u}_c^T \mathbf{v}_i}} \quad . \tag{1}$$

Further, Skip-Gram assumes that the conditional probability of each possible set of words in a window around a context word c factorizes as the product of the respective conditional probabilities:

$$p(w_{-\Delta}, \ldots, w_{\Delta}|c) = \prod_{\substack{\delta=-\Delta \\ \delta \neq 0}}^{\Delta} p(w_\delta|c). \tag{2}$$

(Mikolov et al., 2013a) proposed learning the Skip-Gram parameters on a training corpus by using maximum likelihood estimation under (1) and (2). Thus, if w_i denotes the i-th word in the training corpus and T the length of the corpus, we seek the word vectors that maximize

$$\frac{1}{T} \sum_{i=1}^{T} \sum_{\substack{\delta=-\Delta \\ \delta \neq 0}}^{\Delta} \log p(w_{i+\delta}|w_i) \quad . \tag{3}$$

As mentioned, the normalized context vectors obtained from maximizing (3) under (1) and (2) exhibit additive compositionality. For example, the cosine distance between the sum of the context vectors of the words "Vietnam" and "capital" and the context vector of the word "Hanoi" is small.

While there has been much interest in using algebraic operations on word vectors to carry out semantic operations like composition, and mathematically-flavored explanations have been offered (e.g., in the recent work (Paperno and Baroni, 2016)), the only published work which attempts a rigorous theoretical understanding of this phenomenon is (Arora et al., 2016). This work guarantees that word vectors can be recovered by factorizing the so-called PMI matrix, and that algebraic operations on these word vectors can be used to solve analogies, under certain conditions on the process that generated the training corpus. Specifically, the word vectors must be known *a priori, before their recovery, and to have been generated by randomly scaling uniformly sampled vectors from the unit sphere*[1]. Further, the ith word in the corpus must have been selected with probability proportional to $\mathrm{e}^{\mathbf{u}_w^T \mathbf{c}_i}$, where the "discourse" vector $\mathbf{c}_i$ governs the topic of the corpus at the ith word. Finally, the discourse vector is assumed to evolve according to a random walk on the unit sphere that has a uniform stationary distribution.

By way of contrast, our results assume nothing *a priori* about the properties of the word vectors. In fact, the connection we establish between the Skip-Gram and the Sufficient Dimensionality Reduction model of (Globerson and Tishby, 2003) shows that the word vectors learned by Skip-Gram are information-theoretically optimal. Further, the context word c in the Skip-Gram model essentially serves the role that the discourse vector does in the PMI model of (Arora et al., 2016): the words neighboring c are selected with probability proportional to $\mathrm{e}^{\mathbf{u}_c^T \mathbf{v}_w}$. We find the exact non-linear composition operator when no assumptions are made on the context word. When an analogous assumption to that of (Arora et al., 2016) is made, that the

[1] More generally, it suffices that the word vectors have certain properties consistent with this sampling process.

context words are uniformly distributed, we prove that the composition operator reduces to vector addition.

While our primary motivation has been to provide a better theoretical understanding of word compositionality in the popular Skip-Gram model, our connection with the SDR method illuminates a much more general point about the practical applicability of the Skip-Gram model. In particular, it addresses the question of whether, for a given corpus, fitting a Skip-Gram model will give good embeddings. Even if we are making reasonable linguistic assumptions about how to model words and the interdependencies of words in a corpus, it's not clear that these have to hold universally on all corpuses to which we apply Skip-Gram. However, the fact that when we fit a Skip-Gram model we are fitting an SDR model (up to frequency information), and the fact that SDR models are information-theoretically optimal in a certain sense, argues that regardless of whether the Skip-Gram assumptions hold, Skip-Gram always gives us optimal features in the following sense: the learned context embeddings and target embeddings preserve the maximal amount of mutual information between any pair of random variables X and Y consistent with the observed co-occurence matrix, where Y is the target word and X is the predictor word (in a min-max sense, since there are many ways of coupling X and Y, each of which may have different amounts of mutual information). Importantly, this statement requires no assumptions on the distribution $P(X, Y)$.

2 Compositionality of Skip-Gram

In this section, we first give a mathematical formulation of the intuitive notion of compositionality of words. We then prove that the composition operator for the Skip-Gram model in full generality is a non-linear function of the vectors of the words being composed. Under a single simplifying assumption, the operator *linearizes* and reduces to the addition of the word vectors. Finally, we explain how linear compositionality allows for solving word analogies with vector algebra.

A natural way of capturing the compositionality of words is to say that the *set* of context words $c_1, \ldots, c_m$ has the same meaning as the single word c if for every other word w,

$$p(w|c_1, \ldots, c_m) = p(w|c) \ .$$

Although this is an intuitively satisfying definition, we never expect it to hold exactly; instead, we replace exact equality with the minimization of KL-divergence. That is, we state that the best candidate for having the same meaning as the set of context words C is the word

$$\arg\min_{c \in V} \mathrm{D}_{\mathrm{KL}}(p(\cdot|C) \,|\, p(\cdot|c)) \ . \tag{4}$$

We refer to any vector that minimizes (4) as a *paraphrase* of the set of words C.

There are two natural concerns with (4). The first is that, in general, it is not clear how to define $p(\cdot|C)$. The second is that KL-divergence minimization is a hard problem, as it involves optimization over many high dimensional probability distributions. Our main result shows that both of these problems go away for any language model that satisfies the following two assumptions:

A1. For every word c, there exists Z_c such that for every word w,

$$p(w|c) = \frac{1}{Z_c} \exp(\mathbf{u}_c^T \mathbf{v}_w) \ . \tag{5}$$

A2. For every set of words $C = \{c_1, c_2, \ldots, c_m\}$, there exists Z_C such that for every word w,

$$p(w|C) = \frac{p(w)^{1-m}}{Z_C} \prod_{i=1}^{m} p(w|c_i) \ . \tag{6}$$

Clearly, the Skip-Gram model satisfies A1 by definition. We prove that it also satisfies A2 when $m \leq \Delta$ (Lemma 1). Next, we state a theorem that holds for any model satisfying assumptions A1 and A2, including the Skip-Gram model when $m \leq \Delta$.

Theorem 1. *In every word model that satisfies A1 and A2, for every set of words $C = \{c_1, \ldots, c_m\}$, any paraphase c of C satisfies*

$$\sum_{w \in V} p(w|c)\mathbf{v}_w = \sum_{w \in V} p(w|C)\mathbf{v}_w \ . \tag{7}$$

Theorem 1 characterizes the composition operator for any language model which satisfies our two assumptions; in general, this operator is *not* addition. Instead, a paraphrase c is a vector such that the average word vector under $p(\cdot|c)$ matches that under $p(\cdot|C)$. When the expectations in (7) can be computed, the composition operator can be implemented by solving a non-linear system of equations to find a vector $\mathbf{u}$ for which the left-hand side of (7) equals the right-hand side.

Our next result proves that although the composition operator is nontrivial in the general case, to recover vector addition as the composition operator, it suffices to assume that the word frequency is *uniform*.

Theorem 2. *In every word model that satisfies A1, A2, and where $p(w) = 1/|V|$ for every $w \in V$, the paraphrase of $C = \{c_1, \ldots, c_m\}$ is*

$$\mathbf{u}_1 + \ldots + \mathbf{u}_m \ .$$

As word frequencies are typically much closer to a Zipf distribution (Piantadosi, 2014), the uniformity assumption of Theorem 2 is not realistic. That said, we feel it is important to point out that, as reported in (Mikolov et al., 2013b), additivity captures compositionality *more* accurately when the training set is manipulated so that the prior distribution of the words is made *closer* to uniform.

Using composition to solve analogies. It has been observed that word vectors trained using nonlinear models like Skip-Gram tend to encode semantic relationships between words as linear relationships between the word vectors (Mikolov et al., 2013b; Pennington et al., 2014; Levy and Goldberg, 2014). In particular, analogies of the form "man:woman::king:?" can often be solved by taking ? to be the word in the vocabulary whose context vector has the smallest angle with $\mathbf{u}_{\text{woman}} + (\mathbf{u}_{\text{king}} - \mathbf{u}_{\text{man}})$. Theorems 1 and 2 offer insight into the solution such analogy questions.

We first consider solving an analogy of the form "m:w::k:?"" in the case where the composition operator is nonlinear. The fact that m and w share a relationship means m is a paraphrase of the set of words $\{w, R\}$, where R is a set of words encoding the relationship between m and w. Similarly, the fact that k and ? share the same relationship means k is a paraphrase of the set of words $\{?, R\}$. By Theorem 1, we have that R and ? must satisfy

$$\sum_{\ell \in V} p(\ell|m) v_\ell = \sum_{\ell \in V} p(\ell|w, R) v_\ell \quad \text{and}$$
$$\sum_{\ell \in V} p(\ell|k) v_\ell = \sum_{\ell \in V} p(\ell|?, R) v_\ell .$$

We see that solving analogies when the composition operator is nonlinear requires the solution of two highly nonlinear systems of equations. In sharp contrast, when the composition operator is linear, the solution of analogies delightfully reduces to elementary vector algebra. To see this, we again begin with the assertion that the fact that m and w share a relationship means m is a paraphrase of the set of words $\{w, R\}$; Similarly, k is a paraphrase of $\{?, R\}$. By Theorem 2,

$$\mathbf{u}_m = \mathbf{u}_w + \mathbf{u}_r \quad \text{and}$$
$$\mathbf{u}_k = \mathbf{u}_? + \mathbf{u}_r,$$

which gives the expected relationship

$$\mathbf{u}_? = \mathbf{u}_k + (\mathbf{u}_w - \mathbf{u}_m).$$

Note that because this expression for $\mathbf{u}_?$ is in terms of k, w, and m, there is actually no need to assume that R is a set of actual words in V.

2.1 Proofs

Proof of Theorem 1. Note that $p(w|C)$ equals

$$\frac{p(w)^{1-m}}{Z_C} \prod_{i=1}^{m} p(w|c_i)$$
$$= \frac{p(w)^{1-m}}{Z_C} \exp\left(\sum_{i=1}^{m} \mathbf{u}_{c_i}^T \mathbf{v}_w - \sum_{i=1}^{m} \log Z_{c_i} \right)$$
$$= \frac{1}{Z} p(w)^{1-m} \exp(\mathbf{u}_C^T \mathbf{v}_w) \ ,$$

where $Z = Z_C \prod_{i=1}^m Z_i$, and $\mathbf{u}_C = \sum_{i=1}^m \mathbf{u}_i$.

Minimizing the KL-divergence

$$\mathrm{D}_{\mathrm{KL}}(p(\cdot|c_1, \ldots, c_m) \| p(\cdot|c))$$

as a function of c is equivalent to maximizing the negative cross-entropy as a function of $\mathbf{u}_c$, i.e., as maximizing

$$Q(\mathbf{u}_c) = Z \sum_w \frac{\exp(\mathbf{u}_C^T \mathbf{v}_w)}{p(w)^{m-1}} (\mathbf{u}_c^T \mathbf{v}_w - \log Z_c) \ .$$

Since Q is concave, the maximizers occur where its gradient vanishes. As $\nabla_{\mathbf{u}_c} Q$ equals

$$Z \sum_w \frac{\exp(\mathbf{u}_C^T \mathbf{v}_w)}{p(w)^{m-1}} \left[\mathbf{v}_w - \frac{\sum_{\ell=1}^n \exp(\mathbf{u}_c^T \mathbf{v}_\ell) \mathbf{v}_\ell}{\sum_{k=1}^n \exp(\mathbf{u}_c^T \mathbf{v}_k)} \right]$$
$$= \frac{\sum_{\ell=1}^n \exp(\mathbf{u}_c^T \mathbf{v}_\ell) \mathbf{v}_\ell}{\sum_{k=1}^n \exp(\mathbf{u}_c^T \mathbf{v}_k)} - Z \sum_w \frac{\exp(\mathbf{u}_C^T \mathbf{v}_w) \mathbf{v}_w}{p(w)^{m-1}}$$
$$= \sum_{w \in V} p(w|c) \mathbf{v}_w - \sum_{w \in V} p(w|c_1, \ldots, c_m) \mathbf{v}_w \ ,$$

we see that (7) follows. □

Proof of Theorem 2. Recall that $\mathbf{u}_C = \sum_{i=1}^m \mathbf{u}_i$. When $p(w) = 1/|V|$ for all $w \in V$, the negative cross-entropy simplifies to

$$Q(\mathbf{u}_c) = Z \sum_w \exp\left(\mathbf{u}_C^T \mathbf{v}_w\right) \left(\mathbf{u}_c^T \mathbf{v}_w - \log Z_c\right) \ ,$$

and its gradient $\nabla_{\mathbf{u}_c} Q$ to

$$Z \sum_w \exp({\mathbf{u}_C}^T \mathbf{v}_w) \left[\mathbf{v}_w - \frac{\sum_{\ell=1}^n \exp(\mathbf{u}_c^T \mathbf{v}_\ell) \mathbf{v}_\ell}{\sum_{k=1}^n \exp(\mathbf{u}_c^T \mathbf{v}_k)} \right]$$
$$= Z \sum_w \exp({\mathbf{u}_C}^T \mathbf{v}_w) \mathbf{v}_w - \sum_w \exp(\mathbf{u}_c^T \mathbf{v}_w) \mathbf{v}_w \ .$$

Thus, $\nabla Q(\mathbf{u}_C) = 0$ and since Q is concave, $\mathbf{u}_C$ is its unique maximizer. □

Lemma 1. *The Skip-Gram model satisfies assumption A2 when $m \leq \Delta$.*

Proof of Lemma 1. First, assume that $m = \Delta$. In the Skip-Gram model target words are conditionally independent given a context word, i.e.,

$$p(c_1, \ldots, c_m | w) = \prod_{i=1}^m p(c_i | w).$$

Applying Baye's rule,

$$\begin{aligned} p(w | c_1, \ldots, c_m) &= \frac{p(c_1, \ldots, c_m | w) p(w)}{p(c_1, \ldots, c_m)} \\ &= \frac{p(w)}{p(c_1, \ldots, c_m)} \prod_{i=1}^m p(c_i | w) \\ &= \frac{p(w)}{p(c_1, \ldots, c_m)} \prod_{i=1}^m \frac{p(w | c_i) p(c_i)}{p(w)} \\ &= \frac{p(w)^{1-m}}{Z_C} \prod_{i=1}^m p(w | c_i) \ , \qquad (8) \end{aligned}$$

where $Z_C = 1/\left(\prod_{i=1}^m p(c_i)\right)$. This establishes the result when $m = \Delta$. The cases $m < \Delta$ follow by marginalizing out $\Delta - m$ context words in the equality (8). □

Projection of paraphrases onto the vocabulary Theorem 2 states that if there is a word c in the vocabulary V whose context vector equals the sum of the context vectors of the words $c_1, \ldots, c_m$, then c has the same "meaning", in the sense of (4), as the composition of the words $c_1, \ldots, c_m$. For any given set of words $C = \{c_1, \ldots, c_m\}$, it is unlikely that there exists a word $c \in V$ whose context vector is exactly equal to the sum of the context vectors of the words $c_1, \ldots, c_m$. Similarly, in Theorem 1, the solution(s) to (7) will most likely not equal the context vector of any word in V. In both cases, we thus need to project the vector(s) onto words in our vocabulary in some manner.

Since Theorem 1 holds for any prior over V, in theory, we could enumerate all words in V and find the word(s) that minimize the difference of the left hand side of (7) from the right hand side. In practice, it turns out that the *angle* between the context vector of a word $w \in V$ and solution-vector(s) is a good proxy and one gets very good experimental results by selecting as the paraphrase of a collection of words, the word that minimizes the angle to the paraphrase vector.

Minimizing the angle has been empirically successful at capturing composition in multiple log-linear word models. One way to understand the success of this approach is to recall that each word c is characterized by a categorical distribution over all other words w, as stated in (1). The peaks of this categorical distribution are precisely the words with which c co-occurs most often. These words characterize c more than all the other words in the vocabulary, so it is reasonable to expect that a word c' whose categorical distribution has similar peaks as the categorical distribution of c is similar in meaning to c. Note that the location of the peaks of $p(\cdot|c)$ are immune to the scaling of $\mathbf{u}_c$ (athough the values of $p(\cdot|c)$ may change); thus, the words w which best characterize c are those for which $\mathbf{v}_w$ has a high inner product with $\mathbf{u}_c / \|\mathbf{u}_c\|_2$. Since

$$\left| \frac{\mathbf{u}_c^T \mathbf{v}_w}{\|\mathbf{u}_c\|_2} - \frac{\mathbf{u}_{c'}^T \mathbf{v}_w}{\|\mathbf{u}_{c'}\|_2} \right| \leq \sqrt{2\left(1 - \frac{\mathbf{u}_c^T \mathbf{u}_{c'}}{\|\mathbf{u}_c\|_2 \|\mathbf{u}_{c'}\|_2}\right)} \|\mathbf{v}_w\|_2,$$

it is clear that if the angle between the context representations of c and c' is small, the distributions $p(w|c)$ and $p(w|c')$ will tend to have similar peaks.

3 Skip-Gram learns a Sufficient Dimensionality Reduction Model

The Skip-Gram model assumes that the distribution of the neighbors of a word follows a specific exponential parametrization of a categorical distribution. There is empirical evidence that this model generates features that are useful for NLP tasks, but there is no *a priori* guarantee that the training corpus was generated in this manner. In this section, we provide theoretical support for the usefulness of the features learned even when the Skip-Gram model is misspecified.

To do so, we draw a connection between Skip-Gram and the Sufficient Dimensionality Reduction (SDR) factorization of Globerson and Tishby (Globerson and Tishby, 2003). The SDR model

learns optimal[2] embeddings for discrete random variables X and Y *without assuming any parametric form on the distributions* of X and Y, and it is useful in a variety of applications, including information retrieval, document classification, and association analysis (Globerson and Tishby, 2003). As it turns out, these embeddings, like Skip-Gram, are obtained by learning the parameters of an exponentially parameterized distribution. In Theorem 3 below, we show that if a Skip-Gram model is fit to the cooccurence statistics of X and Y, then the output can be trivially modified (by adding readily-available information on word frequencies) to obtain the parameters of an SDR model.

This connection is significant for two reasons: first, the original algorithm of (Globerson and Tishby, 2003) for learning SDR embeddings is expensive, as it involves information projections. Theorem 3 shows that if one can efficiently fit a Skip-Gram model, then one can efficiently fit an SDR model. This implies that Skip-Gram specific approximation heuristics like negative-sampling, hierarchical softmax, and Glove, which are believed to return high-quality approximations to Skip-Gram parameters (Mikolov et al., 2013b; Pennington et al., 2014), can be used to efficiently approximate SDR model parameters. Second, (Globerson and Tishby, 2003) argues for the optimality of the SDR embedding in any domain where the training information on X and Y consists of their coocurrence statistics; this optimality and the Skip-Gram/SDR connection argues for the use of Skip-Gram approximations in such domains, and supports the positive experimental results that have been observed in applications in network science (Grover and Leskovec, 2016), proteinomics (Asgari and Mofrad, 2015), and other fields.

As stated above, the SDR factorization solves the problem of finding information-theoretically optimal features, given co-occurrence statistics for a pair of discrete random variables X and Y. Associate a vector $\mathbf{w}_i$ to the ith state of X, a vector $\mathbf{h}_j$ to the jth state of Y, and let $\mathbf{W} = [\mathbf{w}_1^T \cdots \mathbf{w}_{|X|}^T]^T$ and $\mathbf{H}$ be defined similarly. Globerson and Tishby show that such optimal features can be obtained from a low-rank factorization of the matrix $\mathbf{G}$ of co-occurence measurements: G_{ij} counts the number of times state i of X has been observed to co-occur with state j of Y. The loss of this factorization is measured using the KL-divergence, and so the optimal features are obtained from solving the problem

$$\arg\min_{\mathbf{W},\mathbf{H}} \mathrm{D}_{\mathrm{KL}} \left(\frac{\mathbf{G}}{Z_{\mathbf{G}}} \middle\| \frac{1}{Z_{\mathbf{W},\mathbf{H}}} \mathrm{e}^{\mathbf{W}\mathbf{H}^T} \right).$$

Here, $Z_{\mathbf{G}} = \sum_{ij} G_{ij}$ normalizes $\mathbf{G}$ into an estimate of the joint pmf of X and Y, and similarly $Z_{\mathbf{W},\mathbf{H}}$ is the constant that normalizes $\mathrm{e}^{\mathbf{W}\mathbf{H}^T}$ into a joint pmf. The expression $\mathrm{e}^{\mathbf{W}\mathbf{H}^T}$ denotes entry-wise exponentiation of $\mathbf{W}\mathbf{H}^T$.

Now we revisit the Skip-Gram training objective, and show that it differs from the SDR objective only slightly. Whereas the SDR objective measures the distance between the pmfs given by (normalized versions of) $\mathbf{G}$ and $\mathrm{e}^{\mathbf{W}\mathbf{H}^T}$, the Skip-Gram objective measures the distance between the pmfs given by (normalized versions of) *the rows* of $\mathbf{G}$ and $\mathrm{e}^{\mathbf{W}\mathbf{H}^T}$. That is, SDR emphasizes fitting the entire pmfs, while Skip-Gram emphasizes fitting conditional distributions.

Before presenting our main result, we state and prove the following lemma, which is of independent interest and is used in the proof of our main theorem. Recall that Skip-Gram represents each word c as a multinomial distribution over all other words w, and it learns the parameters for these distributions by a maximum likelihood estimation. It is known that learning model parameters by maximum likelihood estimation is equivalent to minimizing the KL-divergence of the learned model from the empirical distribution; the following lemma establishes the KL-divergence that Skip-Gram minimizes.

Lemma 2. *Let $\mathbf{G}$ be the word co-occurrence matrix constructed from the corpus on which a Skip-Gram model is trained, in which case G_{cw} is the number of times word w occurs as a neighboring word of c in the corpus. For each word c, let g_c denote the empirical frequency of the word in the corpus, so that*

$$g_c = \sum_w G_{cw} / \sum_{t,w} G_{t,w}.$$

Given a positive vector $\mathbf{x}$, let $\hat{\mathbf{x}} = \mathbf{x}/\|\mathbf{x}\|_1$. Then, the Skip-Gram model parameters $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_{|V|} \end{bmatrix}^T$ and $\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{u}_{|V|} \end{bmatrix}^T$

[2] Optimal in an information-theoretic sense: they preserve the maximal mutual information between any pair of random variables with the observed coocurrence statistics, without regard to the underlying joint distribution.

minimize the objective

$$\sum_c g_c \mathrm{D_{KL}}(\hat{\mathbf{g}^c} \,\|\, \widehat{\mathrm{e}^{\mathbf{u}_c^T \mathbf{V}^T}}),$$

where $\mathbf{g}^c$ *is the cth row of* G.

Proof. Recall that Skip-Gram chooses $\mathbf{U}$ and $\mathbf{V}$ to maximize

$$Q = \frac{1}{T} \sum_{i=1}^{T} \sum_{\substack{\delta=-C \\ \delta \neq 0}}^{C} \log p(w_{i+\delta}|w_i) \ ,$$

where

$$p(w|c) = \frac{\mathrm{e}^{\mathbf{u}_c^T \mathbf{v}_w}}{\sum_{i=1}^{n} \mathrm{e}^{\mathbf{u}_c^T \mathbf{v}_i}} \ .$$

This objective can be rewritten using the pairwise cooccurence statistics as

$$\begin{aligned}
Q &= \frac{1}{T} \sum_{c,w} G_{cw} \log p(w|c) \\
&= \frac{1}{T} \sum_c \left[\left(\sum_t G_{ct} \right) \sum_w \frac{G_{cw}}{\sum_t G_{ct}} \log p(w|c) \right] \\
&\propto \frac{1}{T} \sum_c \left[\frac{(\sum_t G_{ct})}{(\sum_{tw} G_{tw})} \sum_w \frac{G_{cw}}{\sum_t G_{ct}} \log p(w|c) \right] \\
&= \sum_c g_c \left(\sum_w \left(\hat{\mathbf{g}^c}\right)_w \log p(w|c) \right) \\
&= \sum_c g_c \left(-\mathrm{D_{KL}}(\hat{\mathbf{g}^c} \,\|\, p(\cdot|c)) - H(\hat{\mathbf{g}^c}) \right),
\end{aligned}$$

where $H(\cdot)$ denotes the entropy of a distribution. It follows that since Skip-Gram maximizes Q, it minimizes

$$\sum_c g_c \mathrm{D_{KL}}(\hat{\mathbf{g}^c} \,\|\, p(\cdot|c)) = \sum_c g_c \mathrm{D_{KL}}(\hat{\mathbf{g}^c} \,\|\, \widehat{\mathrm{e}^{\mathbf{u}_c^T \mathbf{V}^T}}).$$

□

We now prove our main theorem of this section, which states that SDR parameters can be obtained by augmenting the Skip-Gram embeddings to account for word frequencies.

Theorem 3. *Let* $\mathbf{U}, \mathbf{V}$ *be the results of fitting a Skip-Gram model to* $\mathbf{G}$*, and consider the augmented matrices*

$$\tilde{\mathbf{U}} = [\mathbf{U} \,|\, \boldsymbol{\alpha}] \text{ and } \tilde{\mathbf{V}} = [\mathbf{V} \,|\, \mathbf{1}],$$

where

$$\alpha_c = \log \left(\frac{g_c}{\sum_w \mathrm{e}^{\mathbf{u}_c^T \mathbf{v}_w}} \right) \text{ and } g_c = \frac{\sum_w G_{c,w}}{\sum_{t,w} G_{t,w}}.$$

Then, the features $(\tilde{\mathbf{U}}, \tilde{\mathbf{V}})$ *constitute a sufficient dimensionality reduction of* $\mathbf{G}$.

Proof. For convenience, let $\overline{\mathbf{G}}$ denote the joint pdf matrix $\mathbf{G}/Z_{\mathbf{G}}$, and let $\widehat{\mathbf{G}}$ denote the matrix obtained by normalizing each row of $\mathbf{G}$ to be a probability distribution. Then, it suffices to show that $D_{KL}(\overline{\mathbf{G}} \,\|\, q_{\mathbf{W},\mathbf{H}})$ is minimized over the set of probability distributions

$$\left\{ q_{\mathbf{W},\mathbf{H}} \,\middle|\, q_{\mathbf{W},\mathbf{H}}(w,c) = \frac{1}{Z} \left(\mathrm{e}^{\mathbf{W}\mathbf{H}^T} \right)_{cw} \right\},$$

when $\mathbf{W} = \tilde{\mathbf{U}}$ and $\mathbf{H} = \tilde{\mathbf{V}}$.

To establish this result, we use a chain rule for the KL-divergence. Recall that if we denote the expected KL-divergence between two marginal pmfs by

$$\begin{aligned}
&D_{KL}(p(\cdot|c) \| q(\cdot|c)) \\
&\quad = \sum_c p(c) \left(\sum_w p(w|c) \log \left(\frac{p(w|c)}{q(w|c)} \right) \right),
\end{aligned}$$

then the KL-divergence satisfies the chain rule:

$$\begin{aligned}
&D_{KL}(p(w,c) \| q(w,c)) \\
&\quad = D_{KL}(p(c) \| q(c)) + D_{KL}(p(w|c) \| q(w|c)).
\end{aligned}$$

Using this chain rule, we get

$$\begin{aligned}
&D_{KL}(\overline{\mathbf{G}} \,\|\, q_{\mathbf{W},\mathbf{H}}(w,c)) \qquad (9) \\
&\quad = D_{KL}(\mathbf{g} \,\|\, q_{\mathbf{W},\mathbf{H}}(c)) + D_{KL}(\widehat{\mathbf{G}} \| q_{\mathbf{W},\mathbf{H}}(w|c)).
\end{aligned}$$

Note that the second term in this sum is, in the notation of Lemma 2,

$$D_{KL}(\widehat{\mathbf{G}} \| q_{\mathbf{W},\mathbf{H}}(w|c)) = \sum_c g_c \mathrm{D_{KL}}(\hat{\mathbf{g}^c} \,\|\, \widehat{\mathrm{e}^{\mathbf{w}_c^T \mathbf{H}^T}}),$$

so the matrices $\mathbf{U}$ and $\mathbf{V}$ that are returned by fitting the Skip-Gram model minimize the second term in this sum. We now show that the augmented matrices $\mathbf{W} = \tilde{\mathbf{U}}$ and $\mathbf{H} = \tilde{\mathbf{V}}$ also minimize this second term, and in addition they make the first term vanish.

To see that the first of these claims holds, i.e., that the augmented matrices make the second term in (9) vanish, note that

$$q_{\tilde{\mathbf{U}},\tilde{\mathbf{V}}}(w|c) \propto \mathrm{e}^{\tilde{\mathbf{u}}_c^T \tilde{\mathbf{v}}_w} = \mathrm{e}^{\mathbf{u}_c^T \mathbf{v}_w + \alpha_c} \propto q_{\mathbf{U},\mathbf{V}}(w|c),$$

and the constant of proportionality is independent of w. It follows that $q_{\tilde{\mathbf{U}},\tilde{\mathbf{V}}}(w|c) = q_{\mathbf{U},\mathbf{V}}(w|c)$ and

$$D_{KL}(\widehat{\mathbf{G}} \,\|\, q_{\tilde{\mathbf{U}},\tilde{\mathbf{V}}}(w|c)) = D_{KL}(\widehat{\mathbf{G}} \,\|\, q_{\mathbf{U},\mathbf{V}}(w|c)).$$

Thus, the choice $\mathbf{W} = \tilde{\mathbf{U}}$ and $\mathbf{H} = \tilde{\mathbf{V}}$ minimizes the second term in (9).

To see that the augmented matrices make the first term in (9) vanish, observe that when $\mathbf{W} = \tilde{\mathbf{U}}$ and $\mathbf{H} = \tilde{\mathbf{V}}$, we have that $q_{\tilde{\mathbf{U}},\tilde{\mathbf{V}}}(c) = \mathbf{g}$ by construction. This can be verified by calculation:

$$
\begin{aligned}
q_{\tilde{\mathbf{U}},\tilde{\mathbf{V}}}(c) &= \frac{\sum_w q_{\tilde{\mathbf{U}},\tilde{\mathbf{V}}}(w,c)}{\sum_{w,t} q_{\tilde{\mathbf{U}},\tilde{\mathbf{V}}}(w,t)} = \frac{\sum_w \mathrm{e}^{\mathbf{u}_c^T \mathbf{v}_w + \alpha_c}}{\sum_{w,t} \mathrm{e}^{\mathbf{u}_t^T \mathbf{v}_w + \alpha_t}} \\
&= \frac{\left(\sum_w \mathrm{e}^{\mathbf{u}_c^T \mathbf{v}_w}\right) \mathrm{e}^{\alpha_c}}{\sum_t \left(\sum_w \mathrm{e}^{\mathbf{u}_t^T \mathbf{v}_w}\right) \mathrm{e}^{\alpha_t}} \\
&= \frac{\left[(\mathrm{e}^{\mathbf{U}\mathbf{V}^T}\mathbf{1}) \odot \mathrm{e}^{\boldsymbol{\alpha}}\right]_c}{\mathbf{1}^T \left[(\mathrm{e}^{\mathbf{U}\mathbf{V}^T}\mathbf{1}) \odot \mathrm{e}^{\boldsymbol{\alpha}}\right]}.
\end{aligned}
$$

Here, the notation $\mathbf{x} \odot \mathbf{y}$ denotes entry-wise multiplication of vectors.

Since $\alpha_c = \log(g_c) - \log\left(\left(\mathrm{e}^{\mathbf{U}\mathbf{V}^T}\mathbf{1}\right)_c\right)$, we have

$$
q_{\tilde{\mathbf{U}},\tilde{\mathbf{V}}}(c) = \frac{\left[(\mathrm{e}^{\mathbf{U}\mathbf{V}^T}\mathbf{1}) \odot \mathrm{e}^{\boldsymbol{\alpha}}\right]_c}{\mathbf{1}^T \left[(\mathrm{e}^{\mathbf{U}\mathbf{V}^T}\mathbf{1}) \odot \mathrm{e}^{\boldsymbol{\alpha}}\right]} = \frac{g_c}{\sum_t g_t} = g_c.
$$

The choice $\mathbf{W} = \tilde{\mathbf{U}}$ and $\mathbf{H} = \tilde{\mathbf{V}}$ makes the first term in (9) vanish, and it also minimizes the second term in (9). Thus, it follows that the features $(\tilde{\mathbf{U}}, \tilde{\mathbf{V}})$ constitute a sufficient dimensionality reduction of $\mathbf{G}$. □

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to PMI-based word embeddings. *Transactions of the Association for Computational Linguistics* 4:385–399.

Ehsaneddin Asgari and Mohammad R.K. Mofrad. 2015. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS One* 10(11).

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal Of Machine Learning Research* 3:1137–1155.

Kenneth Ward Church and Patrick Hanks. 1990. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics* 16(1):22–29.

J.R. Firth. 1957. A synopsis of linguistic theory 1930-1955. *Studies in Linguistic Analysis* pages 1–32.

Amir Globerson and Naftali Tishby. 2003. Sufficient Dimensionality Reduction. *Journal of Machine Learning Research* 3:1307–1331.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 855–864.

Omer Levy and Yoav Goldberg. 2014. Linguistic Regularities in Sparse and Explicit Word Representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. pages 171–180.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems*. pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings*. pages 746–751.

Andriy Mnih and Geoffrey Hinton. 2007. Three New Graphical Models for Statistical Language Modelling. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, pages 641–648.

Denis Paperno and Marco Baroni. 2016. When the Whole is Less than the Sum of Its Parts: How Composition Affects PMI Values in Distributional Semantic Vectors. *Computational Linguistics* 42:345–350.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Steven T. Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review* 21(5):1112–1130.

Douglas L. T. Rohde, Laura M. Gonnerman, and David C. Plaut. 2006. An improved model of semantic similarity based on lexical co-occurence. *Communications of the ACM* 8:627–633.

The State of the Art in Semantic Representation

Omri Abend **Ari Rappoport**

Department of Computer Science, The Hebrew University of Jerusalem
{oabend|arir}@cs.huji.ac.il

Abstract

Semantic representation is receiving growing attention in NLP in the past few years, and many proposals for semantic schemes (e.g., AMR, UCCA, GMB, UDS) have been put forth. Yet, little has been done to assess the achievements and the shortcomings of these new contenders, compare them with syntactic schemes, and clarify the general goals of research on semantic representation. We address these gaps by critically surveying the state of the art in the field.

1 Introduction

Schemes for Semantic Representation of Text (SRT) aim to reflect the meaning of sentences and texts in a transparent way. There has recently been an influx of proposals for semantic representations and corpora, e.g. GMB (Basile et al., 2012), AMR (Banarescu et al., 2013), UCCA (Abend and Rappoport, 2013b) and Universal Decompositional Semantics (UDS; White et al., 2016). Nevertheless, no detailed assessment of the relative merits of the different schemes has been carried out, nor their comparison to previous sentential analysis schemes, notably syntactic ones. An understanding of the achievements and gaps of semantic analysis in NLP is crucial to its future prospects.

In this paper we begin to chart the various proposals for semantic schemes according to the **content** they support. As not many semantic queries on texts can at present be answered with near human-like reliability without using manual symbolic annotation, we will mostly focus on schemes that represent semantic distinctions explicitly.[1]

We begin by discussing the goals of SRT in Section 2. Section 3 surveys major represented meaning components, including predicate-argument relations, discourse relations and logical structure. Section 4 details the various concrete proposals for SRT schemes and annotated resources, while Sections 5 and 6 discuss criteria for their evaluation and their relation to syntax, respectively.

We find that despite the major differences in terms of formalism and interface with syntax, in terms of their content there is a great deal of convergence of SRT schemes. Principal differences between schemes are mostly related to their ability to abstract away from formal and syntactic variation, namely to assign similar structures to different constructions that have a similar meaning, and to assign different structures to constructions that have different meanings, despite their surface similarity. Other important differences are in the level of training they require from their annotators (e.g., expert annotators vs. crowd-sourcing) and in their cross-linguistic generality. We discuss the complementary strengths of different schemes, and suggest paths for future integration.

2 Defining Semantic Representation

The term *semantics* is used differently in different contexts. For the purposes of this paper we define a semantic representation as one that reflects the meaning of the text as it is understood by a language speaker. A semantic representation should thus be paired with a method for extracting information from it that can be directly evaluated by humans. The extraction process should be reliable and computationally efficient.

[1] Note that even a string representation of text can be regarded as semantic given a reliable enough parser.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 77–89
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1008

We stipulate that a fundamental component of the content conveyed by SRTs is argument structure – who did what to whom, where, when and why, i.e., events, their participants and the relations between them. Indeed, the fundamental status of argument structure has been recognized by essentially all approaches to semantics both in theoretical linguistics (Levin and Hovav, 2005) and in NLP, through approaches such as Semantic Role Labeling (SRL; Gildea and Jurafsky, 2002), formal semantic analysis (e.g., Bos, 2008), and Abstract Meaning Representation (AMR; Banarescu et al., 2013). Many other useful meaning components have been proposed, and are discussed at a greater depth in Section 3.

Another approach to defining an SRT is through external (extra-textual) criteria or applications. For instance, a semantic representation can be defined to support inference, as in textual entailment (Dagan et al., 2006) or natural logic (Angeli and Manning, 2014). Other examples include defining a semantic representation in terms of supporting knowledge base querying (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005), or defining semantics through a different modality, for instance interpreting text in terms of images that correspond to it (Kiros et al., 2014), or in terms of embodied motor and perceptual schemas (Feldman et al., 2010).

A different approach to SRT is taken by Vector Space Models (VSM), which eschew the use of symbolic structures, instead modeling all linguistic elements as vectors, from the level of words to phrases and sentences. Proponents of this approach generally invoke neural network methods, obtaining impressive results on a variety of tasks including lexical tasks such as cross-linguistic word similarity (Ammar et al., 2016), machine translation (Bahdanau et al., 2015), and dependency parsing (Andor et al., 2016). VSMs are also attractive in being flexible enough to model non-local and gradient phenomena (e.g., Socher et al., 2013). However, more research is needed to clarify the scope of semantic phenomena that such models are able to reliably capture. We therefore only lightly touch on VSMs in this survey.

Finally, a major consideration in semantic analysis, and one of its great potential advantages, is its cross-linguistic universality. While languages differ in terms of their form (e.g., in their phonology, lexicon, and syntax), they have often been assumed to be much closer in terms of their semantic content (Bar-Hillel, 1960; Fodor, 1975). See Section 5 for further discussion.

A terminological note: within formal linguistics, semantics is often the study of the relation between symbols (e.g., words, syntactic constructions) and what they signify. In this sense, semantics is the study of the aspects of meaning that are overtly expressed by the lexicon and grammar of a language, and is thus tightly associated with a theory of the syntax-semantics interface. We note that this definition of semantics is somewhat different from the one intended here, which defines semantic schemes as theories of meaning.

3 Semantic Content

We turn to discussing the main content types encoded by semantic representation schemes. Due to space limitations, we focus only on text semantics, which studies the meaning relationships between lexical items, rather than the meaning of the lexical items themselves.[2] We also defer discussion of more targeted semantic distinctions, such as sentiment, to future work.

We will use the following as a running example:

(1) Although Ann was leaving, she gave the present to John.

Events. Events (sometimes called frames, propositions or scenes) are the basic building blocks of argument structure representations. An event includes a predicate (main relation, frame-evoking element), which is the main determinant of what the event is about. It also includes arguments (participants, core elements) and secondary relations (modifiers, non-core elements). Example 1 is usually viewed as having two events, evoked by "leaving" and "gave".

Schemes commonly provide an ontology or a lexicon of event types (also a predicate lexicon), which categorizes semantically similar events evoked by different lexical items. For instance, FrameNet defines frames as schematized story fragments evoked by a set of conceptually similar predicates. In (1), the frames evoked by "leaving" and "gave" are DEPARTING and GIVING, but DEPARTING may also be evoked by "depart" and "exit", and GIVING by "donate" and "gift".

[2] We use the term "Text Semantics", rather than the commonly used "Sentence Semantics" to include inter-sentence semantic relations as well.

The events discussed here should not be confused with events as defined in Information Extraction and related tasks such as event coreference (Humphreys et al., 1997), which correspond more closely to the everyday notion of an event, such as a political or financial event, and generally consist of multiple events in the sense discussed here. The representation of such events is recently receiving considerable interest within NLP, e.g. the Richer Event Descriptions framework (RED; Ikuta et al., 2014).

Predicates and Arguments. While predicate-argument relations are universally recognized as fundamental to semantic representation, the interpretation of the terms varies across schemes. Most SRL schemes cover a wide variety of verbal predicates, but differ in which nominal and adjectival predicates are covered. For example, PropBank (Palmer et al., 2005), one of the major resources for SRL, covers verbs, and in its recent versions also eventive nouns and multi-argument adjectives. FrameNet (Ruppenhofer et al., 2016) covers all these, but also covers relational nouns that do not evoke an event, such as "president". Other lines of work address semantic arguments that appear outside sentence boundaries, or that do not explicitly appear anywhere in the text (Gerber and Chai, 2010; Roth and Frank, 2015).

Core and Non-core Arguments. Perhaps the most common distinction between argument types is between core and non-core arguments (Dowty, 2003). While it is possible to define the distinction distributionally as one between obligatory and optional arguments, here we focus on the semantic dimension, which distinguishes arguments whose meaning is predicate-specific and are necessary components of the described event (core), and those which are predicate-general (non-core). For example, FrameNet defines core arguments as conceptually necessary components of a frame, that make the frame unique and different from other frames, and peripheral arguments as those that introduce additional, independent or distinct relations from that of the frame such as time, place, manner, means and degree (Ruppenhofer et al., 2016, pp. 23-24).

Semantic Roles. Semantic roles are categories of arguments. Many different semantic role inventories have been proposed and used in NLP over the years, the most prominent being FrameNet (where roles are shared across predicates that evoke the same frame type, such as "leave" and "depart"), and PropBank (where roles are verb-specific). PropBank's role sets were extended by subsequent projects such as AMR. Another prominent semantic role inventory is VerbNet (Kipper et al., 2008) and subsequent projects (Bonial et al., 2011; Schneider et al., 2015), which define a closed set of abstract semantic roles (such as AGENT, PATIENT and INSTRUMENT) that apply to all predicate arguments.

Co-reference and Anaphora. Co-reference allows to abstract away from the different ways to refer to the same entity, and is commonly included in semantic resources. Coreference interacts with argument structure annotation, as in its absence each argument is arbitrarily linked to one of its textual instances. Most SRL schemes would mark "Ann" in (1) as an argument of "leaving" and "she" as an argument of "gave", although on semantic grounds "Ann" is an argument of both.

Some SRTs distinguish between the cases of argument sharing which is encoded by the syntax and is thus explicit (e.g., in "John went home and took a shower", "John" is both an argument of "went home" and of "took a shower"), and cases where the sharing of arguments is inferred (as in (1)). This distinction may be important for text understanding, as the inferred cases tend to be more ambiguous ("she" in (1) might not refer to "Ann"). Other schemes, such as AMR, eschew this distinction and use the same terms to represent all cases of coreference.

Temporal Relations. Most temporal semantic work in NLP has focused on temporal relations between events, either by timestamping them according to time expressions found in the text, or by predicting their relative order in time. Important resources include TimeML, a specification language for temporal relations (Pustejovsky et al., 2003), and the TempEval series of shared tasks and annotated corpora (Verhagen et al., 2009, 2010; UzZaman et al., 2013). A different line of work explores scripts: schematic, temporally ordered sequences of events associated with a certain scenario (Chambers and Jurafsky, 2008, 2009; Regneri et al., 2010). For instance, going to a restaurant includes sitting at a table, ordering, eating and paying, generally in this order.

Related to temporal relations, are causal relations between events, which are ubiquitous in language, and central for a variety of applications,

including planning and entailment. See (Mirza et al., 2014) and (Dunietz et al., 2015) for recently proposed annotation schemes for causality and its sub-types. Mostafazadeh et al. (2016) integrated causal and TimeML-style temporal relations into a unified representation.

The internal temporal structure of events has been less frequently tackled. Moens and Steedman (1988) defined an ontology for the temporal components of an event, such as its preparatory process (e.g., "climbing a mountain"), or its culmination ("reaching its top"). Statistical work on this topic is unfortunately scarce, and mostly focuses on lexical categories such as aspectual classes (Siegel and McKeown, 2000; Palmer et al., 2007; Friedrich et al., 2016; White et al., 2016), and tense distinctions (Elson and McKeown, 2010). Still, casting events in terms of their temporal components, characterizing an annotation scheme for doing so and rooting it in theoretical foundations, is an open challenge for NLP.

Spatial Relations. The representation of spatial relations is pivotal in cognitive theories of meaning (e.g., Langacker, 2008), and in application domains such as geographical information systems or robotic navigation. Important tasks in this field include Spatial Role Labeling (Kordjamshidi et al., 2012) and the more recent SpaceEval (Pustejovsky et al., 2015). The tasks include the identification and classification of spatial elements and relations, such as places, paths, directions and motions, and their relative configuration.

Discourse Relations encompass any semantic relation between events or larger semantic units. For example, in (1) the leaving and the giving events are sometimes related through a discourse relation of type CONCESSION, evoked by "although". Such information is useful, often essential for a variety of NLP tasks such as summarization, machine translation and information extraction, but is commonly overlooked in the development of such systems (Webber and Joshi, 2012).

The Penn Discourse Treebank (PeDT; Miltsakaki et al., 2004) annotates discourse units, and classifies the relations between them into a hierarchical, closed category set, including high-level relation types like TEMPORAL, COMPARISON and CONTINGENCY and finer-grained ones such as JUSTIFICATION and EXCEPTION. Another commonly used resource is the RST Discourse Treebank (Carlson et al., 2003), which places more focus on higher-order discourse structures, resulting in deeper hierarchical structures than the PeDT's, which focuses on local discourse structure.

Another discourse information type explored in NLP is discourse segmentation, where texts are partitioned into shallow structures of discourse units categorized either according to their topic or according to their function within the text. An example is the segmentation of scientific papers into functional segments and their labeling with categories such as BACKGROUND and DISCUSSION (Liakata et al., 2010). See (Webber et al., 2011) for a survey of discourse structure in NLP.

Discourse relations beyond the scope of a single sentence are often represented by specialized semantic resources and not by general ones, despite the absence of a clear boundary line between them. This, however, is beginning to change with some schemes, e.g., GMB and UCCA, already supporting cross-sentence semantic relations.[3]

Logical Structure. Logical structure, including quantification, negation, coordination and their associated scope distinctions, is the cornerstone of semantic analysis in much of theoretical linguistics, and has attracted much attention in NLP as well. Common representations are often based on variants of predicate calculus, and are useful for applications that require mapping text into an external, often executable, formal language, such as a querying language (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005) or robot instructions (Artzi and Zettlemoyer, 2013). Logical structures are also useful for recognizing entailment relations between sentences, as some entailments can be computed from the text's logical structure by formal provers (Bos and Markert, 2005; Lewis and Steedman, 2013).

Inference and Entailment. A primary motivation for many semantic schemes is their ability to support inference and entailment. Indeed, means for predicting logical entailment are built into many forms of semantic representations. A different approach was taken in the tasks of Recognizing Textual Entailment (Dagan et al., 2013), and Natural Logic (van Eijck, 2005), which considers an inference valid if a reasonable annotator would find the hypothesis likely to hold given

[3] AMR will also support discourse structure in its future versions (N. Schneider; personal communication).

the premise, even if it cannot be deduced from it. See (Manning, 2006) for a discussion of this point. Such inference relations are usually not included in semantic treebanks, but annotated in specialized resources (e.g., Dagan et al., 2006; Bowman et al., 2015).

4 Semantic Schemes and Resources

This section briefly surveys the different schemes and resources for SRT. We focus on design principles rather than specific features, as the latter are likely to change as the schemes undergo continuous development. In general, schemes discussed in Section 3 are not repeated here.

Semantic Role Labeling. SRL schemes diverge in their event types, the type of predicates they cover, their granularity, their cross-linguistic applicability, their organizing principles and their relation with syntax. Most SRL schemes define their annotation relative to some syntactic structure, such as parse trees of the PTB in the case of PropBank, or specialized syntactic categories defined for SRL purposes in the case of FrameNet. Other than PropBank, FrameNet and VerbNet discussed above, other notable resources include Semlink (Loper et al., 2007) that links corresponding entries in different resources such as PropBank, FrameNet, VerbNet and WordNet, and the Preposition Supersenses project (Schneider et al., 2015), which focuses on roles evoked by prepositions. See (Palmer et al., 2010, 2013) for a review of SRL schemes and resources. SRL schemes are often termed "shallow semantic analysis" due to their focus on argument structure, leaving out other relations such as discourse events, or how predicates and arguments are internally structured.

AMR. AMR covers predicate-argument relations, including semantic roles (adapted from PropBank) that apply to a wide variety of predicates (including verbal, nominal and adjectival predicates), modifiers, co-reference, named entities and some time expressions.

AMR does not currently support relations above the sentence level, and is admittedly English-centric, which results in an occasional conflation of semantic phenomena that happen to be similarly realized in English, into a single semantic category. AMR thus faces difficulties when assessing the invariance of its structures across translations (Xue et al., 2014). As an example, consider the sentences "I happened to meet Jack in the office", and "I asked to meet Jack in the office". While the two have similar syntactic forms, the first describes a single "meeting" event, where "happened" is a modifier, while the second describes two distinct events: asking and meeting. AMR annotates both in similar terms, which may be suitable for English, where aspectual relations are predominantly expressed as subordinating verbs (e.g., "begin", "want"), and are syntactically similar to primary verbs that take an infinitival complement (such as "ask to meet" or "learn to swim"). However, this approach is less suitable cross-linguistically. For instance, when translating the sentences to German, the divergence between the semantics of the two sentences is clear: in the first "happened" is translated to an adverb: "Ich habe Jack im Büro zufällig getroffen" (lit. "I have Jack in-the office by-chance met"), and in the second "asked" is translated to a verb: "Ich habe gebeten, Jack im Büro zu treffen" (lit. "I have asked, Jack in-the office to meet").

UCCA. UCCA (Universal Conceptual Cognitive Annotation) (Abend and Rappoport, 2013a,b) is a cross-linguistically applicable scheme for semantic annotation, building on typological theory, primarily on Basic Linguistic Theory (Dixon, 2010). UCCA's foundational layer of categories focuses on argument structures of various types and relations between them. In its current state, UCCA is considerably more coarse-grained than the above mentioned schemes (e.g., it does not include semantic role information). However, its distinctions tend to generalize well across languages (Sulem et al., 2015). For example, unlike AMR, it distinguishes between primary and aspectual verbs, so cases such as "happened to meet" are annotated similarly to cases such as "met by chance", and differently from "asked to meet".

Another design principle UCCA evokes is support for annotation by non-experts. To do so the scheme reformulates some of the harder distinctions into more intuitive ones. For instance, the core/non-core distinction is replaced in UCCA with the distinction between pure relations (Adverbials) and those evoking an object (Participants), which has been found easier for annotators to apply.

UDS. Universal Decompositional Semantics (White et al., 2016) is a multi-layered scheme, which currently includes semantic role anno-

tation, word senses and aspectual classes (e.g., realis/irrealis). UDS emphasizes accessible distinctions, which can be collected through crowd-sourcing. However, the skeletal structure of UDS representations is derived from syntactic dependencies, and only includes verbal argument structures that can be so extracted. Notably, many of the distinctions in UDS are defined using feature bundles, rather than mutually exclusive categories. For instance, a semantic role may be represented as having the features +VOLITION and +AWARENESS, rather than as having the category AGENT.

The Prague Dependency Treebank (PDT) Tectogrammatical Layer (PDT-TL) (Sgall, 1992; Böhmová et al., 2003) covers a rich variety of functional and semantic distinctions, such as argument structure (including semantic roles), tense, ellipsis, topic/focus, co-reference, word sense disambiguation and local discourse information. The PDT-TL results from an abstraction over PDT's syntactic layers, and its close relation with syntax is apparent. For instance, the PDT-TL encodes the distinction between a governing clause and a dependent clause, which is primarily syntactic in nature, so in the clauses "John came just as we were leaving" and "We were leaving just as John came" the governing and dependent clause are swapped, despite their semantic similarity.

CCG-based Schemes. CCG (Steedman, 2000) is a lexicalized grammar (i.e., nearly all semantic content is encoded in the lexicon), which defines a theory of how lexical information is composed to form the meaning of phrases and sentences (see Section 6.2), and has proven effective in a variety of semantic tasks (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010; Artzi and Zettlemoyer, 2013, *inter alia)*. Several projects have constructed logical representations by associating CCG with semantic forms (by assigning logical forms to the leaves). For example, Boxer (Bos, 2008) and GMB, which builds on Boxer, use Discourse Representation Structures (Kamp and Reyle, 1993), while Lewis and Steedman (2013) used Davidsonian-style λ-expressions, accompanied by lexical categorization of the predicates. These schemes encode events with their argument structures, and include an elaborate logical structure, as well as lexical and discourse information.

HPSG-based Schemes. Related to CCG-based schemes are SRTs based on Head-driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994), where syntactic and semantic features are represented as feature bundles, which are iteratively composed through unification rules to form composite units. HPSG-based SRT schemes commonly use the Minimal Recursion Semantics (Copestake et al., 2005) formalism. Annotated corpora and manually crafted grammars exist for multiple languages (Flickinger, 2002; Oepen et al., 2004; Bender and Flickinger, 2005, *inter alia)*, and generally focus on argument structural and logical semantic phenomena. The Broad-coverage Semantic Dependency Parsing shared task and corpora (Oepen et al., 2014, 2015) include corpora annotated with the PDT-TL, and dependencies extracted from the HPSG grammars Enju (Miyao, 2006) and the LinGO English Reference Grammar (ERG; Flickinger, 2002).

Like the PDT-TL, projects based on CCG, HPSG, and other expressive grammars such as LTAG (Joshi and Vijay-Shanker, 1999) and LFG (Kaplan and Bresnan, 1982) (e.g., GlueTag (Frank and van Genabith, 2001)), yield semantic representations that are coupled with syntactic ones. While this approach provides powerful tools for inference, type checking, and mapping into external formal languages, it also often results in difficulties in abstracting away from some syntactic details. For instance, the dependencies derived from ERG in the SDP corpus use the same label for different senses of the English possessive construction, regardless of whether they correspond to ownership (e.g., "John's dog") or to a different meaning, such as marking an argument of a nominal predicate (e.g., "John's kick"). See Section 6.

OntoNotes is a useful resource with multiple inter-linked layers of annotation, borrowed from different schemes. The layers include syntactic, SRL, co-reference and word sense disambiguation content. Some properties of the predicate, such as which nouns are eventive, are encoded as well.

To summarize, while SRT schemes differ in the types of content they support, schemes evolve to continuously add new content types, making these differences less consequential. The fundamental difference between the schemes is the extent that they abstract away from syntax. For instance, AMR and UCCA abstract away from syntax as part of their design, while in most other schemes syntax and semantics are more tightly coupled.

Schemes also differ in other aspects discussed in Sections 5 and 6.

5 Evaluation

Human evaluation is the ultimate criterion for validating an SRT scheme given our definition of semantics as meaning as it is understood by a language speaker. Determining how well an SRT scheme corresponds to human interpretation of a text is ideally carried out by asking annotators to make some semantic prediction or annotation according to pre-specified guidelines, and to compare this to the information extracted from the SRT. Question Answering SRL (QASRL; He et al., 2015) is an SRL scheme which solicits non-experts to answer mostly wh-questions, converting their output to an SRL annotation. Hartshorne et al. (2013) and Reisinger et al. (2015) use crowdsourcing to elicit semantic role features, such as whether the argument was volitional in the described event, in order to evaluate proposals for semantic role sets.

Another evaluation approach is task-based evaluation. Many semantic representations in NLP are defined with an application in mind, making this type of evaluation natural. For instance, a major motivation for AMR is its applicability to machine translation, making MT a natural (albeit hitherto unexplored) testbed for AMR evaluation. Another example is using question answering to evaluate semantic parsing into knowledge-base queries.

Another common criterion for evaluating a semantic scheme is *invariance*, where semantic analysis should be similar across paraphrases or translation pairs (Xue et al., 2014; Sulem et al., 2015). For instance, most SRL schemes abstract away from the syntactic divergence between the sentences (1) "He gave a present to John" and (2) "It was John who was given a present" (although a complete analysis would reflect the difference of focus between them).

Importantly, these evaluation criteria also apply in cases where the representation is automatically induced, rather than manually defined. For instance, vector space representations are generally evaluated either through task-based evaluation, or in terms of semantic features computed from them, whose validity is established by human annotators (e.g., Agirre et al., 2013, 2014).

Finally, where semantic schemes are induced through manual annotation (and not through automated procedures), a common criterion for determining whether the guidelines are sufficiently clear, and whether the categories are well-defined is to measure agreement between annotators, by assigning them the same texts and measuring the similarity of the resulting structures. Measures include the SMATCH measure for AMR (Cai and Knight, 2013), and the PARSEVAL F-score (Black et al., 1991) adapted for DAGs for UCCA.

SRT schemes diverge in the background and training they require from their annotators. Some schemes require extensive training (e.g., AMR), while others can be (at least partially) collected by crowdsourcing (e.g., UDS). Other examples include FrameNet, which requires expert annotators for creating new frames, but employs less trained in-house annotators for applying existing frames to texts; QASRL, which employs non-expert annotators remotely; and UCCA, which uses in-house non-experts, demonstrating no advantage to expert over non-expert annotators after an initial training period. Another approach is taken by GMB, which uses online collaboration where expert collaborators participate in manually correcting automatically created representations. They further employ gamification strategies for collecting some aspects of the annotation.

Universality. One of the great promises of semantic analysis (over more surface forms of analysis) is its cross-linguistic potential. However, while the theoretical and applicative importance of universality in semantics has long been recognized (Goddard, 2011), the nature of universal semantics remains unknown. Recently, projects such as BabelNet (Ehrmann et al., 2014), UBY (Gurevych et al., 2012) and Open Multilingual Wordnet[4], constructed huge multi-lingual semantic nets, by linking resources such as Wikipedia and WordNet and processing them using modern NLP. However, such projects currently focus on lexical semantic and encyclopedic information rather than on text semantics.

Symbolic SRT schemes such as SRL schemes and AMR have also been studied for their cross-linguistic applicability (Padó and Lapata, 2009; Sun et al., 2010; Xue et al., 2014), indicating partial portability across languages. Translated versions of PropBank and FrameNet have been constructed for multiple languages (e.g., Akbik et al., 2016; Hartmann and Gurevych, 2013). How-

[4] http://compling.hss.ntu.edu.sg/omw/

ever, as both PropBank and FrameNet are lexicalized schemes, and as lexicons diverge wildly across languages, these schemes require considerable adaptation when ported across languages (Kozhevnikov and Titov, 2013). Ongoing research tackles the generalization of VerbNet's unlexicalized roles to a universally applicable set (e.g., Schneider et al., 2015). Few SRT schemes place cross-linguistically applicability as one of their main criteria, examples include UCCA, and the LinGO Grammar Matrix (Bender and Flickinger, 2005), both of which draw on typological theory.

Vector space models, which embed words and sentences in a vector space, have also been applied to induce a shared cross-linguistic space (Klementiev et al., 2012; Rajendran et al., 2015; Wu et al., 2016). However, further evaluation is required in order to determine what aspects of meaning these representations reflect reliably.

6 Syntax and Semantics

6.1 Syntactic and Semantic Generalization

Syntactic distinctions are generally guided by a combination of semantic and distributional considerations, where emphasis varies across schemes.

Consider phrase-based syntactic structures, common examples of which, such as the Penn Treebank for English (Marcus et al., 1993) and the Penn Chinese Treebank (Xue et al., 2005), are adaptations of X-bar theory. Constituents are commonly defined in terms of distributional criteria, such as whether they can serve as conjuncts, be passivized, elided or fronted (Carnie, 2002, pp. 50-53). Moreover, phrase categories are defined according to the POS category of their headword, such as Noun Phrase, Verb Phrase or Preposition Phrase, which are also at least partly distributional, motivated by their similar morphological and syntactic distribution. In contrast, SRT schemes tend to abstract away from these realizational differences and directly reflect the argument structure of the sentence using the same set of categories, irrespective of the POS of the predicate, or the case marking of its arguments.

Distributional considerations are also apparent with functional syntactic schemes (the most commonly used form of which in NLP are lexicalist dependency structures), albeit to a lesser extent. A prominent example is Universal Dependencies (UD; Nivre et al., 2016), which aims at producing a cross-linguistically consistent dependency-based annotation, and whose categories are motivated by a combination of distributional and semantic considerations. For example, UD would distinguish between the dependency type between "John" and "brother" in "John, my brother, arrived" and "John, who is my brother, arrived", despite their similar semantics. This is due to the former invoking an apposition, and the latter a relative clause, which are different in their distribution.

As an example of the different categorization employed by UD and by purely semantic schemes such as AMR and UCCA consider (1) "founding of the school", (2) "president of the United States" and (3) "United States president". UD is faithful to the syntactic structure and represents (1) and (2) similarly, while assigning a different structure to (3). In contrast, AMR and UCCA perform a semantic generalization and represents examples (2) and (3) similarly and differently from (1).

6.2 The Syntax-Semantics Interface

A common assumption on the interface between syntax and semantics is that semantics of phrases and sentences is compositional – it is determined recursively by the meaning of its immediate constituents and their syntactic relationships, which are generally assumed to form a closed set (Montague, 1970, and much subsequent work). Thus, the interpretation of a sentence can be computed bottom-up, by establishing the meaning of individual words, and recursively composing them, to obtain the full sentential semantics. The order and type of these compositions are determined by the syntactic structure.

Compositionality is employed by linguistically expressive grammars, such as those based on CCG and HPSG, and has proven to be a powerful method for various applications. See (Bender et al., 2015) for a recent discussion of the advantages of compositional SRTs. Nevertheless, a compositional account meets difficulties when faced with multi-word expressions and in accounting for cases like "he sneezed the napkin off the table", where it is difficult to determine whether "sneezed" or "off" account for the constructional meaning. Construction Grammar (Fillmore et al., 1988; Goldberg, 1995) answers these issues by using an open set of construction-specific compositional operators, and supporting lexical en-

tries of varying lengths. Several ongoing projects address the implementation of the principles of Construction Grammar into explicit grammars, including Sign-based Construction Grammar (Fillmore et al., 2012), Embodied Construction Grammar (Feldman et al., 2010) and Fluid Construction Grammar (Steels and de Beules, 2006).

The achievements of machine learning methods in many areas, and optimism as to its prospects, have enabled the approaches to semantics discussed in this paper. Machine learning allows to define semantic structures on purely semantic grounds and to let algorithms identify how these distinctions are mapped to surface/distributional forms. Some of the schemes discussed in this paper take this approach in its pure form (e.g., AMR and UCCA).

7 Conclusion

Semantic representation in NLP is undergoing rapid changes. Traditional semantic work has either used shallow methods that focus on specific semantic phenomena, or adopted formal semantic theories which are coupled with a syntactic scheme through a theory of the syntax-semantics interface. Recent years have seen increasing interest in an alternative approach that defines semantic structures independently from any syntactic or distributional criteria, much due to the availability of semantic treebanks that implement this approach.

Semantic schemes diverge in whether they are anchored in the words and phrases of the text (e.g., all types of semantic dependencies and UCCA) or not (e.g., AMR and logic-based representations). We do not view this as a major difference, because most unanchored representations (including AMR) retain their close affinity with the words of the sentence, possibly because of the absence of a workable scheme for lexical decomposition, while dependency structures can be converted into logic-based representations (Reddy et al., 2016). In practice, anchoring facilitates parsing, while unanchored representations are more flexible to use where words and semantic components are not in a one-to-one correspondence.

Our survey concludes that the main distinguishing factors between schemes are their relation to syntax, their degree of universality, and the expertise and training they require from annotators, an important factor in addressing the annotation bottleneck. We hope this survey of the state of the art in semantic representation will promote discussion, expose more researchers to the most pressing questions in semantic representation, and lead to the wide adoption of the best components from each scheme.

Acknowledgements. We thank Nathan Schneider for his helpful comments. The work was support by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

References

Omri Abend and Ari Rappoport. 2013a. UCCA: A semantic-based grammatical annotation scheme. In *Proc. of IWCS*. pages 1–12.

Omri Abend and Ari Rappoport. 2013b. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL*. pages 228–238.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proc. of SemEval*. pages 81–91.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Proc. of SemEval*. pages 32–43.

Alan Akbik, vishwajeet kumar, and Yunyao Li. 2016. Towards semi-automatic generation of proposition banks for low-resource languages. In *Proc. of EMNLP*. pages 993–998.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR* abs/1602.01925.

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

Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *EMNLP*. pages 534–545.

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

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proc. of LAW*. pages 178–186.

Yehoshua Bar-Hillel. 1960. The present status of automatic translation of languages. In *Advances in computers*, Academic Press, New York, volume 1, pages 91–163.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proc. of LREC*. pages 3196–3200.

Emily Bender and Dan Flickinger. 2005. Rapid prototyping of scalable grammars: Towards modularity in extensions to a language-independent core. In *Proc. of IJCNLP*. pages 203–208.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proc. of IWCS*. pages 239–249.

Ezra Black, Steve Abney, Dan Flickinger, C. Gdaniec, Ralph Grishman, P. Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith Klavans, Mark Liberman, Mitch Marcus, Salim Roukos, Beatrice Santorini, and Thomas Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proc. of the DARPA Speech and Natural Language Workshop*. pages 204–210.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague dependency treebank. In *Treebanks*, Springer, pages 103–127.

Claire Bonial, William Corvey, Martha Palmer, Volha V Petukhova, and Harry Bunt. 2011. A hierarchical unification of lirics and verbnet semantic roles. In *Semantic Computing (ICSC)*. pages 483–489.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In Johan Bos and Rodolfo Delmonte, editors, *Proc. of the Conference on Semantics in Text Processing (STEP)*. College Publications, Research in Computational Semantics, pages 277–286.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proc. of EMNLP*. pages 628–635.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proc. of EMNLP*. pages 632–642.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proc. of ACL*. pages 748–752.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, Springer, pages 85–112.

Andrew Carnie. 2002. *Syntax: A Generative Introduction*. Wiley-Blackwell.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proc. of ACL-HLT*. pages 789–797.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proc. of ACL-IJCNLP*. pages 602–610.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation* 3:281–332.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising text entailment challenge. In Bernardo Magnini Joaquin Quiñonero Candela, Ido Dagan and Florence d'Alché Buc, editors, *Machine Learning Challenges*, Springer, Berlin, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190.

Ido Dagan, Dan Roth, and Mark Sammons. 2013. *Recognizing textual entailment*. Morgan & Claypool Publishers.

Robert M.W. Dixon. 2010. *Basic Linguistic Theory: Methodology*, volume 1. Oxford University Press.

David Dowty. 2003. The dual analysis of adjuncts/complements in categorial grammar. In Ewald Lang, Claudia Maienborn, and Cathry Fabricius-Hansen, editors, *Modifying Adjuncts*, Mouton de Gruyter, Berlin, pages 33–66.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2015. Annotating causal language using corpus lexicography of constructions. In *Proc. of LAW*. pages 188–196.

Maud Ehrmann, Francesco Cecconi, Daniele Vannella, John Philip McCrae, Philipp Cimiano, and Roberto Navigli. 2014. Representing multilingual data as linked data: the case of babelnet 2.0. In *Proc. of LREC*. pages 401–408.

David K Elson and Kathleen R McKeown. 2010. Tense and aspect assignment in narrative discourse. In *Proc. of the International Natural Language Generation Conference*. pages 47–56.

Jerome Feldman, Ellen Dodge, and John Bryant. 2010. Embodied construction grammar. In Bernd Heine and Heiko Narrog, editors, *The Oxford Handbook of Linguistic Analysis*, Oxford University Press, pages 111–158.

Charles Fillmore, Russell Lee-Goldman, and Russell Rhodes. 2012. The FrameNet Constructicon. In Hans Boas and Ivan Sag, editors, *Sign-based construction grammar*, CSLI Publications, pages 309–372.

Charles J Fillmore, Paul Kay, and Mary C O'Connor. 1988. Regularity and idiomaticity in grammatical constructions: The case of *let alone*. *Language* 64(3):501–538.

Daniel Flickinger. 2002. On building a more efficient grammar by exploiting types. In Jun'ichi Tsujii, Stefan Oepen, Daniel Flickinger, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, CLSI, Stanford, CA.

Jerry A Fodor. 1975. *The language of thought*, volume 5. Harvard University Press.

Anette Frank and Josef van Genabith. 2001. Gluetag: Linear logic based semantics construction for ltag and what it teaches us about the relation between LFG and LTAG. In *Proc. of LFG*.

Annemarie Friedrich, Alexis Palmer, and Manfred Pinkal. 2016. Situation entity types: automatic classification of clause-level aspect. In *Proceedings of ACL 2016*. pages 1757–1768.

Matthew Gerber and Joyce Y Chai. 2010. Beyond nombank: A study of implicit arguments for nominal predicates. In *Proc. of ACL*. pages 1583–1592.

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

Cliff Goddard. 2011. *Semantic analysis: A practical introduction*. Oxford University Press, 2nd edition.

Adèle Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. Chicago University Press, Chicago.

Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer, and Christian Wirth. 2012. UBY - a large-scale unified lexical-semantic resource based on lmf. In *Proc. of EACL*. pages 580–590.

Silvana Hartmann and Iryna Gurevych. 2013. Framenet on the way to babel: Creating a bilingual framenet using wiktionary as interlingual connection. In *Proc. of ACL*. pages 1363–1373.

Joshua K. Hartshorne, Claire Bonial, and Martha Palmer. 2013. The VerbCorner project: Toward an empirically-based semantic decomposition of verbs. In *Proc. of EMNLP*. pages 1438–1442.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proc. of EMNLP*. pages 643–653.

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. 1997. Event coreference for information extraction. In *Proc. of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*. pages 75–81.

Rei Ikuta, Will Styler, Mariah Hamang, Tim O'Gorman, and Martha Palmer. 2014. Challenges of adding causation to richer event descriptions. In *Proc. of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*. pages 12–20.

Aravind Joshi and K. Vijay-Shanker. 1999. Compositional semantics with Lexicalized Tree-Adjoining Grammar (LTAG). In *Proc. of IWCS*. pages 131–146.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer, Dordrecht.

Ronald M Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. *Formal Issues in Lexical-Functional Grammar* pages 29–130.

Karen Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of English verbs. *Language Resources and Evaluation* 42:21–40.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR* abs/1411.2539.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proc. of COLING*. pages 1459–1474.

Parisa Kordjamshidi, Steven Bethard, and Marie-Francine Moens. 2012. Semeval-2012 task 3: Spatial role labeling. In *In Proc. of *SEM*. pages 365–373.

Mikhail Kozhevnikov and Ivan Titov. 2013. Cross-lingual transfer of semantic role labeling models. In *Proc. of ACL*. pages 1190–1200.

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

Ronald Langacker. 2008. *Cognitive Grammar: A Basic Introduction*. Oxford University Press, Oxford.

Beth Levin and Malka Rappaport Hovav. 2005. *Argument realization*. Cambridge University Press.

Michael Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *TACL* 1:179–192.

Maria Liakata, Simone Teufel, Advaith Siddharthan, and Colin Batchelor. 2010. Corpora for the conceptualisation and zoning of scientific papers. In *Proc. of LREC*. pages 2054–2061.

Edward Loper, Szu-Ting Yi, and Martha Palmer. 2007. Combining lexical resources: Mapping between PropBank and VerbNet. In *Proc. of the 7th International Workshop on Computational Linguistics*.

Christopher Manning. 2006. Local textual inference: It's hard to circumscribe, but you know it when you see it—and nlp needs it. unpublished ms.

Mitch Marcus, Beatrice Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19:313–330.

Eleni Miltsakaki, Rashmi Prasad, Aravind K Joshi, and Bonnie L Webber. 2004. The penn discourse treebank. In *LREC*. pages 2237–2240.

Paramita Mirza, Rachele Sprugnoli, Sara Tonelli, and Manuela Speranza. 2014. Annotating causality in the tempeval-3 corpus. In *Proc. of the EACL Workshop on Computational Approaches to Causality in Language (CAtoCL)*. pages 10–19.

Yusuke Miyao. 2006. *Corpus-oriented grammar development and feature forest model*. Ph.D. thesis, University of Tokyo.

Marc Moens and Mark Steedman. 1988. Temporal ontology and temporal reference. *Computational Linguistics* 14:15–28. Reprinted in Inderjeet Mani, James Pustejovsky, and Robert Gaizauskas (eds.) *The Language of Time: A Reader*. Oxford University Press, 93-114.

Richard Montague. 1970. English as a formal language. In Bruno Visentini, editor, *Linguaggi nella Società e nella Technica*, Edizioni di Communità, Milan, pages 189–224. Reprinted as Thomason 1974:188-221.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. Caters: Causal and temporal relation scheme for semantic annotation of event structures. In *Proc. of the Fourth Workshop on Events*. pages 51–61.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proc. of LREC*. pages 1659–1666.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Chris Manning. 2004. Lingo Redwoods. *Research on Language & Computation* 2:575–596.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*. pages 915–926.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*. pages 63–72.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection of semantic roles. *Journal of Artificial Intelligence Research* 36:307–340.

Alexis Palmer, Elias Ponvert, Jason Baldridge, and Carlota Smith. 2007. A sequencing model for situation entity classification. In *Proc. of ACL*. pages 896–903.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1):71–106.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. *Semantic Role Labeling*. Synthesis lectures on human language technologies. Morgan & Claypool Publishers.

Martha Palmer, Ivan Titov, and Shumin Wu. 2013. Semantic role labeling tutorial at naacl 2013. http://ivan-titov.org/teaching/srl-tutorial-naacl13/.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

James Pustejovsky, José Casteño, Robert Ingria, Roser Saurí, Robert Gaizauiuskas, Andrea Setzer, Graham Katz, and Dragomir Radev. 2003. Timeml: Robust specification of event and temporal expressions in text. In *Proc. of the 5th International Workshop on Computational Semantics*.

James Pustejovsky, Parisa Kordjamshidi, Marie-Francine Moens, Aaron Levine, Seth Dworman, and Zachary Yocum. 2015. Semeval-2015 task 8: Spaceeval. In *Proc. of SemEval*. pages 884–894.

Janarthanan Rajendran, Mitesh M. Khapra, Sarath Chandar, and Balaraman Ravindran. 2015. Bridge correlational neural networks for multilingual multimodal representation learning. *CoRR* abs/1510.03519.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *TACL* 4:127–140.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proc. of ACL*. pages 979–988.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *TACL* 3:475–488.

Michael Roth and Anette Frank. 2015. Inducing implicit arguments from comparable texts: A framework and its applications. *Computational Linguistics* 41:625–664.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, Collin F. Baker, and Jan Scheffczyk. 2016. *FrameNet II: Extended Theory and Practice*. The Berkeley FrameNet Project.

Nathan Schneider, Vivek Srikumar, Jena D. Hwang, and Martha Palmer. 2015. A hierarchy with, of, and for preposition supersenses. In *Proc. of LAW*. pages 112–123.

Petr Sgall. 1992. Underlying Structure of Sentences and Its Relations to Semantics. In T. Reuthe, editor, *Wiener Slawistischer Almanach. Sonderband 33*, Wien: Gesellschaft zur Förderung slawistischer Studien, pages 273–282.

Eric Siegel and Kathy McKeown. 2000. Learning methods to combine linguistic indicators: Improving aspectual classification and revealing linguistic insights. *Computational Linguistics* 26:595–628.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proc. of ACL*. pages 455–465.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Luc Steels and Joachim de Beules. 2006. A (very) brief introduction to fluid construction grammar. In *Proc. of the 3rd Workshop on Scalable Natural Language Understanding*. pages 73–80.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations: A French-English case study. In *ACL 2015 Workshop on Semantics-Driven Statistical Machine Translation (S2MT)*. pages 11–22.

Lin Sun, Anna Korhonen, Thierry Poibeau, and Cédric Messiant. 2010. Investigating the cross-linguistic potential of verbnet: style classification. In *Proc. of COLING*. pages 1056–1064.

Richmond Thomason, editor. 1974. *Formal Philosophy: Papers of Richard Montague*. Yale University Press, New Haven, CT.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In **SEM-SemEval '13*. pages 1–9.

Jan van Eijck. 2005. Natural logic for natural language. In Balder ten Cate and Henk Zeevat, editors, *Logic, Language, and Computation*. Springer, Berlin, Lecture Notes in Computer Science 4363, pages 216–230.

Marc Verhagen, Roser Sauri, Tomasso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proc. of the 5th International Workshop on Semantic Evaluation*. ACL, pages 57–62.

Mark Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowitcz, and James Pustejovsky. 2009. The tempeval challenge: Identifying temporal relations in text. *Language Resources and Evaluation* 43:161–179.

Bonnie Webber, Markus Egg, and Valia Kordoni. 2011. Discourse structure and language technology. *Natural Language Engineering* 18(4):437–490.

Bonnie Webber and Aravind Joshi. 2012. Discourse structure and computation: Past, present and future. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*. pages 42–54.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proc. of EMNLP*. pages 1713–1723.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering* 11(02):207–238.

Nianwen Xue, Odrej Bojar, Jan Hajic, Martha Palmer, Zdenka Uresova, and Xiuhong Zhang. 2014. Not an intelingua, but close: comparison of English AMRs to Chinese and Czech. In *Proc. of LREC*. pages 1765–1772.

John Zelle and Ray Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proc. of the 14th National Conference on Artificial Intelligence*. pages 1050–1055.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with Probabilistic Categorial Grammars. In *Proc. of UAI*. pages 658–666.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proc. of EMNLP-CoNLL*. pages 678–687.

Joint Learning for Event Coreference Resolution

Jing Lu and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{ljwinnie,vince}@hlt.utdallas.edu

Abstract

While joint models have been developed for many NLP tasks, the vast majority of event coreference resolvers, including the top-performing resolvers competing in the recent TAC KBP 2016 Event Nugget Detection and Coreference task, are pipeline-based, where the propagation of errors from the trigger detection component to the event coreference component is a major performance limiting factor. To address this problem, we propose a model for jointly learning event coreference, trigger detection, and event anaphoricity. Our joint model is novel in its choice of tasks and its features for capturing cross-task interactions. To our knowledge, this is the first attempt to train a mention-ranking model and employ event anaphoricity for event coreference. Our model achieves the best results to date on the KBP 2016 English and Chinese datasets.

1 Introduction

Within-document event coreference resolution is the task of determining which event mentions in a text refer to the same real-world event. Compared to entity coreference resolution, event coreference resolution is not only much less studied, but it is arguably more challenging. The challenge stems in part from the fact that an event coreference resolver typically lies towards the end of the standard information extraction pipeline, assuming as input the noisy outputs of its upstream components. One such component is the trigger detection system, which is responsible for identifying event triggers and determining their event subtypes.

As is commonly known, trigger detection is another challenging task that is far from being solved. In fact, in the recent TAC KBP 2016 Event Nugget Detection and Coreference task, trigger detection (a.k.a. event nugget detection in KBP) is deliberately made more challenging by focusing only on detecting the 18 subtypes of triggers on which the KBP 2015 participating systems' performances were the poorest (Mitamura et al., 2016). The best-performing KBP 2016 system on English trigger detection achieved only an F-score of 47 (Lu and Ng, 2016).[1]

Given the difficulty of trigger detection, it is conceivable that many errors will propagate from the trigger detection component to the event coreference component in any pipeline architecture where trigger detection precedes event coreference resolution. These trigger detection errors could severely harm event coreference performance. For instance, two event mentions could be wrongly posited as coreferent if the underlying triggers were wrongly predicted to have the same subtype. Nevertheless, the top-performing systems in the KBP 2016 event coreference task all adopted the aforementioned pipeline architecture (Liu et al., 2016; Lu and Ng, 2016; Nguyen et al., 2016). Their performances are not particularly impressive, however: the best English event coreference F-score (averaged over four scoring metrics) is only around 30%.

To address this error propagation problem, we describe a joint model of trigger detection, event coreference, and event anaphoricity in this paper. Our choice of these three tasks is motivated in part by their inter-dependencies. As mentioned above, it is well-known that trigger detection performance has a huge impact on event coreference performance. Though largely under-investigated, event coreference could also improve

[1] This is the best English nugget *type* result in KBP 2016. In this paper, we will not be concerned with *realis* classification, as it does not play any role in event coreference.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 90–101
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1009

trigger detection. For instance, if two event mentions are posited as coreferent, then the underlying triggers must have the same event subtype. While the use of anaphoricity information for entity coreference has been extensively studied (see Ng (2010)), to our knowledge there has thus far been no attempt to explicitly model event anaphoricity for event coreference.[2] Although the mention-ranking model we employ for event coreference also allows an event mention to be posited as non-anaphoric (by resolving it to a *null* candidate antecedent), our decision to train a separate anaphoricity model and integrate it into our joint model is motivated in part by the recent successes of Wiseman et al. (2015), who showed that there are benefits in jointly training a noun phrase anaphoricity model and a mention-ranking model for entity coreference resolution. Finally, event anaphoricity and trigger detection can also mutually benefit each other. For instance, any verb posited as a non-trigger cannot be anaphoric, and any verb posited as anaphoric must be a trigger. Note that in our joint model, anaphoricity serves as an *auxiliary* task: its intended use is to improve trigger detection and event coreference, potentially mediating the interaction between trigger detection and event coreference.

Being a structured conditional random field, our model encompasses two types of factors. Unary factors encode the features specific for each task. Binary and ternary factors capture the interaction between each pair of tasks in a soft manner, enabling the learner to learn which combinations of values of the output variables are more probable. For instance, the learner should learn that it is not a good idea to classify a verb both as anaphoric and as a non-trigger. Our model is similar in spirit to Durrett and Klein's (2014) joint model for *entity* analysis, which performs joint learning for entity coreference, entity linking and semantic typing via the use of interaction features.

Our contributions are two-fold. First, we present a joint model of event coreference, trigger detection, and anaphoricity that is novel in terms of the choice of tasks and the features used to capture cross-task interactions. Second, our model achieves the best results to date on the KBP 2016 English and Chinese event coreference tasks.

[2]Following the entity coreference literature, we overload the term *anaphoricity*, saying that an event mention is anaphoric if it is coreferent with a preceding mention in the associated text.

2 Definitions, Task, and Corpora

2.1 Definitions

We employ the following definitions in our discussion of trigger detection and event coreference:

- An **event mention** is an explicit occurrence of an event consisting of a textual trigger, arguments or participants (if any), and the event type/subtype.
- An **event trigger** is a string of text that most clearly expresses the occurrence of an event, usually a word or a multi-word phrase
- An **event argument** is an argument filler that plays a certain role in an event.
- An **event coreference chain** (a.k.a. an **event hopper**) is a group of event mentions that refer to the same real-world event. They must have the same event (sub)type.

To understand these definitions, consider the example in Table 1, which contains two coreferent event mentions, $ev1$ and $ev2$. *left* is the trigger for $ev1$ and *departed* is the trigger for $ev2$. Both triggers have subtype Movement.Transport-Person. $ev1$ has three arguments, *Georges Cipriani*, *prison*, and *Wednesday* with roles *Person*, *Origin*, and *Time* respectively. $ev2$ also has three arguments, *He*, *Ensisheim*, and *police vehicle* with roles *Person*, *Origin*, and *Instrument* respectively.

2.2 Task

The version of the event coreference task we focus on in this paper is the Event Nugget Detection and Coreference task in the TAC KBP 2016 Event Track. While we discuss the role played by event arguments in event coreference in the previous subsection, KBP 2016 addresses event argument detection as a separate shared task. In other words, the KBP 2016 Event Nugget Detection and Coreference task focuses solely on trigger detection and event coreference.

It is worth mentioning that the KBP Event Nugget Detection and Coreference task, which started in 2015, aims to address a major weakness of the ACE 2005 event coreference task. Specifically, ACE 2005 adopts a strict notion of event identity, with which two event mentions were annotated as coreferent if and only if "they had the same agent(s), patient(s), time, and location" (Song et al., 2015), and their event attributes (polarity, modality, genericity, and tense) were not incompatible. In contrast, KBP adopts a more relaxed definition of event coreference, allowing two

Georges Cipriani$_{[Person]}$, {left}$_{ev1}$ the prison$_{[Origin]}$ in Ensisheim in northern France on parole on Wednesday$_{[Time]}$. He$_{[Person]}$ {departed}$_{ev2}$ Ensisheim$_{[Origin]}$ in a police vehicle$_{[Instrument]}$ bound for an open prison near Strasbourg.

Table 1: Event coreference resolution example.

event mentions to be coreferent as long as they *intuitively* refer to the same real-world event. Under this definition, two event mentions can be coreferent even if their time and location arguments are not coreferent. In our example in Table 1, $ev1$ and $ev2$ are coreferent in KBP because they both refer to the same event of Cipriani leaving the prison. However, they are *not* coreferent in ACE because their *Origin* arguments are not coreferent (one Origin argument involves a prison in Ensisheim while the other involves the city Ensisheim).

2.3 Corpora

Given our focus on the KBP 2016 Event Nugget Detection and Coreference task, we employ the English and Chinese corpora used in this task for evaluation, referring to these corpora as the KBP 2016 English and Chinese corpora for brevity. There are no official training sets: the task organizers simply made available a number of event coreference-annotated corpora for training. For English, we use LDC2015E29, E68, E73, and E94 for training. These corpora are composed of two types of documents, newswire documents and discussion forum documents. Together they contain 648 documents with 18739 event mentions distributed over 9955 event coreference chains. For Chinese, we use LDC2015E78, E105, and E112 for training. These corpora are composed of discussion forum documents only. Together they contain 383 documents with 4870 event mentions distributed over 3614 event coreference chains.

The test set for English consists of 169 newswire and discussion forum documents with 4155 event mentions distributed over 3191 event coreference chains. The test set for Chinese consists of 167 newswire and discussion forum documents with 2518 event mentions distributed over 1912 event coreference chains. Note that these test sets contain only annotations for event triggers and event coreference (i.e., there are no event argument annotations). While some of the training sets additionally contain event argument annotations, we do not make use of event argument annotations in model training to ensure a fairer comparison to the teams participating in the KBP 2016 Event Nugget Detection and Coreference task.

3 Model

3.1 Overview

Our model, which is a structured conditional random field, operates at the document level. Specifically, given a test document, we first extract from it (1) all single-word nouns and verbs and (2) all words and phrases that have appeared at least once as a trigger in the training data. We treat each of these extracted words and phrases as a candidate event mention.[3] The goal of the model is to make joint predictions for the candidate event mentions in a document. Three predictions will be made for each candidate event mention that correspond to the three tasks in the model: its trigger subtype, its anaphoricity, and its antecedent.

Given this formulation, we define three types of output variables:

- Event subtype variables $\mathbf{t} = (t_1, \ldots, t_n)$. Each t_i takes a value in the set of 18 event subtypes defined in KBP 2016 or NONE, which indicates that the event mention is not a trigger.
- Anaphoricity variables $\mathbf{a} = (a_1, \ldots, a_n)$. Each a_i is either ANAPHORIC or NOT ANAPHORIC.
- Coreference variables $\mathbf{c} = (c_1, \ldots, c_n)$, where $c_i \in \{1, \ldots, i-1, \text{NEW}\}$. In other words, the value of each c_i is the id of its antecedent, which can be one of the preceding event mentions or NEW (if the event mention underlying c_i starts a new cluster).

Each candidate event mention is associated with exactly one coreference variable, one event subtype variable, and one anaphoricity variable. Our model induces the following log-linear probability distribution over these variables:

$$p(\mathbf{t}, \mathbf{a}, \mathbf{c}|x; \Theta) \propto \exp(\sum_i \theta_i f_i(\mathbf{t}, \mathbf{a}, \mathbf{c}, x))$$

[3] According to the KBP annotation guidelines, each word may trigger multiple event mentions (e.g., *murder* can trigger two event mentions with subtypes Life.Die and Conflict.Attack). Hence, our treating each extracted word as a candidate event mention effectively prevents a word from triggering multiple event mentions. Rather than complicate model design by relaxing this simplifying assumption, we present an alternative, though partial, solution to this problem wherein we allow each event mention to be associated with multiple event subtypes. See the Appendix for details.

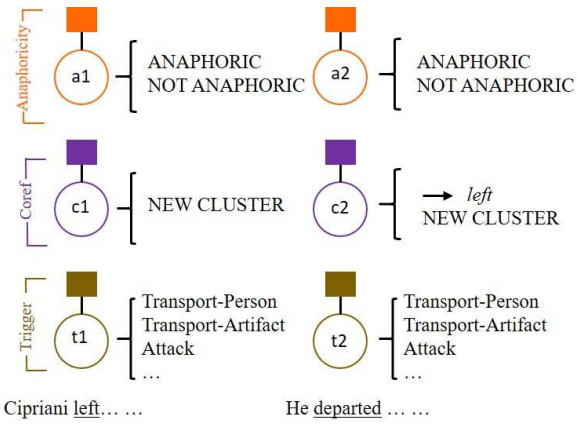


Figure 1: Unary factors for the three tasks, the variables they are connected to, and the possible values of the variables. Unary factors encode task-specific features. Each factor is connected to the corresponding output node. The features associated with a factor are used to predict the value of the output node it is connected to when a model is run independently of other models.

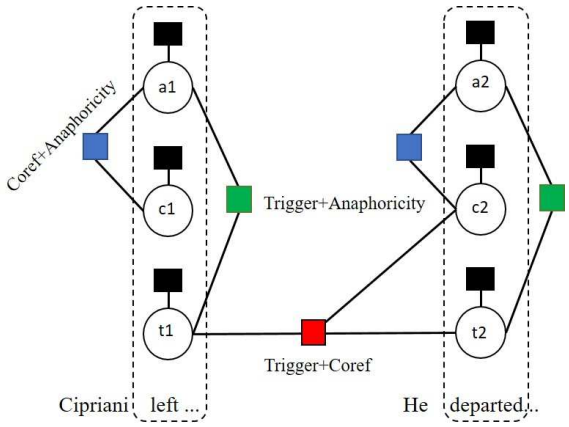


Figure 2: Binary and ternary factors. These higher-order factors capture cross-task interactions. The binary anaphoricity and trigger factors encourage anaphoric mentions to be triggers. The binary anaphoricity and coreference factors encourage non-anaphoric mentions to start a NEW coreference cluster. The ternary trigger and coreference factors encourage coreferent mentions to be triggers.

where $\theta_i \in \Theta$ is the weight associated with feature function f_i and x is the input document.

3.2 Features

Given that our model is a structured conditional random field, the features can be divided into two types: (1) task-specific features, and (2) cross-task features, which capture the interactions between a pair of tasks. We express these two types of features in factor graph notation. The task-specific features are encoded in unary factors, each of which is connected to the corresponding variable (Figure 1). The cross-task features are encoded in binary or ternary factors, each of which couples the output variables from two tasks (Figure 2). Next, we describe these two types of features. Each feature is used to train models for both English and Chinese unless otherwise stated.

3.2.1 Task-Specific Features

We begin by describing the task-specific features, which are encoded in unary factors, as well as each of the three independent models.

3.2.1.1 Trigger Detection

When applied in isolation, our trigger detection model returns a distribution over possible subtypes given a candidate trigger. Each candidate trigger t is represented using t's word, t's lemma, word bigrams formed with a window size of three from t, as well as feature conjunctions created by pairing t's lemma with each of the following features: the head word of the entity syntactically closest to t, the head word of the entity textually closest to t, the entity type of the entity that is syntactically closest to t, and the entity type of the entity that is textually closest to t.[4] In addition, for event mentions with verb triggers, we use the head words and the entity types of their subjects and objects as features, where the subjects and objects are extracted from the dependency parse trees obtained using Stanford CoreNLP (Manning et al., 2014). For event mentions with noun triggers, we create the same features that we did for verb triggers, except that we replace the subjects and verbs with heuristically extracted agents and patients. Finally, for the Chinese trigger detector, we additionally create two features from each character in t, one encoding the character itself and the other encoding the entry number of the corresponding character in a Chinese synonym dictionary.[5]

3.2.1.2 Event Coreference

We employ a mention-ranking model for event coreference that selects the most probable antecedent for a mention to be resolved (or NEW if the mention is non-anaphoric) from its set of candidate antecedents. When applied in isolation, the model is trained to maximize the condi-

[4] We train a CRF-based entity extraction model for jointly identifying the entity mentions and their types. Details can be found in Lu et al. (2016).

[5] The dictionary is available from http://ir.hit.edu.cn/. An entry number in this dictionary conceptually resembles a synset id in WordNet (Fellbaum, 1998).

tional likelihood of collectively resolving the mentions to their correct antecedents in the training texts (Durrett and Klein, 2013). Below we describe the features used to represent the candidate antecedents for the mention to be resolved, m_j.

Features representing the NULL candidate antecedent: Besides m_j's word and m_j's lemma, we employ feature conjunctions given their usefulness in entity coreference (Fernandes et al., 2014). Specifically, we create a conjunction between m_j's lemma and the number of sentences preceding m_j, as well as a conjunction between m_j's lemma and the number of mentions preceding m_j in the document.

Features representing a non-NULL candidate antecedent, m_i: m_i's word, m_i's lemma, whether m_i and m_j have the same lemma, and feature conjunctions including: (1) m_i's word paired with m_j's word, (2) m_i's lemma paired with m_j's lemma, (3) the sentence distance between m_i and m_j paired with m_i's lemma and m_j's lemma, (4) the mention distance between m_i and m_j paired with m_i's lemma and m_j's lemma, (5) a quadruple consisting of m_i and m_j's subjects and their lemmas, and (6) a quadruple consisting of m_i and m_j's objects and their lemmas.

3.2.1.3 Anaphoricity Determination

When used in isolation, the anaphoricity model returns the probability that the given event mention is anaphoric. To train the model, we represent each event mention m_j using the following features: (1) the head word of each candidate antecedent paired with m_j's word, (2) whether at least one candidate antecedent has the same lemma as that of m_j, and (3) the probability that m_j is anaphoric in the training data (if m_j never appears in the training data, this probability is set to 0.5).

3.2.2 Cross-Task Interaction Features

Cross-task interaction features are associated with the binary and ternary factors.

3.2.2.1 Trigger Detection and Anaphoricity

We fire features that conjoin each candidate event mention's event subtype, the lemma of its trigger and its anaphoricity.

3.2.2.2 Trigger Detection and Coreference

We define our joint coreference and trigger detection factors such that the features defined on subtype variables t_i and t_j are fired only if current mention m_j is coreferent with preceding mention m_i. These features are: (1) the pair of m_i and m_j's subtypes, (2) the pair of m_j's subtype and m_i's word, and (3) the pair of m_i's subtype and m_j's word.

3.2.2.3 Coreference and Anaphoricity

We fire a feature that conjoins event mention m_j's anaphoricity and whether or not a non-NULL antecedent is selected for m_j.

3.3 Training

We learn the model parameters Θ from a set of d training documents, where document i contains content x_i, gold triggers $\mathbf{t_i^*}$ and gold event coreference partition C_i^*. Before learning, there are a couple of issues we need to address.

First, we need to derive gold anaphoricity labels $\mathbf{a_i^*}$ from C_i^*. This is straightforward: the first mention of each coreference chain is NOT ANAPHORIC, whereas the rest are ANAPHORIC.

Second, we employ gold event mentions for model training, but training models only on gold mentions is not sufficient: for instance, a trigger detector trained solely on gold mentions will not be able to classify a candidate event mention as NONE during testing. To address this issue, we additionally train the models on candidate event mentions corresponding to non-triggers. We create these candidate event mentions as follows. For each word w that appears as a true trigger at least once in the training data, we create a candidate event mention from each occurrence of w in the training data that is not annotated as a true trigger.

Third, since our model produces event coreference output in the form of an antecedent vector (with one antecedent per event mention), it needs to be trained on antecedent vectors. However, since the coreference annotation for each document i is provided in the form of a clustering C_i^*, we follow previous work on entity coreference resolution (Durrett and Klein, 2013): we sum over all antecedent structures $A(C_i^*)$ that are *consistent* with C_i^* (i.e., the first mention of a cluster has antecedent NEW, whereas each of the subsequent mentions can select any of the preceding mentions in the same cluster as its antecedent).

Next, we learn the model parameters to maximize the following conditional likelihood of the training data with L1 regularization:

$$L(\Theta) = \sum_{i=1}^{d} \log \sum_{\mathbf{c}^* \in A(C_i^*)} p'(\mathbf{t_i^*}, \mathbf{a_i^*}, \mathbf{c}^* | x_i; \Theta) + \lambda \|\Theta\|_1$$

In this objective, p' is obtained by augmenting the distribution p (defined in Section 3.1) with task-specific parameterized loss functions:

$$p'(\mathbf{t}, \mathbf{a}, \mathbf{c}|x_i; \Theta) \propto p(\mathbf{t}, \mathbf{a}, \mathbf{c}|x_i; \Theta) \exp[\alpha_t l_t(\mathbf{t}, \mathbf{t}^*) + \alpha_a l_a(\mathbf{a}, \mathbf{a}^*) + \alpha_c l_c(\mathbf{c}, C^*)]$$

where l_t, l_a and l_c are task-specific loss functions, and α_t, α_a and α_c are the associated weight parameters that specify the relative importance of the three tasks in the objective function.

Softmax-margin, the technique of integrating task-specific loss functions into the objective function, was introduced by Gimpel and Smith (2010) and subsequently used by Durrett and Klein (2013, 2014). By encoding task-specific knowledge, these loss functions can help train a model that places less probability mass on less desirable output configurations.

Our loss function for event coreference, l_c, is motivated by the one Durrett and Klein (2013) developed for entity coreference. It is a weighted sum of the counts of three error types:

$$l_c(\mathbf{c}, C^*) = \alpha_{c,FA} FA(\mathbf{c}, C^*) + \alpha_{c,FN} FN(\mathbf{c}, C^*) + \alpha_{c,WL} WL(\mathbf{c}, C^*)$$

where $FA(\mathbf{c}, C^*)$ is the number of non-anaphoric mentions misclassified as anaphoric, $FN(\mathbf{c}, C^*)$ is the number of anaphoric mentions misclassified as non-anaphoric, and $WL(\mathbf{c}, C^*)$ is the number of incorrectly resolved anaphoric mentions.

Our loss function for trigger detection, l_t, is parameterized in a similar way, having three parameters associated with three error types: $\alpha_{t,FT}$ is associated with the number of non-triggers misclassified as triggers, $\alpha_{t,FN}$ is associated with the number of triggers misclassified as non-triggers, and $\alpha_{t,WL}$ is associated with the number of triggers labeled with the wrong subtype.

Finally, our loss function for anaphoricity determination, l_a, is also similarly parameterized, having two parameters: $\alpha_{a,FA}$ and $\alpha_{a,FN}$ are associated with the number of false anaphors and the number of false non-anaphors, respectively.

Following Durrett and Klein (2014), we use AdaGrad (Duchi et al., 2011) to optimize our objective with $\lambda = 0.001$ in our experiments.

3.4 Inference

Inference, which is performed during training and decoding, involves computing the marginals for a variable or a set of variables to which a factor connects. For efficiency, we perform approximate inference using belief propagation rather than exact inference. Given that convergence can typically be reached within five iterations of belief propagation, we employ five iterations in all experiments.

Performing inference using belief propagation on the full factor graph defined in Section 3.1 can still be computationally expensive, however. One reason is that the number of ternary factors grows quadratically with the number of event mentions in a document. To improve scalability, we restrict the domains of the coreference variables. Rather than allow the domain of coreference variable c_j to be of size j, we allow a preceding mention m_i to be a candidate antecedent of mention m_j if (1) the sentence distance between the two mentions is less than an empirically determined threshold and (2) either they are coreferent at least once in the training data or their head words have the same lemma. Doing so effectively enables us to prune the unlikely candidate antecedents for each event mention. As Durrett and Klein (2014) point out, such pruning has the additional benefit of reducing "the memory footprint and time needed to build a factor graph", as we do not need to create any factor between m_i and m_j and its associated features if m_i is pruned. To further reduce the memory footprint, we additionally restrict the domains of the event subtype variables. Given a candidate event mention created from word w, we allow the domain of its subtype variable to include only NONE as well as those subtypes that w is labeled with at least once in the training data.

For decoding, we employ minimum Bayes risk, which computes the marginals of each variable w.r.t. the joint model and derives the most probable assignment to each variable.

4 Evaluation

4.1 Experimental Setup

We perform training and evaluation on the KBP 2016 English and Chinese corpora. For English, we train models on 509 of the training documents, tune parameters on 139 training documents, and report results on the official KBP 2016 English test set.[6] For Chinese, we train models on 302 of the training documents, tune parameters on 81 training documents, and report results on the official

[6]The parameters to be tuned are the α's multiplying the loss functions and those inside the loss functions.

	English						
	MUC	B^3	CEAF_e	BLANC	AVG-F	Trigger	Anaphoricity
KBP2016	26.37	37.49	34.21	22.25	30.08	46.99	–
INDEP.	22.71	40.72	39.00	22.71	31.28	48.82	27.35
JOINT	27.41	40.90	39.00	25.00	33.08	49.30	31.94
Δ over INDEP.	+4.70	+0.18	0.00	+2.29	+1.80	+0.48	+4.59
	Chinese						
	MUC	B^3	CEAF_e	BLANC	AVG-F	Trigger	Anaphoricity
KBP2016	24.27	32.83	30.82	17.80	26.43	40.01	–
INDEP.	22.68	32.97	29.96	17.74	25.84	39.82	19.31
JOINT	27.94	33.01	29.96	20.24	27.79	40.53	23.33
Δ over INDEP.	+5.26	+0.04	0.00	+2.50	+1.95	+0.71	+4.02

Table 2: Results of all three tasks on the KBP 2016 evaluation sets. The KBP2016 results are those achieved by the best-performing coreference resolver in the official KBP 2016 evaluation. Δ is the performance difference between the JOINT model and the corresponding INDEP. model. All results are expressed in terms of F-score.

KBP 2016 Chinese test set.

Results of event coreference and trigger detection are obtained using version 1.7.2 of the official scorer provided by the KBP 2016 organizers. To evaluate event coreference performance, the scorer employs four scoring measures, namely MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF_e (Luo, 2005) and BLANC (Recasens and Hovy, 2011), as well as the unweighted average of their F-scores (AVG-F). The scorer reports event mention detection performance in terms of F-score, considering a mention correctly detected if it has an exact match with a gold mention in terms of boundary, event type, and event subtype. In addition, we report anaphoricity determination performance in terms of the F-score computed over anaphoric mentions, counting an extracted anaphoric mention as a true positive if it has an exact match with a gold anaphoric mention in terms of boundary.

4.2 Results and Discussion

Results are shown in Table 2 where performance on all three tasks (event coreference, trigger detection, and anaphoricity determination) is expressed in terms of F-score. The top half of the table shows the results on the English evaluation set. Specifically, row 1 shows the performance of the best event coreference system participating in KBP 2016 (Lu and Ng, 2016). This system adopts a pipeline architecture. It first uses an ensemble of one-nearest-neighbor classifiers for trigger detection. Using the extracted triggers, it then applies a pipeline of three sieves, each of which is a one-nearest-neighbor classifier, for event coreference. As we can see, this system achieves an AVG-F of 30.08 for event coreference and an F-score of 46.99 for trigger detection.

Row 2 shows the performance of the independent models, each of which is trained independently of the other models. Specifically, each independent model is trained using only the unary factors associated with it. As we can see, the independent models outperform the top KBP 2016 system by 1.2 points in AVG-F for event coreference and 1.83 points for trigger detection.

Results of our joint model are shown in row 3. The absolute performance differences between the joint model and the independent models are shown in row 4. As we can see, the joint model outperforms the independent models for all three tasks: by 1.80 points for event coreference, 0.48 points for trigger detection and 4.59 points for anaphoricity determination. Most encouragingly, the joint model outperforms the top KBP 2016 system for both event coreference and trigger detection. For event coreference, it outperforms the top KBP system w.r.t. all scoring metrics, yielding an improvement of 3 points in AVG-F. For trigger detection, it outperforms the top KBP system by 2.31 points.

The bottom half of Table 2 shows the results on the Chinese evaluation set. The top KBP 2016 event coreference system on Chinese is also the Lu and Ng (2016) system. While the top KBP system outperforms the independent models for both tasks (by 0.59 points in AVG-F for event coreference and 0.19 points for trigger detection), our joint model outperforms the independent models

	English			Chinese		
	Coref	Trigger	Anaph	Coref	Trigger	Anaph
INDEP.	31.28	48.82	27.35	25.84	39.82	19.31
INDEP.+CorefTrigger	+0.39	+0.42	−0.05	+0.95	+0.56	−0.37
INDEP.+CorefAnaph	+0.40	−0.08	+3.45	+0.37	+0.04	−0.11
INDEP.+TriggerAnaph	+0.11	+0.38	+1.35	+0.14	+0.52	+0.02
JOINT−CorefTrigger	+0.56	−0.06	+4.41	+0.19	+0.35	+3.34
JOINT−CorefAnaph	+0.63	+0.66	+1.46	+1.50	+0.88	+0.28
JOINT−TriggerAnaph	+1.89	+0.50	+4.01	+1.65	+0.50	+1.79
JOINT	+1.80	+0.48	+4.59	+1.95	+0.71	+4.02

Table 3: Results of model ablations on the KBP 2016 evaluation sets. Each row of ablation results is obtained by either adding one type of interaction factor to the INDEP. model or deleting one type of interaction factor from the JOINT model. For each column, the results are expressed in terms of changes to the INDEP. model's F-score shown in row 1.

for all three tasks: by 1.95 points for event coreference, 4.02 points for anaphoricity determination, and 0.71 points for trigger detection. Like its English counterpart, our Chinese joint model outperforms the top KBP system for both event coreference and trigger detection. For event coreference, it outperforms the top KBP system w.r.t. all but the CEAF_e metric, yielding an absolute improvement of 1.36 points in AVG-F. For trigger detection, it outperforms the top KBP system by 0.52 points.

For both datasets, the joint model's superior performance to the independent coreference model stems primarily from considerable improvements in MUC F-score. As MUC is a link-based measure, these results provide suggestive evidence that joint modeling has enabled more event coreference links to be discovered.

4.3 Model Ablations

To evaluate the importance of each of the three types of joint factors in the joint model, we perform ablation experiments.[7] Table 3 shows the results on the English and Chinese datasets when we add each type of joint factors to the independent model and remove each type of joint factors from the full joint model. The results of each task are expressed in terms of changes to the corresponding independent model's F-score.

[7] Chen and Ng (2013) also performed ablation on their ACE-style Chinese event coreference resolver. However, given the differences in the tasks involved (e.g., they did not model event anaphoricity, but included tasks such as event argument extraction and role classification, entity coreference, and event mention attribute value computation) and the ablation setup (e.g., they ablated individual tasks/components in their pipeline-based system in an incremental fashion, whereas we ablate interaction factors rather than tasks), a direct comparison of their observations and ours is difficult.

Coref-Trigger interactions. Among the three types of factors, this one contributes the most to coreference performance, regardless of whether it is applied in isolation or in combination with the other two types of factors to the independent coreference model. In addition, it is the most effective type of factor for improving trigger detection. When applied in combination, it also improves anaphoricity determination, although less effectively than the other two types of factors.

Coref-Anaphoricity interactions. When applied in isolation to the independent models, this type of factor improves coreference resolution but has a mixed impact on anaphoricity determination. When applied in combination with other types of factors, it improves both tasks, particularly anaphoricity determination. Its impact on trigger detection, however, is generally negative.

Trigger-Anaphoricity interactions. When applied in isolation to the independent models, this type of factor improves both trigger detection and anaphoricity determination. When applied in combination with other types of factors, it still improves anaphoricity determination (particularly on Chinese), but has a mixed effect on trigger detection. Among the three types of factors, it has the least impact on coreference resolution.

4.4 Error Analysis

Next, we conduct an analysis of the major sources of error made by our joint coreference model.

4.4.1 Two Major Types of Precision Error

Erroneous and mistyped triggers. Our trigger model tends to assign the same subtype to event mentions triggered by the same word. As a result, it often assigns the wrong subtype to triggers that

possess different subtypes in different contexts. For the same reason, words that are only sometimes used as triggers are often wrongly posited as triggers when they are not. These two types of triggers have in turn led to the establishment of incorrect coreference links.[8]

Failure to extract arguments. In the absence of an annotated corpus for training an argument classifier, we exploit dependency relations for argument extraction. Doing so proves inadequate, particularly for noun triggers, owing to the absence of dependency relations that can be used to reliably extract their arguments. Moreover, using dependency relations does not allow the extraction of arguments that do not appear in the same sentence as their trigger. Since the presence of incompatible arguments is an important indicator of non-coreference, our model's failure to extract arguments has resulted in incorrect coreference links.

4.4.2 Three Major Types of Recall Error

Missing triggers. Our trigger model fails to identify trigger words that are unseen or rarely-occurring in the training data. As a result, many coreference links cannot be established.

Lack of entity coreference information. Entity coreference information is useful for event coreference because the corresponding arguments of two event mentions are typically coreferent. Since our model does not exploit entity coreference information, it treats two lexically different event arguments as non-coreferent/unrelated. This in turn weakens its ability to determine whether two event mentions are coreferent. This issue is particularly serious in discussion forum documents, where it is not uncommon to see pronouns serve as subjects and objects of event mentions. The situation is further aggravated in Chinese documents, where zero pronouns are prevalent.

Lack of contextual understanding. Our model only extracts features from the sentence in which an event mention appears. However, additional contextual information present in neighboring sentences may be needed for correct coreference resolution. This is particularly true in discussion forum documents, where the same event may be described differently by different people. For example, when describing the fact that Tim Cook will attend Apple's Istanbul store opening, one person said "Cook is expected to return to Turkey for the store opening", and another person described this event as "Tim travels abroad YET AGAIN to be feted by the not-so-high-and-mighty". It is by no means easy to determine that *return* and *travel* trigger two coreferent mentions in these sentences.

[8] In our joint model, mentions that are posited as coreferent are encouraged to have the same subtype. While it can potentially fix the errors involving coreferent mentions that have different subtypes, it cannot fix the errors in which the two mentions involved have the same erroneous subtype.

5 Related Work

Existing event coreference resolvers have been evaluated on different corpora, such as MUC (e.g., Humphreys et al. (1997)), ACE (e.g., Ahn (2006), Chen and Ji (2009), McConky et al. (2012), Sangeetha and Arock (2012), Chen and Ng (2015, 2016), Krause et al. (2016)), OntoNotes (e.g., Chen et al. (2011)), the Intelligence Community corpus (e.g., Cybulska and Vossen (2012), Araki et al. (2014), Liu et al. (2014)), the ECB corpus (e.g., Lee et al. (2012), Bejan and Harabagiu (2014)) and its extension ECB+ (e.g., Yang et al. (2015)), and ProcessBank (e.g., Araki and Mitamura (2015)). The newest event coreference corpora are the ones used in the KBP 2015 and 2016 Event Nugget Detection and Coreference shared tasks, in which the best performers in 2015 and 2016 are RPI's system (Hong et al., 2015) and UTD's system (Lu and Ng, 2016), respectively. The KBP 2015 corpus has recently been used to evaluate Peng et al.'s (2016) minimally supervised approach and Lu et al.'s (2016) joint inference approach to event coreference. With the rarest exceptions (e.g., Lu et al. (2016)), existing resolvers have adopted a pipeline architecture in which trigger detection is performed prior to coreference resolution.

6 Conclusion

We proposed a joint model of event coreference resolution, trigger detection, and event anaphoricity determination. The model is novel in its choice of tasks and the cross-task interaction features. When evaluated on the KBP 2016 English and Chinese corpora, our model not only outperforms the independent models but also achieves the best results to date on these corpora.

Acknowledgments

We thank the three anonymous reviewers for their detailed comments. This work was supported in part by NSF Grants IIS-1219142 and IIS-1528037.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the COLING/ACL Workshop on Annotating and Reasoning about Time and Events*. pages 1–8.

Jun Araki, Zhengzhong Liu, Eduard Hovy, and Teruko Mitamura. 2014. Detecting subevent structure for event coreference resolution. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 4553–4558.

Jun Araki and Teruko Mitamura. 2015. Joint event trigger identification and event coreference resolution with structured perceptron. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2074–2080.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the Linguistic Coreference Workshop at The First International Conference on Language Resources and Evaluation*, pages 563–566.

Cosmin Adrian Bejan and Sanda Harabagiu. 2014. Unsupervised event coreference resolution. *Computational Linguistics* 40(2):311–347.

Bin Chen, Jian Su, Sinno Jialin Pan, and Chew Lim Tan. 2011. A unified event coreference resolution by integrating multiple resolvers. In *Proceedings of the Fifth International Conference on Natural Language Processing*. pages 102–110.

Chen Chen and Vincent Ng. 2013. Chinese event coreference resolution: Understanding the state of the art. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*. pages 822–828.

Chen Chen and Vincent Ng. 2015. Chinese event coreference resolution: An unsupervised probabilistic model rivaling supervised resolvers. In *Proceedings of Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. pages 1097–1107.

Chen Chen and Vincent Ng. 2016. Joint inference over a lightly supervised information extraction pipeline: Towards event coreference resolution for resource-scarce languages. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. pages 2913–2920.

Zheng Chen and Heng Ji. 2009. Graph-based event coreference resolution. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing (TextGraphs-4)*, pages 54–57.

Agata Cybulska and Piek Vossen. 2012. Using semantic relations to solve event coreference in text. In *Proceedings of the LREC Workshop on Semantic Relations-II Enhancing Resources and Applications*, pages 60–67.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics* 2:477–490.

Christiane Fellbaum. 1998. *WordNet: An Electronical Lexical Database*. MIT Press, Cambridge, MA.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiu. 2014. Latent trees for coreference resolution. *Computational Linguistics* 40(4):801–835.

Kevin Gimpel and Noah A Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736.

Yu Hong, Di Lu, Dian Yu, Xiaoman Pan, Xiaobin Wang, Yadong Chen, Lifu Huang, and Heng Ji. 2015. RPI BLENDER TAC-KBP2015 system description. In *Proceedings of the Eighth Text Analysis Conference*.

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. 1997. Event coreference for information extraction. In *Proceedings of the ACL/EACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 75–81.

Sebastian Krause, Feiyu Xu, Hans Uszkoreit, and Dirk Weissenborn. 2016. Event linking with sentential features from convolutional neural networks. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 239–249.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500.

Zhengzhong Liu, Jun Araki, Eduard Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 4539–4544.

Zhengzhong Liu, Jun Araki, Teruko Mitamura, and Eduard Hovy. 2016. CMU-LTI at KBP 2016 event nugget track. In *Proceedings of the Ninth Text Analysis Conference*.

Jing Lu and Vincent Ng. 2016. UTD's event nugget detection and coreference system at KBP 2016. In *Proceedings of the Ninth Text Analysis Conference*.

Jing Lu, Deepak Venugopal, Vibhav Gogate, and Vincent Ng. 2016. Joint inference for event coreference resolution. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 3264–3275.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Katie McConky, Rakesh Nagi, Moises Sudit, and William Hughes. 2012. Improving event coreference by context extraction and dynamic feature weighting. In *Proceedings of the 2012 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, pages 38–43.

Teruko Mitamura, Zhengzhong Liu, and Eduard Hovy. 2016. Overview of TAC-KBP 2016 event nugget track. In *Proceedings of the Ninth Text Analysis Conference*.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. pages 1396–1411.

Thien Huu Nguyen, Adam Meyers, and Ralph Grishman. 2016. New York University 2016 system for KBP event nugget: A deep learning approach. In *Proceedings of Ninth Text Analysis Conference*.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 392–402.

Marta Recasens and Eduard Hovy. 2011. BLANC: Implementing the Rand Index for coreference evaluation. *Natural Language Engineering* 17(4):485–510.

S. Sangeetha and Michael Arock. 2012. Event coreference resolution using mincut based graph clustering. In *Proceedings of the Fourth International Workshop on Computer Networks & Communications* pages 253–260.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ERE: Annotation of entities, relations, and events. In *Proceedings of the 3rd Workshop on EVENTS*, pages 89–98.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference*, pages 45–52.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1416–1426.

Bishan Yang, Claire Cardie, and Peter Frazier. 2015. A hierarchical distance-dependent Bayesian model for event coreference resolution. *Transactions of the Association for Computational Linguistics* 3:517–528.

Appendix: Handling Words that Trigger Multiple Event Mentions

In KBP, a word can trigger multiple event mentions. However, since we create exactly one candidate event mention from each extracted word in each test document, our model effectively prevents a word from triggering multiple event mentions. This poses a problem: each word cannot be associated with more than one event subtype. This appendix describes how we (partially) address this issue that involves allowing each event mention to be associated with multiple event subtypes.

To address this problem, we preprocess the *gold trigger annotations* in the *training* data as follows. First, for each word triggering multiple event mentions (with different event subtypes), we merge their event mentions into one event mention having the combined subtype. In principle, we can add each of these combined subtypes into our event subtype inventory and allow our model to make predictions using them. However, to avoid over-complicating the prediction task (by having a large subtype inventory), we only add the three most frequently occurring combined subtypes in the training data to the inventory. Merged mentions whose combined subtype is not among the most frequent three will be unmerged in order to recover the original mentions so that the model can still be trained on them.

To train our joint model, however, the trigger annotations and the event coreference annotations in the training data must be consistent. Since we modified the trigger annotations (by merging event mentions and allowing combined subtypes), we make two modifications to the event coreference annotations to ensure consistency between the two sets of annotations. First, let C_1 and C_2 be two event coreference chains in a training document such that the set of words triggering the event mentions in C_1 (with subtype t_1) is the same as that triggering the event mentions in C_2 (with subtype t_2). If each of the event mentions in C_1 was merged with the corresponding event mention in C_2 during the aforementioned preprocessing of the trigger annotations (because combining t_1 and t_2 results in one of the three most frequent combined subtypes), then we delete one of the two coreference chains, and assign the combined subtype to the remaining chain. Finally, we remove any remaining event mentions that were merged during the preprocessing of trigger annotations from their respective coreference chains and create a singleton cluster for each of the merged mentions.

Generating and Exploiting Large-scale Pseudo Training Data for Zero Pronoun Resolution

Ting Liu[†], **Yiming Cui**[‡], **Qingyu Yin**[†], **Weinan Zhang**[†], **Shijin Wang**[‡] and **Guoping Hu**[‡]
[†]Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology, Harbin, China
[‡]iFLYTEK Research, Beijing, China
[†]{tliu,qyyin,wnzhang}@ir.hit.edu.cn
[‡]{ymcui,sjwang3,gphu}@iflytek.com

Abstract

Most existing approaches for zero pronoun resolution are heavily relying on annotated data, which is often released by shared task organizers. Therefore, the lack of annotated data becomes a major obstacle in the progress of zero pronoun resolution task. Also, it is expensive to spend manpower on labeling the data for better performance. To alleviate the problem above, in this paper, we propose a simple but novel approach to automatically generate large-scale pseudo training data for zero pronoun resolution. Furthermore, we successfully transfer the cloze-style reading comprehension neural network model into zero pronoun resolution task and propose a two-step training mechanism to overcome the gap between the pseudo training data and the real one. Experimental results show that the proposed approach significantly outperforms the state-of-the-art systems with an absolute improvements of 3.1% F-score on OntoNotes 5.0 data.

1 Introduction

Previous works on zero pronoun (ZP) resolution mainly focused on the supervised learning approaches (Han, 2006; Zhao and Ng, 2007; Iida et al., 2007; Kong and Zhou, 2010; Iida and Poesio, 2011; Chen and Ng, 2013). However, a major obstacle for training the supervised learning models for ZP resolution is the lack of annotated data. An important step is to organize the shared task on anaphora and coreference resolution, such as the ACE evaluations, SemEval-2010 shared task on Coreference Resolution in Multiple Languages (Marta Recasens, 2010) and CoNLL-2012 shared task on Modeling Multilingual Unrestricted Coreference in OntoNotes (Sameer Pradhan, 2012). Following these shared tasks, the annotated evaluation data can be released for the following researches. Despite the success and contributions of these shared tasks, it still faces the challenge of spending manpower on labeling the extended data for better training performance and domain adaptation.

To address the problem above, in this paper, we propose a simple but novel approach to automatically generate large-scale pseudo training data for zero pronoun resolution. Inspired by data generation on cloze-style reading comprehension, we can treat the zero pronoun resolution task as a special case of reading comprehension problem. So we can adopt similar data generation methods of reading comprehension to the zero pronoun resolution task. For the noun or pronoun in the document, which has the frequency equal to or greater than 2, we randomly choose one position where the noun or pronoun is located on, and replace it with a specific symbol $\langle blank \rangle$. Let query $\mathcal{Q}$ and answer $\mathcal{A}$ denote the sentence that contains a $\langle blank \rangle$, and the noun or pronoun which is replaced by the $\langle blank \rangle$, respectively. Thus, a pseudo training sample can be represented as a triple:

$$\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle \quad (1)$$

For the zero pronoun resolution task, a $\langle blank \rangle$ represents a zero pronoun (ZP) in query $\mathcal{Q}$, and $\mathcal{A}$ indicates the corresponding antecedent of the ZP. In this way, tremendous pseudo training samples can be generated from the various documents, such as news corpus.

Towards the shortcomings of the previous approaches that are based on feature engineering, we propose a neural network architecture, which is an attention-based neural network model, for zero pronoun resolution. Also we propose a two-step

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 102–111
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1010

training method, which benefit from both large-scale pseudo training data and task-specific data, showing promising performance.

To sum up, the contributions of this paper are listed as follows.

- To our knowledge, this is the first time that utilizing reading comprehension neural network model into zero pronoun resolution task.
- We propose a two-step training approach, namely pre-training-then-adaptation, which benefits from both the large-scale automatically generated pseudo training data and task-specific data.
- Towards the shortcomings of the feature engineering approaches, we first propose an attention-based neural network model for zero pronoun resolution.

2 The Proposed Approach

In this section, we will describe our approach in detail. First, we will describe our method of generating large-scale pseudo training data for zero pronoun resolution. Then we will introduce two-step training approach to alleviate the gaps between pseudo and real training data. Finally, the attention-based neural network model as well as associated unknown words processing techniques will be described.

2.1 Generating Pseudo Training Data

In order to get large quantities of training data for neural network model, we propose an approach, which is inspired by (Hermann et al., 2015), to automatically generate large-scale pseudo training data for zero pronoun resolution. However, our approach is much more simple and general than that of (Hermann et al., 2015). We will introduce the details of generating the pseudo training data for zero pronoun resolution as follows.

First, we collect a large number of documents that are relevant (or homogenous in some sense) to the released OntoNote 5.0 data for zero pronoun resolution task in terms of its domain. In our experiments, we used large-scale news data for training.

Given a certain document $\mathcal{D}$, which is composed by a set of sentences $\mathcal{D} = \{s_1, s_2, ..., s_n\}$, we randomly choose an answer word $\mathcal{A}$ in the document. Note that, we restrict $\mathcal{A}$ to be either a noun or pronoun, where the part-of-speech is identified using LTP Toolkit (Che et al., 2010), as well as the answer word should appear at least twice in the document. Second, after the answer word $\mathcal{A}$ is chosen, the sentence that contains $\mathcal{A}$ is defined as a query $\mathcal{Q}$, in which the answer word $\mathcal{A}$ is replaced by a specific symbol $\langle blank \rangle$. In this way, given the query $\mathcal{Q}$ and document $\mathcal{D}$, the target of the prediction is to recover the answer $\mathcal{A}$. That is quite similar to the zero pronoun resolution task. Therefore, the automatically generated training samples is called ***pseudo*** training data. Figure 1 shows an example of a pseudo training sample.

In this way, we can generate tremendous triples of $\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$ for training neural network, without making any assumptions on the nature of the original corpus.

2.2 Two-step Training

It should be noted that, though we have generated large-scale pseudo training data for neural network training, there is still a gap between pseudo training data and the real zero pronoun resolution task in terms of the query style. So we should do some adaptations to our model to deal with the zero pronoun resolution problems ideally.

In this paper, we used an effective approach to deal with the mismatch between pseudo training data and zero pronoun resolution task-specific data. Generally speaking, in the first stage, we use a large amount of the pseudo training data to train a fundamental model, and choose the best model according to the validation accuracy. Then we continue to train from the previous best model using the zero pronoun resolution task-specific training data, which is exactly the same domain and query type as the standard zero pronoun resolution task data.

The using of the combination of proposed pseudo training data and task-specific data, i.e. zero pronoun resolution task data, is far more effective than using either of them alone. Though there is a gap between these two data, they share many similar characteristics to each other as illustrated in the previous part, so it is promising to utilize these two types of data together, which will compensate to each other.

The two-step training procedure can be concluded as,

Document:

1 ||| welcome both of you to the studio to participate in our program ,
欢迎 两 位 呢 来 演播室 参与 我们 的 节目 ,

2 ||| it happened that i was going to have lunch with a friend at noon .
正好 因为 我 也 和 朋友 这个 , 这个 中午 一起 吃饭 。

3 ||| after that , i received an sms from 1860 .
然后 我 就 收到 1860 的 短信 。

4 ||| uh-huh , it was by sms .
嗯 , 是 通过 短信 的 方式 ,

5 ||| uh-huh , that means , er , you knew about the accident through the source of radio station .
嗯 , 就是说 呃 你 是 通过 台 里面 的 一 个 信息 的 渠道 知道 这儿 出 了 这样 的 事故 。

6 ||| although we live in the west instead of the east part , and it did not affect us that much ,
虽然 我们 生活 在 西部 不 是 在 东部 , 对 我们 影响 不 是 很 大 ,

7 ||| but i think it is very useful to inform people using sms .
但是 呢 , 我 觉得 有 这样 一 个 短信 告诉 大家 呢 是 非常 有用 的 啊 。

Query:

8 ||| some car owners said that **<blank>** was very good。
有 车主 表示 , 说 这 **<blank>** 非常 的 好。

Answer:

sms
短信

Figure 1: Example of pseudo training sample for zero pronoun resolution system. (The original data is in Chinese, we translate this sample into English for clarity)

- Pre-training stage: by using large-scale training data to train the neural network model, we can learn richer word embeddings, as well as relatively reasonable weights in neural networks than just training with a small amount of zero pronoun resolution task training data;
- Adaptation stage: after getting the best model that is produced in the previous step, we continue to train the model with task-specific data, which can force the previous model to adapt to the new data, without losing much knowledge that has learned in the previous stage (such as word embeddings).

As we will see in the experiment section that the proposed two-step training approach is effective and brings significant improvements.

2.3 Attention-based Neural Network Model

Our model is primarily an attention-based neural network model, which is similar to *Attentive Reader* proposed by (Hermann et al., 2015). Formally, when given a set of training triple $\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$, we will construct our network in the following way.

Firstly, we project one-hot representation of document $\mathcal{D}$ and query $\mathcal{Q}$ into a continuous space with the shared embedding matrix W_e. Then we input these embeddings into different bi-directional RNN to get their contextual representations respectively. In our model, we used the bidirectional Gated Recurrent Unit (GRU) as RNN implementation (Cho et al., 2014).

$$e(x) = W_e \cdot x, \ where \ \ x \in \mathcal{D}, \mathcal{Q} \tag{2}$$

$$\overrightarrow{h_s} = \overrightarrow{GRU}(e(x)); \overleftarrow{h_s} = \overleftarrow{GRU}(e(x)) \tag{3}$$

$$h_s = [\overrightarrow{h_s}; \overleftarrow{h_s}] \tag{4}$$

For the query representation, instead of concatenating the final forward and backward states as its representations, we directly get an averaged representations on all bi-directional RNN slices, which can be illustrated as

$$h_{query} = \frac{1}{n} \sum_{t=1}^{n} h_{query}(t) \tag{5}$$

For the document, we place a soft attention over all words in document (Bahdanau et al., 2014), which indicate the degree to which part of document is attended when filling the blank in the query sentence. Then we calculate a weighted sum of all document tokens to get the attended representation of document.

$$m(t) = \tanh(W \cdot h_{doc}(t) + U \cdot h_{query}) \tag{6}$$

$$\alpha(t) = \frac{\exp(W_s \cdot m(t))}{\sum_{j=1}^{n} \exp(W_s \cdot m(j))} \tag{7}$$

$$h_{doc\_att} = h_{doc} \cdot \alpha \tag{8}$$

where variable $\alpha(t)$ is the normalized attention weight at tth word in document, h_{doc} is a matrix that concatenate all $h_{doc}(t)$ in sequence.

$$h_{doc} = concat[h_{doc}(1), h_{doc}(2), ..., h_{doc}(t)] \tag{9}$$

Then we use attended document representation and query representation to estimate the final answer, which can be illustrated as follows, where V

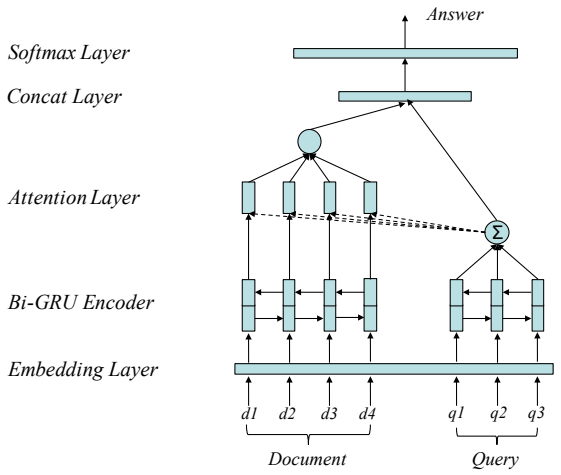


Figure 2: Architecture of attention-based neural network model for zero pronoun resolution task.

is the vocabulary,

$$r = concat[h_{doc\_att}, h_{query}] \tag{10}$$

$$P(\mathcal{A}|\mathcal{D}, \mathcal{Q}) \propto softmax(W_r \cdot r) \ \ , s.t. \ \mathcal{A} \in V \tag{11}$$

Figure 2 shows the proposed neural network architecture.

Note that, for zero pronoun resolution task, antecedents of zero pronouns are always noun phrases (NPs), while our model generates only one word (a noun or a pronoun) as the result. To better adapt our model to zero pronoun resolution task, we further process the output result in the following procedure. First, for a given zero pronoun, we extract a set of NPs as its candidates utilizing the same strategy as (Chen and Ng, 2015). Then, we use our model to generate an answer (one word) for the zero pronoun. After that, we go through all the candidates from the nearest to the far-most. For an NP candidate, if the produced answer is its head word, we then regard this NP as the antecedent of the given zero pronoun. By doing so, for a given zero pronoun, we generate an NP as the prediction of its antecedent.

2.4 Unknown Words Processing

Because of the restriction on both memory occupation and training time, it is usually suggested to use a shortlist of vocabulary in neural network training. However, we often replace the out-of-vocabularies to a unique special token, such as $\langle unk \rangle$. But this may place an obstacle in real world test. When the model predicts the answer as $\langle unk \rangle$, we do not know what is the exact word it represents in the document, as there may have many $\langle unk \rangle$s in the document.

In this paper, we propose to use a simple but effective way to handle unknown words issue. The idea is straightforward, which can be illustrated as follows.

- Identify all unknown words inside of each $\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$;
- Instead of replacing all these unknown words into one unique token $\langle unk \rangle$, we make a hash table to project these unique unknown words to numbered tokens, such as $\langle unk1 \rangle, \langle unk2 \rangle, ..., \langle unkN \rangle$ in terms of its occurrence order in the document. Note that, the same words are projected to the same unknown word tokens, and all these projections are only valid inside of current sample. For example, $\langle unk1 \rangle$ indicate the first unknown word, say "apple", in the current sample, but in another sample the $\langle unk1 \rangle$ may indicate the unknown word "orange". That is, the unknown word labels are indicating position features rather than the exact word;
- Insert these unknown marks in the vocabulary. These marks may only take up dozens of slots, which is negligible to the size of shortlists (usually 30K $\sim$ 100K).

(a) *The weather today is not as pleasant as the weather of yesterday.*
(b) *The **<unk>** today is not as **<unk>** as the **<unk>** of yesterday.*
(c) *The **<unk1>** today is not as **<unk2>** as the **<unk1>** of yesterday.*

Figure 3: Example of unknown words processing. a) original sentence; b) original unknown words processing method; c) our method

We take one sentence "The weather of today is not as pleasant as the weather of yesterday." as an example to show our unknown word processing method, which is shown in Figure 3.

If we do not discriminate the unknown words and assign different unknown words with the same token $\langle unk \rangle$, it would be impossible for us to know what is the exact word that $\langle unk \rangle$ represents for in the real test. However, when using our proposed unknown word processing method, if the model predicts a $\langle unkX \rangle$ as the answer,

we can simply scan through the original document and identify its position according to its unknown word number X and replace the $\langle unkX \rangle$ with the real word. For example, in Figure 3, if we adopt original unknown words processing method, we could not know whether the $\langle unk \rangle$ is the word "weather" or "pleasant". However, when using our approach, if the model predicts an answer as $\langle unk1 \rangle$, from the original text, we can know that $\langle unk1 \rangle$ represents the word "weather".

3 Experiments

3.1 Data

In our experiments, we choose a selection of public news data to generate large-scale pseudo training data for pre-training our neural network model (pre-training step)[1]. In the adaptation step, we used the official dataset OntoNotes Release 5.0[2] which is provided by CoNLL-2012 shared task, to carry out our experiments. The CoNLL-2012 shared task dataset consists of three parts: a training set, a development set and a test set. The datasets are made up of 6 different domains, namely Broadcast News (BN), Newswires (NW), Broadcast Conversations (BC), Telephone Conversations (TC), Web Blogs (WB), and Magazines (MZ). We closely follow the experimental settings as (Kong and Zhou, 2010; Chen and Ng, 2014, 2015, 2016), where we treat the training set for training and the development set for testing, because only the training and development set are annotated with ZPs. The statistics of training and testing data is shown in Table 1 and 2 respectively.

	Sentences #	Query #
General Train	18.47M	1.81M
Domain Train	122.8K	9.4K
Validation	11,191	2,667

Table 1: Statistics of training data, including pseudo training data and OntoNotes 5.0 training data.

3.2 Neural Network Setups

Training details of our neural network models are listed as follows.

[1]The news data is available at http://www.sogou.com/labs/dl/cs.html

[2]http://catalog.ldc.upenn.edu/LDC2013T19

	Docs	Sentences	Words	AZPs
Test	172	6,083	110K	1,713

Table 2: Statistics of test set (OntoNotes 5.0 development data).

- Embedding: We use randomly initialized embedding matrix with uniformed distribution in the interval [-0.1,0.1], and set units number as 256. No pre-trained word embeddings are used.
- Hidden Layer: We use GRU with 256 units, and initialize the internal matrix by random orthogonal matrices (Saxe et al., 2013). As GRU still suffers from the gradient exploding problem, we set gradient clipping threshold to 10.
- Vocabulary: As the whole vocabulary is very large (over 800K), we set a shortlist of 100K according to the word frequency and unknown words are mapped to 20 $\langle unkX \rangle$ using the proposed method.
- Optimization: We used ADAM update rule (Kingma and Ba, 2014) with an initial learning rate of 0.001, and used negative log-likelihood as the training objective. The batch size is set to 32.

All models are trained on Tesla K40 GPU. Our model is implemented with Theano (Theano Development Team, 2016) and Keras (Chollet, 2015).

3.3 Experimental results

Same to the previous researches that are related to zero pronoun resolution, we evaluate our system performance in terms of F-score (F). We focus on AZP resolution process, where we assume that gold AZPs and gold parse trees are given[3]. The same experimental setting is utilized in (Chen and Ng, 2014, 2015, 2016). The overall results are shown in Table 3, where the performances of each domain are listed in detail and overall performance is also shown in the last column.

- **Overall Performance**

We employ four Chinese ZP resolution baseline systems on OntoNotes 5.0 dataset. As we can

[3]All gold information are provided by the CoNLL-2012 shared task dataset

	NW (84)	MZ (162)	WB (284)	BN (390)	BC (510)	TC (283)	**Overall**
Kong and Zhou (2010)	34.5	32.7	45.4	51.0	43.5	48.4	44.9
Chen and Ng (2014)	38.1	31.0	50.4	45.9	53.8	**54.9**	48.7
Chen and Ng (2015)	46.4	39.0	51.8	53.8	49.4	52.7	50.2
Chen and Ng (2016)	48.8	41.5	56.3	**55.4**	50.8	53.1	52.2
Our Approach†	**59.2**	**51.3**	**60.5**	53.9	**55.5**	52.9	**55.3**

Table 3: Experimental result (F-score) on the OntoNotes 5.0 test data. The best results are marked with bold face. † indicates that our approach is statistical significant over the baselines (using t-test, with $p < 0.05$). The number in the brackets indicate the number of AZPs.

see that our model significantly outperforms the previous state-of-the-art system (Chen and Ng, 2016) by 3.1% in overall F-score, and substantially outperform the other systems by a large margin. When observing the performances of different domains, our approach also gives relatively consistent improvements among various domains, except for BN and TC with a slight drop. All these results approve that our proposed approach is effective and achieves significant improvements in AZP resolution.

In our quantitative analysis, we investigated the reasons of the declines in the BN and TC domain. A primary observation is that the word distributions in these domains are fairly different from others. The average document length of BN and TC are quite longer than other domains, which suggest that there is a bigger chance to have unknown words than other domains, and add difficulties to the model training. Also, we have found that in the BN and TC domains, the texts are often in oral form, which means that there are many irregular expressions in the context. Such expressions add noise to the model, and it is difficult for the model to extract useful information in these contexts. These phenomena indicate that further improvements can be obtained by filtering stop words in contexts, or increasing the size of task-specific data, while we leave this in the future work.

- **Effect of UNK processing**

As we have mentioned in the previous section, traditional unknown word replacing methods are vulnerable to the real word test. To alleviate this issue, we proposed the UNK processing mechanism to recover the UNK tokens to the real words. In Table 4, we compared the performance that with and without the proposed UNK processing, to show whether the proposed UNK processing method is effective. As we can see that, by applying our UNK processing mechanism, the model do learned the positional features in these low-frequency words, and brings over 3% improvements in F-score, which indicated the effectiveness of our UNK processing approach.

	F-score
Without UNK replacement	52.2
With UNK replacement	**55.3**

Table 4: Performance comparison on whether using the proposed unknown words processing.

- **Effect of Domain Adaptation**

We also tested out whether our domain adaptation method is effective. In this experiments, we used three different types of training data: only pseudo training data, only task-specific data, and our adaptation method, i.e. using pseudo training data in the pre-training step and task-specific data for domain adaptation step. The results are given in Table 5. As we can see that, using either pseudo training data or task-specific data alone can not bring inspiring result. By adopting our domain adaptation method, the model could give significant improvements over the other models, which demonstrate the effectiveness of our proposed two-step training approach. An intuition behind this phenomenon is that though pseudo training data is fairly big enough to train a reliable model parameters, there is still a gap to the real zero pronoun resolution tasks. On the contrary, though task-specific training data is exactly the same type as the real test, the quantity is not enough to train a reasonable model (such as word embedding). So it is better to make use of both to

take the full advantage.

However, as the original task-specific data is fairly small compared to pseudo training data, we also wondered if the large-scale pseudo training data is only providing rich word embedding information. So we use the large-scale pseudo training data for embedding training using GloVe toolkit (Pennington et al., 2014), and initialize the word embeddings in the "only task-specific data" system. From the result we can see that the pseudo training data provide more information than word embeddings, because though we used GloVe embeddings in "only task-specific data", it still can not outperform the system that uses domain adaptation which supports our claim.

	F-score
Only Pseudo Training Data	41.1
Only Task-Specific Data	44.2
Only Task-Specific Data + GloVe	50.9
Domain Adaptation	**55.3**

Table 5: Performance comparison of using different training data.

4 Error Analysis

To better evaluate our proposed approach, we performed a qualitative analysis of errors, where two major errors are revealed by our analysis, as discussed below.

4.1 Effect of Unknown Words

Our approach does not do well when there are lots of $\langle unk \rangle$s in the context of ZPs, especially when the $\langle unk \rangle$s appears near the ZP. An example is given below, where words with # are regarded as $\langle unk \rangle$s in our model.

> ϕ 登上# 太平山# 顶，将 香港岛# 和 维多利亚港# 的 美景 尽收眼底 。
> ϕ Successfully climbed# the peak of [Taiping Mountain]#, to have a panoramic view of the beauty of [Hong Kong Island]# and [Victoria Harbour]#.

In this case, the words "登上/climbed" and "太平山/Taiping Mountain" that appears immediately after the ZP "ϕ" are all regarded as $\langle unk \rangle$s in our model. As we model the sequence of words by RNN, the $\langle unk \rangle$s make the model more difficult to capture the semantic information of the sentence, which in turn influence the overall performance. Especially for the words that are near the ZP, which play important roles when modeling context information for the ZP. By looking at the word "顶/peak", it is hard to comprehend the context information, due to the several surrounding $\langle unk \rangle$s. Though our proposed unknown words processing method is effective in empirical evaluation, we think that more advanced method for unknown words processing would be of a great help in improving comprehension of the context.

4.2 Long Distance Antecedents

Also, our model makes incorrect decisions when the correct antecedents of ZPs are in long distance. As our model chooses answer from words in the context, if there are lots of words between the ZP and its antecedent, more noise information are introduced, and adds more difficulty in choosing the right answer. For example:

> 我 帮 不 了 那个 人 ... ... 那 天 结束 后 ϕ 回到 家中 。
> I can't help that guy ... ... After that day, ϕ return home.

In this case, the correct antecedent of ZP "ϕ" is the NP candidate "我/I". By seeing the contexts, we observe that there are over 30 words between the ZP and its antecedent. Although our model does not intend to fill the ZP gap only with the words near the ZP, as most of the antecedents appear just a few words before the ZPs, our model prefers the nearer words as correct antecedents. Hence, once there are lots of words between ZP and its nearest antecedent, our model can sometimes make wrong decisions. To correctly handle such cases, our model should learn how to filter the useless words and enhance the learning of long-term dependency.

5 Related Work

5.1 Zero pronoun resolution

For Chinese zero pronoun (ZP) resolution, early studies employed heuristic rules to Chinese ZP resolution. Converse (2006) proposes a rule-based method to resolve the zero pronouns, by utilizing Hobbs algorithm (Hobbs, 1978) in the CTB documents. Then, supervised approaches to this task have been vastly explored. Zhao and Ng (2007) first present a supervised machine learning approach to the identification and resolution of Chinese ZPs. Kong and Zhou (2010) develop a tree-kernel based approach for Chinese ZP resolution. More recently, unsupervised approaches

have been proposed. Chen and Ng (2014) develop an unsupervised language-independent approach, utilizing the integer linear programming to using ten overt pronouns. Chen and Ng (2015) propose an end-to-end unsupervised probabilistic model for Chinese ZP resolution, using a salience model to capture discourse information. Also, there have been many works on ZP resolution for other languages. These studies can be divided into rule-based and supervised machine learning approaches. Ferrández and Peral (2000) proposed a set of hand-crafted rules for Spanish ZP resolution. Recently, supervised approaches have been exploited for ZP resolution in Korean (Han, 2006) and Japanese (Isozaki and Hirao, 2003; Iida et al., 2006, 2007; Sasano and Kurohashi, 2011). Iida and Poesio (2011) developed a cross-lingual approach for Japanese and Italian ZPs where an ILP-based model was employed to zero anaphora detection and resolution.

In sum, most recent researches on ZP resolution are supervised approaches, which means that their performance highly relies on large-scale annotated data. Even for the unsupervised approach (Chen and Ng, 2014), they also utilize a supervised pronoun resolver to resolve ZPs. Therefore, the advantage of our proposed approach is obvious. We are able to generate large-scale pseudo training data for ZP resolution, and also we can benefit from the task-specific data for fine-tuning via the proposed two-step training approach.

5.2 Cloze-style Reading Comprehension

Our neural network model is mainly motivated by the recent researches on cloze-style reading comprehension tasks, which aims to predict one-word answer given the document and query. These models can be seen as a general model of mining the relations between the document and query, so it is promising to combine these models to the specific domain.

A representative work of cloze-style reading comprehension is done by Hermann et al. (2015). They proposed a methodology for obtaining large quantities of $\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$ triples. By using this method, a large number of training data can be obtained without much human intervention, and make it possible to train a reliable neural network. They used attention-based neural networks for this task. Evaluation on CNN/DailyMail datasets showed that their approach is much effective than traditional baseline systems.

While our work is similar to Hermann et al. (2015), there are several differences which can be illustrated as follows. Firstly, though we both utilize the large-scale corpus, they require that the document should accompany with a brief summary of it, while this is not always available in most of the document, and it may place an obstacle in generating limitless training data. In our work, we do not assume any prerequisite of the training data, and directly extract queries from the document, which makes it easy to generate large-scale training data. Secondly, their work mainly focuses on reading comprehension in the general domain. We are able to exploit large-scale training data for solving problems in the specific domain, and we proposed two-step training method which can be easily adapted to other domains as well.

6 Conclusion

In this study, we propose an effective way to generate and exploit large-scale pseudo training data for zero pronoun resolution task. The main idea behind our approach is to automatically generate large-scale pseudo training data and then utilize an attention-based neural network model to resolve zero pronouns. For training purpose, two-step training approach is employed, i.e. a **pre-training** and **adaptation** step, and this can be also easily applied to other tasks as well. The experimental results on OntoNotes 5.0 corpus are encouraging, showing that the proposed model and accompanying approaches significantly outperforms the state-of-the-art systems.

The future work will be carried out on two main aspects: First, as experimental results show that the unknown words processing is a critical part in comprehending context, we will explore more effective way to handle the UNK issue. Second, we will develop other neural network architecture to make it more appropriate for zero pronoun resolution task.

Acknowledgements

We would like to thank the anonymous reviewers for their thorough reviewing and proposing thoughtful comments to improve our paper. This work was supported by the National 863 Leading Technology Research Project via grant 2015AA015407, Key Projects of National Natural Science Foundation of China via grant 61632011,

and National Natural Science Youth Foundation of China via grant 61502120.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*. Association for Computational Linguistics, pages 13–16.

Chen Chen and Vincent Ng. 2013. Chinese zero pronoun resolution: Some recent advances. In *EMNLP*. pages 1360–1365.

Chen Chen and Vincent Ng. 2014. Chinese zero pronoun resolution: An unsupervised approach combining ranking and integer linear programming. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Chen Chen and Vincent Ng. 2015. Chinese zero pronoun resolution: A joint unsupervised discourse-aware model rivaling state-of-the-art resolvers. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. page 320.

Chen Chen and Vincent Ng. 2016. Chinese zero pronoun resolution with deep neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 778–788. http://aclweb.org/anthology/P16-1074.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Susan P Converse. 2006. Pronominal anaphora resolution in chinese .

Antonio Ferrández and Jesús Peral. 2000. A computational approach to zero-pronouns in spanish. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 166–172.

Na-Rae Han. 2006. *Korean zero pronouns: analysis and resolution*. Ph.D. thesis, Citeseer.

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

Jerry R Hobbs. 1978. Resolving pronoun references. *Lingua* 44(4):311–338.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 625–632.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing (TALIP)* 6(4):1.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ilp solution to zero anaphora resolution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 804–813.

Hideki Isozaki and Tsutomu Hirao. 2003. Japanese zero pronoun resolution based on ranking rules and machine learning. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*. Association for Computational Linguistics, pages 184–191.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Fang Kong and Guodong Zhou. 2010. A tree kernel-based unified framework for chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 882–891.

Lluis Marquez Emili Sapena M Antonia Marti Mariona Taule Veronique Hoste Massimo Poesio Yannick Versley Marta Recasens. 2010. Semeval-2010 task 1: Coreference resolution in multiple languages .

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1532–1543. http://aclweb.org/anthology/D14-1162.

Alessandro Moschitti Nianwen Xue Olga Uryupina Yuchen Zhang Sameer Pradhan. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes .

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to japanese zero anaphora resolution with large-scale lexicalized case frames. In *IJCNLP*. pages 758–766.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* .

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of chinese zero pronouns: A machine learning approach. In *EMNLP-CoNLL*. volume 2007, pages 541–550.

Discourse Mode Identification in Essays

Wei Song[†], **Dong Wang**[‡], **Ruiji Fu**[‡], **Lizhen Liu**[†], **Ting Liu**[§], **Guoping Hu**[‡]
[†]Information Engineering, Capital Normal University, Beijing, China
[‡]iFLYTEK Research, Beijing, China
[§]Harbin Institute of Technology, Harbin, China
{wsong, lzliu}@cnu.edu.cn, {dongwang4,rjfu, gphu}@iflytek.com, tliu@ir.hit.edu.cn

Abstract

Discourse modes play an important role in writing composition and evaluation. This paper presents a study on the manual and automatic identification of *narration*, *exposition*, *description*, *argument* and *emotion expressing* sentences in narrative essays. We annotate a corpus to study the characteristics of discourse modes and describe a neural sequence labeling model for identification. Evaluation results show that discourse modes can be identified automatically with an average F1-score of 0.7. We further demonstrate that discourse modes can be used as features that improve automatic essay scoring (AES). The impacts of discourse modes for AES are also discussed.

1 Introduction

Discourse modes, also known as rhetorical modes, describe the purpose and conventions of the main kinds of language based communication.Most common discourse modes include narration, description, exposition and argument. A typical text would make use of all the modes, although in a given one there will often be a main mode. Despite their importance in writing composition and assessment (Braddock et al., 1963), there is relatively little work on analyzing discourse modes based on computational models. We aim to contribute for automatic discourse mode identification and its application on writing assessment.

The use of discourse modes is important in writing composition, because they relate to several aspects that would influence the quality of a text.

First, discourse modes reflect the organization of a text. Natural language texts consist of sentences which form a unified whole and make up the discourse (Clark et al., 2013). Recognizing the structure of text organization is a key part for discourse analysis. Meurer (2002) points that discourse modes stand for unity as they constitute general patterns of language organization strategically used by the writer. Smith (2003) also proposes to study discourse passages from a linguistic view of point through discourse modes. The organization of a text can be realized by segmenting text into passages according to the set of discourse modes that are used to indicate the functional relationship between the several parts of the text. For example, the writer can present major events through narration, provide details with description and establish ideas with argument. The combination and interaction of various discourse modes make an organized unified text.

Second, discourse modes have rhetorical significance. Discourse modes are closely related to rhetoric (Connors, 1981; Brooks and Warren, 1958), which offers a principle for learning how to express material in the best way. Discourse modes have different preferences on expressive styles. Narration mainly controls story progression by introducing and connecting events; exposition is to instruct or explain so that the language should be precise and informative; argument is used to convince or persuade through logical and inspiring statements; description attempts to bring detailed observations of people and scenery, which is related to the writing of figurative language; the way to express emotions may relate to the use of rhetorical devices and poetic language. Discourse modes reflect the variety of expressive styles. The flexible use of various discourse modes should be important evidence of language proficiency.

According to the above thought, we propose the discourse mode identification task. In particular, we make the following contributions:

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 112–122
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1011

- We build a corpus of narrative essays written by Chinese students in native language. Sentence level discourse modes are annotated with acceptable inter-annotator agreement. Corpus analysis reveals the characteristics of discourse modes in several aspects, including discourse mode distribution, co-occurrence and transition patterns.

- We describe a multi-label neural sequence labeling approach for discourse mode identification so that the co-occurrence and transition preferences can be captured. Experimental results show that discourse modes can be identified with an average F1-score of 0.7, indicating that automatic discourse mode identification is feasible.

- We demonstrate the effectiveness of taking discourse modes into account for automatic essay scoring. A higher ratio of description and emotion expressing can indicate essay quality to a certain extent. Discourse modes can be potentially used as features for other NLP applications.

2 Related Work

2.1 Discourse Analysis

Discourse analysis is an important subfield of natural language processing (Webber et al., 2011). Discourse is expected to be both cohesive and coherent. Many principles are proposed for discourse analysis, such as coherence relations (Hobbs, 1979; Mann and Thompson, 1988), the centering theory for local coherence (Grosz et al., 1995) and topic-based text segmentation (Hearst, 1997). In some domains, discourse can be segmented according to specific discourse elements (Hutchins, 1977; Teufel and Moens, 2002; Burstein et al., 2003; Clerehan and Buchbinder, 2006; Song et al., 2015).

This paper focuses on discourse modes influenced by Smith (2003). From the linguistic view of point, discourse modes are supposed to have different distributions of situation entity types such as event, state and generic (Smith, 2003; Mavridou et al., 2015). Therefore, there is work on automatically labeling clause level situation entity types (Palmer et al., 2007; Friedrich et al., 2016). Actually, situation entity type identification is also a challenging problem. It is even harder for processing Chinese language, since Chinese doesn't have grammatical tense (Xue and Zhang, 2014) and sentence components are often omitted. This increases the difficulties for situation entity type based discourse mode identification. In this paper, we investigate an end-to-end approach to directly model discourse modes without the necessity of identifying situation entity types first.

2.2 Automatic Writing Assessment

Automatic writing assessment is an important application of natural language processing. The task aims to let computers have the ability to appreciate and criticize writing. It would be hugely beneficial for applications like automatic essay scoring (AES) and content recommendation.

AES is the task of building a computer-aided scoring system, in order to reduce the involvement of human raters. Traditional approaches are based on supervised learning with designed feature templates (Larkey, 1998; Burstein, 2003; Attali and Burstein, 2006; Chen and He, 2013; Phandi et al., 2015; Cummins et al., 2016). Recently, automatic feature learning based on neural networks starts to draw attentions (Alikaniotis et al., 2016; Dong and Zhang, 2016; Taghipour and Ng, 2016).

Writing assessment involves highly technical aspects of language and discourse. In addition to give a score, it would be better to provide explainable feedbacks to learners at the same time. Some work has studied several aspects such as spelling errors (Brill and Moore, 2000), grammar errors (Rozovskaya and Roth, 2010), coherence (Barzilay and Lapata, 2008), organization of argumentative essays (Persing et al., 2010) and the use of figurative language (Louis and Nenkova, 2013). This paper extends this line of work by taking discourse modes into account.

2.3 Neural Sequence Modeling

A main challenge of discourse analysis is hard to collect large scale data due to its complexity, which may lead to data sparseness problem. Recently, neural networks become popular for natural language processing (Bengio et al., 2003; Collobert et al., 2011). One of the advantages is the ability of automatic representation learning. Representing words or relations with continuous vectors (Mikolov et al., 2013; Ji and Eisenstein, 2014) embeds semantics in the same space, which benefits alleviating the data sparseness problem

and enables end-to-end and multi-task learning. Recurrent neural networks (RNNs) (Graves, 2012) and the variants like Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent (GRU) (Cho et al., 2014) neural networks show good performance for capturing long distance dependencies on tasks like Named Entity Recognition (NER) (Chiu and Nichols, 2016; Ma and Hovy, 2016), dependency parsing (Dyer et al., 2015) and semantic composition of documents (Tang et al., 2015). This work describes a hierarchical neural architecture with multiple label outputs for modeling the discourse mode sequence of sentences.

3 Discourse Mode Annotation

We are interested in the use of discourse modes in writing composition. This section describes the discourse modes we are going to study, an annotated corpus of student essays and what we learn from corpus analysis.

3.1 Discourse Modes

Discourse modes have several taxonomies in the literature. Four basic discourse modes are *narration*, *description*, *exposition* and *argument* in English composition and rhetoric (Bain, 1890). Smith (2003) proposes five modes for studying discourse passages: *narrative*, *description*, *report*, *information* and *argument*. In Chinese composition, discourse modes are categorized into *narration*, *description*, *exposition*, *argument* and *emotion expressing* (Zhu, 1983).

These taxonomies are similar. Their elements can mostly find corresponding ones in other taxonomies literally or conceptually, e.g., exposition mode has similar functions to information mode. Emotion expressing that is to express the writer's emotions is relatively special. It can be realized by expressing directly or through lyrical writing with beautiful and poetic language. It is also related to appeal to emotion, which is a method for argumentation by the manipulation of the recipient's emotions in classical rhetoric (Aristotle and Kennedy, 2006). Proper emotion expressing can touch the hearts of the readers and improve the expressiveness of writing. Therefore, considering it as an independent mode is also reasonable.

We cope with essays written in Chinese in this work so that we follow the Chinese convention with five discourse modes. Emotion expressing is added on the basis of four recognized discourse modes and Smith's report mode is viewed as a subtype of description mode: *dialogue description*.

In summary, we study the following discourse modes:

- **Narration** introduces an event or series of events into the universe of discourse. The events are temporally related according to narrative time.
  E.g., *Last year, we drove to San Francisco along the State Route 1 (SR 1).*

- **Exposition** has a function to explain or instruct. It provides background information in narrative context. The information presented should be general and (expected to be) well accepted truth.
  E.g., *SR 1 is a major north-south state highway that runs along most of the Pacific coastline of the U.S.*

- **Description** re-creates, invents, or vividly show what things are like according to the five senses so that the reader can picture that which is being described.
  E.g., *Along SR 1 are stunning rugged coastline, coastal forests and cliffs, beautiful little towns and some of the West coast's most amazing nature.*

- **Argument** makes a point of view and proves its validity towards a topic in order to convince or persuade the reader.
  E.g., *Point Arena Lighthouse is a must see along SR 1, in my opinion.*

- **Emotion expressing**[1] presents the writer's emotions, usually in a subjective, personal and lyrical way, to involve the reader to experience the same situations and to be touched.
  E.g., *I really love the ocean, the coastline and all the amazing scenery along the route. When could I come back again?*

The distinction between discourse modes is expected to be clarified conceptually by considering their different communication purposes. However, there would still be specific ambiguous and vague cases. We will describe the data annotation and corpus analysis in the following parts.

[1] In some cases, we use emotion for short.

	INITIAL			FINAL		
	P	R	F	P	R	F
Nar	0.90	0.88	0.89	0.96	0.84	0.90
Exp	0.79	0.73	0.76	0.89	0.76	0.81
Des	0.84	0.74	0.79	0.87	0.65	0.74
Emo	0.75	0.68	0.71	0.79	0.73	0.76
Arg	0.35	0.28	0.31	0.76	0.61	0.68
Avg.	0.73	0.66	0.69	0.87	0.71	0.78
κ		0.55			0.72	

Table 1: Inter-annotator agreement between two annotators on the dominant discourse mode. Initial: The result of the first round annotation; Final: The result of the final annotation; κ: Agreement measured with Cohen's Kappa.

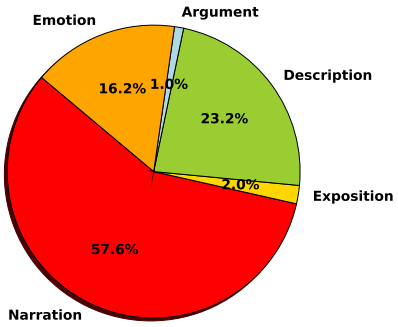


Figure 1: The distribution of dominant modes.

3.2 Data Annotation

Discourse modes are almost never found in a pure form but are embedded one within another to help the writer achieve the purpose, but the emphasis varies in different types of writing. We focus on narrative essays. A good narrative composition must properly manipulate multiple discourse modes to make it vivid and impressive.

The corpus has 415 narrative essays written by high school students in their native Chinese language.The average number of sentences is 32 and the average length is 670 words.

We invited two high school teachers to annotate discourse modes at sentence level, expecting their background help for annotation. A detail manual was discussed before annotation.

We notice that discourse modes can mix in the same sentence. Therefore, the annotation standard allows that one sentence can have multiple modes. But we require that every sentence should have a *dominant mode*. The annotators should try to think in the writer's perspective and guess the writer's main purpose of writing the sentence in order to decide the dominant mode.

Among the discourse modes, description can be applied in various situations. We focus on the following description types: portrait, appearance, action, dialogue, psychological, environment and detail description. If a sentence has any type of description, it would be assigned a description label.

3.3 Corpus Analysis

We conducted corpus analysis on the annotated data to gain observations on several aspects.

Inter-Annotator Agreement: 50 essays were independently annotated by two annotators. We evaluate the inter-annotator agreement on the dominant mode. The two annotators' annotations are used as the golden answer and prediction respectively. We compute the precision, recall and F1-score for each discourse mode separately to measure the inter-annotator agreement. Precision and recall are symmetric for the two annotators.

The result of the first round annotation is shown in the INITIAL columns of Table 1. The agreement on argument mode is low, while the agreement on other modes is acceptable. The average F1-score is 0.69. The Cohen's Kappa (Cohen et al., 1960) is 0.55 over all judgements on the dominant mode.

The main disagreement on argument lies in the confusion with emotion expressing. Consider the following sentence:

> *Father's love is the fire that lights the lamp of hope.*

One annotator thought that it is expressed in an emotional and lyrical way so that the discourse mode should be emotion expressing. The other one thought that it (implicitly) gives a point and should be an argument. Many disagreements happened in cases like this.

Based on the observations of the first round annotation, we discussed and updated the manual and let the annotators rechecked their annotations. The final result is shown in the FINAL columns of Table 1. The agreement on description decreases. Annotators seem to be more conservative on labeling description as the dominant mode. The overall average F1-score increases to 0.78 and the Cohen's Kappa is 0.72. This indicates that humans can reach an acceptable agreement on the dominant discourse mode of sentences after training.

Discourse mode distribution: After the training phase, the annotators labeled the whole corpus. Figure 1 shows the distribution of dominant

Mode	Nar	Exp	Des	Emo	Arg
Nar	5285	11	2552	65	2
Exp	-	148	11	1	1
Des	-	-	2538	105	8
Emo	-	-	-	1947	63
Arg	-	-	-	-	318

Table 2: Co-occurrence of discourse modes in the same sentences. The numbers in diagonal indicate the number of sentences with a single mode.

from \ to	Nar	Exp	Des	Emo	Arg
Nar	72%	-	17%	7%	1%
Exp	59%	8%	8%	16%	6%
Des	42%	-	53%	3%	-
Emo	25%	2%	4%	66%	1%
Arg	27%	-	4%	12%	54%
Begin with	50%	3%	6%	32%	7%
End with	12%	1%	2%	76%	6%

Table 3: Transition between discourse modes of consecutive sentences and the distribution of discourse modes that essays begin with and end with.

discourse modes. The distribution is imbalanced. Narration, description and emotion expressing are the main discourse modes in narrative essays, while exposition and argument are rare.

Co-occurrence: Statistics show that 78% of sentences have only one discourse mode, and 19% have two discourse modes, and 3% have more than two discourse modes.

Table 2 shows the co-occurrence of discourse modes. The numbers that are in the diagonal represent the distribution of discourse modes of sentences with only one mode. The numbers that are not in the diagonal indicate the co-occurrence of modes in the same sentences. We can see that description tends to co-occur with narration and emotion expressing. Description can provide states that happen together with events and emotion-evoking scenes are often described to elicit a strong emotional response, for example:

> *The bright moon hanging on the distant sky reminds me of my hometown miles away.*

Emotion expressing and argument also co-occur in some cases. It is reasonable, since a successful emotional appeal can enhance the effectiveness of an argument.

Generally, these observations are consistent with intuition. Properly combining multiple modes could produce impressive sentences.

Transition: Table 3 shows the transition matrix between the dominant modes of consecutive sentences within the same paragraphs. All modes tend to transit to themselves except exposition, which is rare and usually brief. This means that discourse modes of adjacent sentences have high correlation. We also see that narration and emotion are more often at the beginning and the end of essays. The above observations indicate that discourse modes have local preferred patterns.

To summarize, the implications of corpus analysis include: (1) Manual identification of discourse modes is feasible with an acceptable inter-annotator agreement; (2) The distribution of discourse modes in narrative essays is imbalanced; (3) About 22% sentences have multiple discourse modes; (4) Discourse modes have local transition patterns that consecutive discourse modes have high correlation.

4 Discourse Mode Identification based on Neural Sequence Labeling

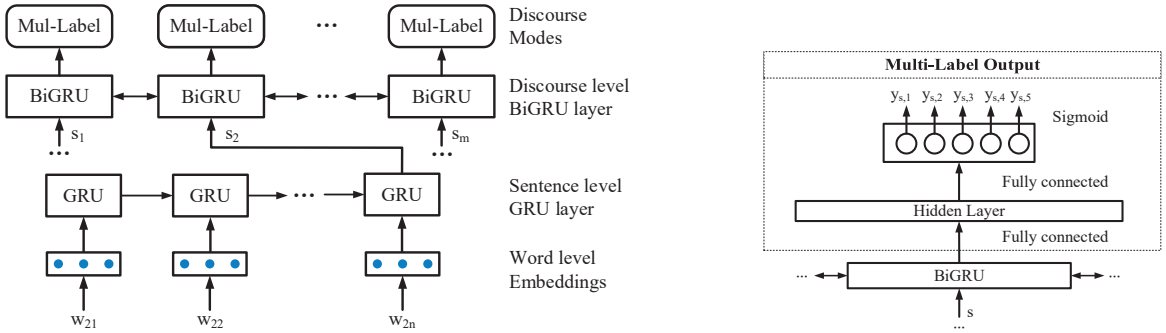
This section describes the proposed method for discourse mode identification. According to the corpus analysis, sentences often have multiple discourse modes and prefer local transition patterns. Therefore, we view this task as a multi-label sequence labeling problem.

4.1 Model

We propose a hierarchical neural sequence labeling model to capture multiple level information. Figure 2(a) shows the basic architecture. We introduce it from the bottom up.

Word level embedding layer: We transform words into continuous vectors, word embeddings. Vector representation of words is useful for capturing semantic relatedness. This should be effective in our case, since large amount of training data is not available. It is unrealistic to learn the embedding parameters on limited data so that we just look up embeddings of words from a pre-trained word embedding table. The pre-trained word embeddings were learned with the Word2Vec toolkit (Mikolov et al., 2013) on a domain corpus which consists of about 490,000 student essays. The embeddings are kept unchanged during learning and prediction.

Sentence level GRU layer: Each sentence is a sequence of words. We feed the word embeddings into a forward recurrent neural networks. Here,



(a) The basic hierarchical architecture.

(b) The detail of the Mul-Label layer

Figure 2: The multi-label neural sequence labeling model for discourse mode identification.

we use the GRU (Cho et al., 2014) as the recurrent unit. The GRU is to make each recurrent unit to adaptively capture dependencies of different time scales. The output of the last time-step is used as the representation of a sentence.

Discourse level bidirectional-GRU layer: An essay consists of a sequence of sentences. Accessing information of past and future sentences provides more contextual information for current prediction. Therefore, we use a bidirectional RNN to connect sentences. We use the GRU as the recurrent unit, which is also shown effective on semantic composition of documents for sentiment classification (Tang et al., 2015). The BiGRU represents the concatenation of the hidden states of the forward GRU and the backward GRU units.

Multi-Label layer: Since one sentence can have more than one discourse mode, our model allows multiple label outputs. Figure 2(b) details the Mul-Label layer in Figure 2(a). The representation of each sentence after the bidirectional-GRU layer is first fully connected to a hidden layer. The hidden layer output is then fully connected to a five-way output layer, corresponding to five discourse modes. The sigmoid activation function is applied to each way to get the probability that whether corresponding discourse mode should be assigned to the sentence.

In the training phase, the probability of any labeled discourse modes is set to 1 and the others are set to 0. In the prediction phase, if the predicted probability of a discourse mode is larger than 0.5, the discourse mode would be assigned.

4.1.1 Considering Paragraph Boundaries

Different from NER that processes a single sentence each time, our task processes sequences of sentences in discourse, which are usually grouped by paragraphs to split the whole discourse into several relatively independent segments. Sentences from different paragraphs should have less effect to each other, even though they are adjacent.

To capture paragraph boundary information, we insert an empty sentence at the end of every paragraph to indicate a paragraph boundary. The empty sentence is represented by a zero vector and its outputs are set to zeros as well. We expect this modification can better capture position related information.

4.2 Implementation Details

We implement the model using the Keras library.[2] The models are trained with the binary cross-entropy objective. The optimizer is Adam (Kingma and Ba, 2014). The word embedding dimension is 50. The dimension of the hidden layer in Mul-Label layer is 100. The length of sentences is fixed as 40. All other parameters are set by default parameter values. We adopt early stopping strategy (Caruana et al., 2000) to decide when the training process stops.

4.3 Evaluation

4.3.1 Data

We use 100 essays as the test data. The remaining ones are used as the training data. 10% of the shuffled training data is used for validation.

4.3.2 Comparisons

We compare the following systems:

- SVM: We use bag of ngram (unigram and bigram) features to train a support vector classifier for sentence classification.

[2] https://github.com/fchollet/keras/

- CNN: We implement a convolutional neural network (CNN) based method (Kim, 2014), as it is the state-of-the-art for sentence classification.

- GRU: We use the sentence level representation in Figure 2(a) for sentence classification.

- GRU-GRU(GG): This method is introduced in this paper in §4.1, but it doesn't consider paragraph information.

- GRU-GRU-SEG (GG-SEG): The model considers paragraph information on the top of GG as introduced in §4.1.1.

The first three classification based methods classify sentences independently. To deal with multiple labels, the classifiers are trained for each discourse mode separately. At prediction time, if the classifier for any discourse mode predicts a sentence as positive, the corresponding discourse mode would be assigned.

4.3.3 Evaluation Results

Table 4 shows the experimental results. We evaluate the systems for each discourse mode with F1-score, which is the harmonic mean of precision and recall. The best performance is in bold.

The SVM performs worst among all systems. The reason is due to the data sparseness and term-mismatch problem, since the size of the annotated dataset is not big enough. In contrast, systems based on neural networks with pre-trained word embeddings achieve much better performance.

The CNN and GRU have comparable performance. The GRU is slightly better. The two methods don't consider the semantic representations of adjacent sentences.

The GG and GG-SEG explore the semantic information of sentences in a sequence by the bidirectional GRU layer. The results demonstrate that considering such information improve the performance on all discourse modes. This proves the advantage of sequential identification compared with isolated sentence classification.

We can see that the GG-SEG further improves the performance on three minority discourse modes compared with GG. This means that the minority modes may have stronger preference to special locations. Exposition benefits most, since many exposition sentences in our dataset are isolated.

Model \ Mode	Nar	Des	Emo	Arg	Exp
SVM	0.672	0.588	0.407	0.152	0.095
CNN	0.793	0.764	0.594	0.333	0.293
GRU	0.800	0.784	0.615	0.402	0.364
GG	**0.822**	**0.797**	0.680	0.423	0.481
GG-SEG	0.815	0.791	**0.717**	**0.483**	**0.667**

Table 4: The F1-scores of systems on each discourse mode.

The performance on argument is not so good. As we discussed in corpus analysis, argument and emotion expressing mode interact frequently. Because the amount of emotion expressing sentences is much more, distinguishing argument from them is hard. Actually, their functions in narrative essays seem to be similar that both are to deepen the author's response or evoke the reader's response to the story.

The overall average F1-score can reach to 0.7 and the performance on identifying three most common discourse modes are consistent, with an average F1-score above 0.76 using the proposed neural sequence labeling models. Automatic discourse mode identification should be feasible.

5 Essay Scoring with Discourse Modes

Discourse mode identification can potentially provide features for downstream NLP applications. This section describes our attempt to explore discourse modes for automatic essay scoring (AES).

5.1 Essay Scoring Framework

We adopt the standard regression framework for essay scoring. We use support vector regression (SVR) and Bayesian linear ridge regression (BLRR), which are used in recent work (Phandi et al., 2015). The key is to design effective features.

5.2 Features

The basic feature sets are based on (Phandi et al., 2015).The original feature sets include:

- Length features
- Part-Of-Speech (POS) features
- Prompt features
- Bag of words features

We re-implement the feature extractors exactly according to the description in (Phandi et al., 2015) except for the POS features, since we don't

Prompt	#Essays	Avg. len	Score	
			Range	Median
1	4000	628	0-60	46
2	4000	660	0-50	41
3	3300	642	0-50	41

Table 5: Details of the three datasets for AES.

have correct POS ngrams for Chinese. We complement two additional features: (1) The number of words occur in Chinese Proficiency Test 6 vocabulary; (2) The number of Chinese idioms used.

We further design discourse mode related features for each essay:

- Mode ratio: For each discourse mode, we compute its mode ratio according to $ratio = \frac{\#\text{sentences with the discourse mode}}{\#\text{sentences in the essay}}$. Such features indicate the distribution of discourse modes.

- Bag of ngrams of discourse modes: We use the number of unigrams and bigrams of the dominant discourse modes of the sequence of sentences in the essay as features.

5.3 Experimental Settings

The experiments were conducted on narrative essays written by Chinese middle school students in native language during regional tests. There are three prompts and students are required to write an essay related to the given prompt with no less than 600 Chinese characters. All these essays were evaluated by professional teachers.

We randomly sampled essays from each prompt for experiments. Table 5 shows the details of the datasets. We ran experiments on each prompt dataset respectively by 5-fold cross-validation.

The GG-SEG model was used to identify discourse modes of sentences. Notice that a sentence can have multiple discourse modes. The mode ratio features are computed for each mode separately. When extracting the bag of ngrams of discourse modes features, the discourse mode with highest prediction probability was chosen as the dominant discourse mode.

We use the Quadratic Weighted Kappa (QWK) as the evaluation metric.

5.4 Evaluation Results

Table 6 shows the evaluation results of AES on three datasets. We can see that the BLRR algorithm performs better than the SVR algorithm. No matter which algorithm is adopted, adding discourse mode features make positive contributions for AES compared with using basic feature sets. The trends are consistent over all three datasets.

Prompt	QWK Score		
	1	2	3
SVR-Basic	0.554	0.468	0.457
+ mode	0.6	0.501	0.481
BLRR-Basic	0.683	0.557	0.513
+ mode	**0.696**	**0.565**	**0.527**

Table 6: Evaluation results of AES on three datasets. Basic: the basic feature sets; mode: discourse mode features.

Prompt	1	2	3	Avg
LEN	0.59	0.52	0.45	0.52
Des	0.23	0.24	0.24	0.24
Emo	0.09	0.15	0.12	0.12
Exp	-0.07	0.01	0.01	-0.03
Arg	-0.08	-0.06	-0.1	-0.08
Nar	-0.11	-0.15	-0.12	-0.13

Table 7: Pearson correlation coefficients of mode ratio to essay score. LEN represents essay length.

Impact of discourse mode ratio on scores: We are interested in which discourse mode correlates to essay scores best. Table 7 shows the Pearson correlation coefficient between the mode ratio and essay score. LEN represents the correlation of essay length and is listed as a reference. We can see that the ratio of narration has a negative correlation, which means just narrating stories without auxiliary discourse modes would lead to poor scores. The description mode ratio has the strongest positive correlation to essay scores. This may indicate that using vivid language to provide detail information is essential in writing narrative essays. Emotion expressing also has a positive correlation. It is reasonable since emotional writing can involve readers into the stories. The ratio of argument shows a negative correlation. The reason may be that: first, the identification of argument is not good enough; second, the existence of an argument doesn't mean the quality of argumentation is good. Exposition has little effect on essay scores.

Generally, the distribution of discourse modes shows correlations to the quality of essays. This may relate to the difficulties of manipulating different discourse modes. It is easy for students to use narration, but it is more difficult to manipulate description and emotion expressing well. As a result, the ability of descriptive and emotional writ-

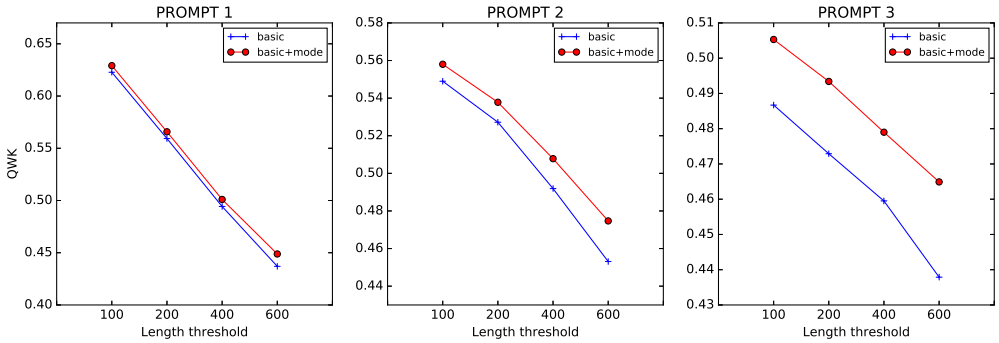


Figure 3: QWK scores on essays satisfying different length thresholds on three prompts. Basic: the basic feature sets; mode: discourse mode features.

ing should be an indicator of language proficiency and can better distinguish the quality of writing.

Impact on scoring essays with various length: It is easy to understand that length is a strong indicator for essay scoring. It is interesting to study that when the effect of length becomes weaker, e.g., the lengths of essays are close, how does the performance of the AES system change?

We conducted experiments on essays with various lengths. Only essays that the length is no less than a given threshold are selected for evaluation. The threshold is set to 100, 200, 400 and 600 Chinese characters respectively. We ran 5-fold cross-validation with BLRR on the datasets after essay selection.

Figure 3 shows the results on three datasets. We can see the following trends: (1) The QWK scores decrease along with shorter essays are removed gradually; (2) Adding discourse mode features always improves the performance; (3) As the threshold becomes larger, the improvements by adding discourse mode features become larger.

The results indicate that the current AES system can achieve a high correlation score when the lengths of essays differ obviously. Even the simple features like length can judge that short essays tend to have low scores. However, when the lengths of essays are close, AES would face greater challenges, because it is required to deeper understand the properties of well written essays. In such situations, features that can model more advanced aspects of writing, such as discourse modes, should play a more important role. It should be also essential for evaluating essays written in the native language of the writer, when spelling and grammar are not big issues any more.

6 Conclusion

This paper has introduced a fundamental but less studied task in NLP—discourse mode identification, which is designed in this work to automatically identify five discourse modes in essays.

A corpus of narrative student essays was manually annotated with discourse modes at sentence level, with acceptable inter-annotator agreement. The corpus analysis revealed several aspects of characteristics of discourse modes including the distribution, co-occurrence and transition patterns.

Considering these characteristics, we proposed a neural sequence labeling approach for identifying discourse modes. The experimental results demonstrate that automatic discourse mode identification is feasible.

We evaluated discourse mode features for automatic essay scoring and draw preliminary observations. Discourse mode features can make positive contributions, especially in challenging situations when simple surface features don't work well. The ratio of description and emotion expressing is shown to be positively correlated to essay scores.

In future, we plan to exploit discourse mode identification for providing novel features for more downstream NLP applications.

Acknowledgements

The research work is partially funded by the National High Technology Research and Development Program (863 Program) of China (No.2015AA015409), National Natural Science Foundation of China (No.61402304), Ministry of Education (No.14YJAZH046), Beijing Municipal Education Commission (KM201610028015, Connotation Development) and Beijing Advanced Innovation Center for Imaging Technology.

References

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. In *Proceedings of ACL 2016*. pages 715–725.

Omer Aristotle and George A Kennedy. 2006. *On rhetoric: A theory of civic discourse*. Oxford University Press.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment* 4(3).

Alexander Bain. 1890. *English composition and rhetoric*. Longmans, Green & Company.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1):1–34.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Richard Reed Braddock, Richard Lloyd-Jones, and Lowell Schoer. 1963. *Research in written composition*. JSTOR.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of ACL 2000*. pages 286–293.

Cleanth Brooks and Robert Penn Warren. 1958. *Modern rhetoric*. Harcourt, Brace.

Jill Burstein. 2003. The e-rater® scoring engine: Automated essay scoring with natural language processing. .

Jill Burstein, Daniel Marcu, and Kevin Knight. 2003. Finding the write stuff: Automatic identification of discourse structure in student essays. *Intelligent Systems, IEEE* 18(1):32–39.

Rich Caruana, Steve Lawrence, and Lee Giles. 2000. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Proceedings of NIPS 2000*. pages 402–408.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of EMNLP 2013*. pages 1741–1752.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics* 4:357–370.

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

Alexander Clark, Chris Fox, and Shalom Lappin. 2013. *The handbook of computational linguistics and natural language processing*. John Wiley & Sons.

Rosemary Clerehan and Rachelle Buchbinder. 2006. Toward a more valid account of functional text quality: The case of the patient information leaflet. *Text & Talk-An Interdisciplinary Journal of Language, Discourse Communication Studies* 26(1):39–68.

Jacob Cohen et al. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20(1):37–46.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Robert J Connors. 1981. The rise and fall of the modes of discourse. *College Composition and Communication* 32(4):444–455.

Ronan Cummins, Meng Zhang, and Ted Briscoe. 2016. Constrained multi-task learning for automated essay scoring. In *Proceedings of ACL 2016*. pages 789–799.

Fei Dong and Yue Zhang. 2016. Automatic features for essay scoring – an empirical study. In *Proceedings of EMNLP 2016*. pages 1072–1077.

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

Annemarie Friedrich, Alexis Palmer, and Manfred Pinkal. 2016. Situation entity types: automatic classification of clause-level aspect. In *Proceedings of ACL 2016*. pages 1757–1768.

Alex Graves. 2012. Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer Berlin Heidelberg, pages 5–13.

Barbara J Grosz, Scott Weinstein, and Aravind K Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics* 21(2):203–225.

Marti A Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics* 23(1):33–64.

Jerry R Hobbs. 1979. Coherence and coreference. *Cognitive science* 3(1):67–90.

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

John Hutchins. 1977. On the structure of scientific texts. *UEA Papers in Linguistics* 5(3):18–39.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of ACL 2014*. pages 13–24.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP 2014*. pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Leah S Larkey. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of SIGIR 1998*. pages 90–95.

Annie Louis and Ani Nenkova. 2013. What makes writing great? first experiments on article quality prediction in the science journalism domain. *Transactions of the Association for Computational Linguistics* 1:341–352.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of ACL 2016*. pages 1064–1074.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse* 8(3):243–281.

Kleio-Isidora Mavridou, Annemarie Friedrich, Melissa Peate Sørensen, Alexis Palmer, and Manfred Pinkal. 2015. Linking discourse modes and situation entity types in a cross-linguistic corpus study. In *Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem)*. page 12.

José Luiz Meurer. 2002. Genre as diversity, and rhetorical mode as unity in language use. *Ilha do Desterro A Journal of English Language, Literatures in English and Cultural Studies* (43):061–082.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS 2013*. pages 3111–3119.

Alexis Palmer, Elias Ponvert, Jason Baldridge, and Carlota Smith. 2007. A sequencing model for situation entity classification. In *Proceedings of ACL 2007*. pages 896–903.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling organization in student essays. In *Proceedings of EMNLP 2010*. pages 229–239.

Peter Phandi, Kian Ming A. Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of EMNLP 2015*. pages 431–439.

Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of EMNLP 2010*. pages 961–970.

Carlota S Smith. 2003. *Modes of discourse: The local structure of texts*, volume 103. Cambridge University Press.

Wei Song, Ruiji Fu, Lizhen Liu, and Ting Liu. 2015. Discourse element identification in student essays based on global and local cohesion. In *Proceedings of EMNLP 2015*. pages 2255–2261.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of EMNLP 2016*. pages 1882–1891.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP 2015*. pages 1422–1432.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Computational linguistics* 28(4):409–445.

Bonnie Webber, Markus Egg, and Valia Kordoni. 2011. Discourse structure and language technology. *Natural Language Engineering* 18(4):437–490.

Nianwen Xue and Yuchen Zhang. 2014. Buy one get one free: Distant annotation of chinese tense, event type and modality. In *Proceedings of LREC 2014*. pages 1412–1416.

Boshi Zhu. 1983. *写作概论(An Introduction to Writing)*. Wuhan, Hubei Educational Press.

A Convolutional Encoder Model for Neural Machine Translation

Jonas Gehring, Michael Auli, David Grangier, Yann N. Dauphin
Facebook AI Research

Abstract

The prevalent approach to neural machine translation relies on bi-directional LSTMs to encode the source sentence. We present a faster and simpler architecture based on a succession of convolutional layers. This allows to encode the source sentence simultaneously compared to recurrent networks for which computation is constrained by temporal dependencies. On WMT'16 English-Romanian translation we achieve competitive accuracy to the state-of-the-art and on WMT'15 English-German we outperform several recently published results. Our models obtain almost the same accuracy as a very deep LSTM setup on WMT'14 English-French translation. We speed up CPU decoding by more than two times at the same or higher accuracy as a strong bi-directional LSTM.[1]

1 Introduction

Neural machine translation (NMT) is an end-to-end approach to machine translation (Sutskever et al., 2014). The most successful approach to date encodes the source sentence with a bi-directional recurrent neural network (RNN) into a variable length representation and then generates the translation left-to-right with another RNN where both components interface via a soft-attention mechanism (Bahdanau et al., 2015; Luong et al., 2015a; Bradbury and Socher, 2016; Sennrich et al., 2016a). Recurrent networks are typically parameterized as long short term memory networks (LSTM; Hochreiter et al. 1997) or gated recurrent units (GRU; Cho et al. 2014), often with residual or skip connections (Wu et al., 2016; Zhou et al., 2016) to enable stacking of several layers (§2).

There have been several attempts to use convolutional encoder models for neural machine translation in the past but they were either only applied to rescoring n-best lists of classical systems (Kalchbrenner and Blunsom, 2013) or were not competitive to recurrent alternatives (Cho et al., 2014a). This is despite several attractive properties of convolutional networks. For example, convolutional networks operate over a fixed-size window of the input sequence which enables the simultaneous computation of all features for a source sentence. This contrasts to RNNs which maintain a hidden state of the entire past that prevents parallel computation within a sequence.

A succession of convolutional layers provides a shorter path to capture relationships between elements of a sequence compared to RNNs.[2] This also eases learning because the resulting tree-structure applies a fixed number of non-linearities compared to a recurrent neural network for which the number of non-linearities vary depending on the time-step. Because processing is bottom-up, all words undergo the same number of transformations, whereas for RNNs the first word is over-processed and the last word is transformed only once.

In this paper we show that an architecture based on convolutional layers is very competitive to recurrent encoders. We investigate simple average pooling as well as parameterized convolutions as an alternative to recurrent encoders and enable very deep convolutional encoders by using residual connections (He et al., 2015; §3).

We experiment on several standard datasets and compare our approach to variants of recurrent encoders such as uni-directional and bi-directional LSTMs. On WMT'16 English-Romanian translation we achieve accuracy that is very competitive to the current state-of-the-art result. We perform competitively on WMT'15 English-German, and nearly match the performance of the best WMT'14 English-French system based on a deep LSTM setup when comparing on a commonly used subset

[1] The source code will be availabe at https://github.com/facebookresearch/fairseq

[2] For kernel width k and sequence length n we require $\max\left(1, \left\lceil \frac{n-1}{k-1} \right\rceil\right)$ forwards on a succession of stacked convolutional layers compared to n forwards with an RNN.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 123–135
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1012

of the training data (Zhou et al. 2016; §4, §5).

2 Recurrent Neural Machine Translation

The general architecture of the models in this work follows the encoder-decoder approach with soft attention first introduced in (Bahdanau et al., 2015). A source sentence $\mathbf{x} = (x_1, \ldots, x_m)$ of m words is processed by an encoder which outputs a sequence of states $\mathbf{z} = (z_1 \ldots, z_m)$.

The decoder is an RNN network that computes a new hidden state s_{i+1} based on the previous state s_i, an embedding g_i of the previous target language word y_i, as well as a conditional input c_i derived from the encoder output $\mathbf{z}$. We use LSTMs (Hochreiter and Schmidhuber, 1997) for all decoder networks whose state s_i comprises of a cell vector and a hidden vector h_i which is output by the LSTM at each time step. We input c_i into the LSTM by concatenating it to g_i.

The translation model computes a distribution over the V possible target words y_{i+1} by transforming the LSTM output h_i via a linear layer with weights W_o and bias b_o:

$$p(y_{i+1}|y_1, \ldots, y_i, \mathbf{x}) = \text{softmax}(W_o h_{i+1} + b_o)$$

The conditional input c_i at time i is computed via a simple dot-product style attention mechanism (Luong et al., 2015a). Specifically, we transform the decoder hidden state h_i by a linear layer with weights W_d and b_d to match the size of the embedding of the previous target word g_i and then sum the two representations to yield d_i. Conditional input c_i is a weighted sum of attention scores $\mathbf{a_i} \in \mathbb{R}^m$ and encoder outputs $\mathbf{z}$. The attention scores $\mathbf{a_i}$ are determined by a dot product between h_i with each z_j, followed by a softmax over the source sequence:

$$d_i = W_d h_i + b_d + g_i,$$
$$a_{ij} = \frac{\exp\left(d_i^T z_j\right)}{\sum_{t=1}^{m} \exp\left(d_i^T z_t\right)}, \quad c_i = \sum_{j=1}^{m} a_{ij} z_j$$

In preliminary experiments, we did not find the MLP attention of (Bahdanau et al., 2015) to perform significantly better in terms of BLEU nor perplexity. However, we found the dot-product attention to be more favorable in terms of training and evaluation speed.

We use bi-directional LSTMs to implement recurrent encoders similar to (Zhou et al., 2016) which achieved some of the best WMT14 English-French results reported to date. First, each word of the input sequence $\mathbf{x}$ is embedded in distributional space resulting in $\mathbf{e} = (e_1, \ldots, e_m)$. The embeddings are input to two stacks of uni-directional RNNs where the output of each layer is reversed before being fed into the next layer. The first stack takes the original sequence while the second takes the reversed input sequence; the output of the second stack is reversed so that the final outputs of the stacks align. Finally, the top-level hidden states of the two stacks are concatenated and fed into a linear layer to yield $\mathbf{z}$. We denote this encoder architecture as BiLSTM.

3 Non-recurrent Encoders

3.1 Pooling Encoder

A simple baseline for non-recurrent encoders is the pooling model described in (Ranzato et al., 2015) which simply averages the embeddings of k consecutive words. Averaging word embeddings does not convey positional information besides that the words in the input are somewhat close to each other. As a remedy, we add position embeddings to encode the absolute position of each source word within a sentence. Each source embedding e_j therefore contains a position embedding l_j as well as the word embedding w_j. Position embeddings have also been found helpful in memory networks for question-answering and language modeling (Sukhbaatar et al., 2015). Similar to the recurrent encoder (§2), the attention scores a_{ij} are computed from the pooled representations z_j, however, the conditional input c_i is a weighted sum of the embeddings e_j, not z_j, i.e.,

$$e_j = w_j + l_j, \quad z_j = \frac{1}{k} \sum_{t=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} e_{j+t},$$

$$c_i = \sum_{j=1}^{m} a_{ij} e_j$$

The input sequence is padded prior to pooling such that the encoder output matches the input length $|\mathbf{z}| = |\mathbf{x}|$. We set k to 5 in all experiments as (Ranzato et al., 2015).

3.2 Convolutional Encoder

A straightforward extension of pooling is to learn the kernel in a convolutional neural network (CNN). The encoder output z_j contains information about a fixed-sized context depending on the kernel width k but the desired context width may vary. This can

be addressed by stacking several layers of convolutions followed by non-linearities: additional layers increase the total context size while non-linearities can modulate the effective size of the context as needed. For instance, stacking 5 convolutions with kernel width $k = 3$ results in an input field of 11 words, i.e., each output depends on 11 input words, and the non-linearities allow the encoder to exploit the full input field, or to concentrate on fewer words as needed.

To ease learning for deep encoders, we add residual connections from the input of each convolution to the output and then apply the non-linear activation function to the output (tanh; He et al., 2015); the non-linearities are therefore not 'bypassed'. Multi-layer CNNs are constructed by stacking several blocks on top of each other. The CNNs do not contain pooling layers which are commonly used for down-sampling, i.e., the full source sequence length will be retained after the network has been applied. Similar to the pooling model, the convolutional encoder uses position embeddings.

The final encoder consists of two stacked convolutional networks (Figure 1): CNN-a produces the encoder output z_j to compute the attention scores $\mathbf{a_i}$, while the conditional input c_i to the decoder is computed by summing the outputs of CNN-c,

$$z_j = \text{CNN-a}(\mathbf{e})_j, \quad c_i = \sum_{j=1}^{m} a_{ij} \, \text{CNN-c}(\mathbf{e})_j.$$

In practice, we found that two different CNNs resulted in better perplexity as well as BLEU compared to using a single one (§5.3). We also found this to perform better than directly summing the e_i without transformation as for the pooling model.

3.3 Related Work

There are several past attempts to use convolutional encoders for neural machine translation, however, to our knowledge none of them were able to match the performance of recurrent encoders. (Kalchbrenner and Blunsom, 2013) introduce a convolutional sentence encoder in which a multi-layer CNN generates a fixed sized embedding for a source sentence, or an n-gram representation followed by transposed convolutions for directly generating a per-token decoder input. The latter requires the length of the translation prior to generation and both models were evaluated by rescoring the output of an existing translation system. (Cho et al., 2014a) propose a gated recursive CNN which is repeatedly applied until a fixed-size representation is obtained but the recurrent encoder achieves higher accuracy. In follow-up work, the authors improved the model via a soft-attention mechanism but did not reconsider convolutional encoder models (Bahdanau et al., 2015).

Concurrently to our work, (Kalchbrenner et al., 2016) have introduced convolutional translation models without an explicit attention mechanism but their approach does not yet result in state-of-the-art accuracy. (Lamb and Xie, 2016) also proposed a multi-layer CNN to generate a fixed-size encoder representation but their work lacks quantitative evaluation in terms of BLEU. Meng et al. (2015) and (Tu et al., 2015) applied convolutional models to score phrase-pairs of traditional phrase-based and dependency-based translation models. Convolutional architectures have also been successful in language modeling but so far failed to outperform LSTMs (Pham et al., 2016).

4 Experimental Setup

4.1 Datasets

We evaluate different encoders and ablate architectural choices on a small dataset from the German-English machine translation track of IWSLT 2014 (Cettolo et al., 2014) with a similar setting to (Ranzato et al., 2015). Unless otherwise stated, we restrict training sentences to have no more than 175 words; test sentences are not filtered. This is a higher threshold compared to other publications but ensures proper training of the position embeddings for non-recurrent encoders; the length threshold did not significantly effect recurrent encoders. Length filtering results in 167K sentence pairs and we test on the concatenation of *tst2010, tst2011, tst2012, tst2013* and *dev2010* comprising 6948 sentence pairs.[3] Our final results are on three major WMT tasks:

WMT'16 English-Romanian. We use the same data and pre-processing as (Sennrich et al., 2016a) and train on 2.8M sentence pairs.[4] Our model is word-based instead of relying on byte-pair encoding (Sennrich et al., 2016b). We evaluate on *newstest2016*.

WMT'15 English-German. We use all available parallel training data, namely Europarl v7, Com-

[3] Different to the other datasets, we lowercase the training data and evaluate with case-insensitive BLEU.

[4] We followed the pre-processing of https://github.com/rsennrich/wmt16-scripts/blob/master/sample/preprocess.sh and added the back-translated data from http://data.statmt.org/rsennrich/wmt16_backtranslations/en-ro.

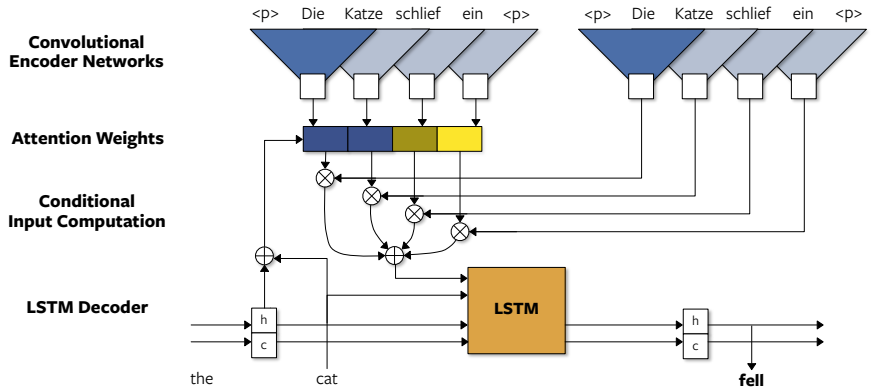


Figure 1: Neural machine translation model with single-layer convolutional encoder networks. CNN-a is on the left and CNN-c is at the right. Embedding layers are not shown.

mon Crawl and News Commentary v10 and apply the standard Moses tokenization to obtain 3.9M sentence pairs (Koehn et al., 2007). We report results on *newstest2015*.

WMT'14 English-French. We use a commonly used subset of 12M sentence pairs (Schwenk, 2014), and remove sentences longer than 150 words. This results in 10.7M sentence-pairs for training. Results are reported on *ntst14*.

A small subset of the training data serves as validation set (5% for IWSLT'14 and 1% for WMT) for early stopping and learning rate annealing (§4.3). For IWSLT'14, we replace words that occur fewer than 3 times with a <unk> symbol, which results in a vocabulary of 24158 English and 35882 German word types. For WMT datasets, we retain 200K source and 80K target words. For English-French only, we set the target vocabulary to 30K types to be comparable with previous work.

4.2 Model parameters

We use 512 hidden units for both recurrent encoders and decoders. We reset the decoder hidden states to zero between sentences. For the convolutional encoder, 512 hidden units are used for each layer in CNN-a, while layers in CNN-c contain 256 units each. All embeddings, including the output produced by the decoder before the final linear layer, are of 256 dimensions. On the WMT corpora, we find that we can improve the performance of the bidirectional LSTM models (BiLSTM) by using 512-dimensional word embeddings.

Model weights are initialized from a uniform distribution within $[-0.05, 0.05]$. For convolutional layers, we use a uniform distribution of $\left[-kd^{-0.5}, kd^{-0.5}\right]$, where k is the kernel width (we use 3 throughout this work) and d is the input size for the first layer and the number of hidden units for subsequent layers (Collobert et al., 2011b). For CNN-c, we transform the input and output with a linear layer each to match the smaller embedding size. The model parameters were tuned on IWSLT'14 and cross-validated on the larger WMT corpora.

4.3 Optimization

Recurrent models are trained with Adam as we found them to benefit from aggressive optimization. We use a step width of $3.125 \cdot 10^{-4}$ and early stopping based on validation perplexity (Kingma and Ba, 2014). For non-recurrent encoders, we obtain best results with stochastic gradient descent (SGD) and annealing: we use a learning rate of 0.1 and once the validation perplexity stops improving, we reduce the learning rate by an order of magnitude each epoch until it falls below 10^{-4}.

For all models, we use mini-batches of 32 sentences for IWSLT'14 and 64 for WMT. We use truncated back-propagation through time to limit the length of target sequences per mini-batch to 25 words. Gradients are normalized by the mini-batch size. We re-normalize the gradients if their norm exceeds 25 (Pascanu et al., 2013). Gradients of convolutional layers are scaled by $\mathrm{sqrt}(\dim(input))^{-1}$ similar to (Collobert et al., 2011b). We use dropout on the embeddings and decoder outputs h_i with a rate of 0.2 for IWSLT'14 and 0.1 for WMT (Srivastava et al., 2014). All models are implemented in Torch (Collobert et al., 2011a) and trained on a single GPU.

4.4 Evaluation

We report accuracy of single systems by training several identical models with different ran-

dom seeds (5 for IWSLT'14, 3 for WMT) and pick the one with the best validation perplexity for final BLEU evaluation. Translations are generated by a beam search and we normalize log-likelihood scores by sentence length. On IWSLT'14 we use a beam width of 10 and for WMT models we tune beam width and word penalty on a separate test set, that is *newsdev2016* for WMT'16 English-Romanian, *newstest2014* for WMT'15 English-German and *ntst1213* for WMT'14 English-French.[5] The word penalty adds a constant factor to log-likelihoods, except for the end-of-sentence token.

Prior to scoring the generated translations against the respective references, we perform unknown word replacement based on attention scores (Jean et al., 2015). Unknown words are replaced by looking up the source word with the maximum attention score in a pre-computed dictionary. If the dictionary contains no translation, then we simply copy the source word. Dictionaries were extracted from the aligned training data that was aligned with `fast_align` (Dyer et al., 2013). Each source word is mapped to the target word it is most frequently aligned to.

For convolutional encoders with stacked CNN-c layers we noticed for some models that the attention maxima were consistently shifted by one word. We determine this per-model offset on the above-mentioned development sets and correct for it. Finally, we compute case-sensitive tokenized BLEU, except for WMT'16 English-Romanian where we use *detokenized* BLEU to be comparable with Sennrich et al. (2016a).[6]

5 Results

5.1 Recurrent vs. Non-recurrent Encoders

We first compare recurrent and non-recurrent encoders in terms of perplexity and BLEU on IWSLT'14 with and without position embeddings (§3.1) and include a phrase-based system (Koehn et al., 2007). Table 1 shows that a single-layer convolutional model with position embeddings (Convolutional) can outperform both a uni-directional LSTM encoder (LSTM) as well as a bi-directional LSTM encoder (BiLSTM). Next, we increase the depth of the convolutional encoder. We choose a good setting by independently varying the number of layers in CNN-a and CNN-c between 1 and 10 and obtained best validation set perplexity with six layers for CNN-a and three layers for CNN-c. This configuration outperforms BiLSTM by 0.7 BLEU (Deep Convolutional 6/3). We investigate depth in the convolutional encoder more in §5.3.

System/Encoder	BLEU wrd+pos	BLEU wrd	PPL wrd+pos
Phrase-based	–	28.4	–
LSTM	27.4	27.3	10.8
BiLSTM	29.7	29.8	9.9
Pooling	26.1	19.7	11.0
Convolutional	29.9	20.1	9.1
Deep Convolutional 6/3	30.4	25.2	8.9

Table 1: Accuracy of encoders with position features (wrd+pos) and without (wrd) in terms of BLEU and perplexity (PPL) on IWSLT'14 German to English translation; results include unknown word replacement. Deep Convolutional 6/3 is the only multi-layer configuration, more layers for the LSTMs did not improve accuracy on this dataset.

Among recurrent encoders, the BiLSTM is 2.3 BLEU better than the uni-directional version. The simple pooling encoder which does not contain any parameters is only 1.3 BLEU lower than a uni-directional LSTM encoder and 3.6 BLEU lower than BiLSTM. The results without position embeddings (words) show that position information is crucial for convolutional encoders. In particular for shallow models (Pooling and Convolutional), whereas deeper models are less effected. Recurrent encoders do not benefit from explicit position information because this information can be naturally extracted through the sequential computation.

When tuning model settings, we generally observe good correlation between perplexity and BLEU. However, for convolutional encoders perplexity gains translate to smaller BLEU improvements compared to recurrent counterparts (Table 1). We observe a similar trend on larger datasets.

5.2 Evaluation on WMT Corpora

Next, we evaluate the BiLSTM encoder and the convolutional encoder architecture on three larger tasks and compare against previously published results. On WMT'16 English-Romanian translation we compare to (Sennrich et al., 2016a), the winning single system entry for this language pair. Their model consists of a bi-directional GRU encoder, a GRU decoder and MLP-based attention.

[5] Specifically, we select a beam from $\{5, 10\}$ and a word penalty from $\{0, -0.5, -1, -1.5\}$

[6] https://github.com/moses-smt/mosesdecoder/blob/617e8c8ed1630fb1d1/scripts/generic/{multi-bleu.perl, mteval-v13a.pl}

WMT'16 English-Romanian	Encoder	Vocabulary	BLEU
(Sennrich et al., 2016a)	BiGRU	BPE 90K	28.1
Single-layer decoder	BiLSTM	80K	27.5
	Convolutional	80K	27.1
	Deep Convolutional 8/4	80K	27.8
WMT'15 English-German	**Encoder**	**Vocabulary**	**BLEU**
(Jean et al., 2015) RNNsearch-LV	BiGRU	500K	22.4
(Chung et al., 2016) BPE-Char	BiGRU	Char 500	23.9
(Yang et al., 2016) RNNSearch + UNK replace	BiLSTM	50K	24.3
+ *recurrent attention*	BiLSTM	50K	25.0
Single-layer decoder	BiLSTM	80K	23.5
	Deep Convolutional 8/4	80K	23.6
Two-layer decoder	Two-layer BiLSTM	80K	24.1
	Deep Convolutional 15/5	80K	24.2
WMT'14 English-French (12M)	**Encoder**	**Vocabulary**	**BLEU**
(Bahdanau et al., 2015) RNNsearch	BiGRU	30K	28.5
(Luong et al., 2015b) Single LSTM	6-layer LSTM	40K	32.7
(Jean et al., 2014) RNNsearch-LV	BiGRU	500K	34.6
(Zhou et al., 2016) Deep-Att	Deep BiLSTM	30K	35.9
Single-layer decoder	BiLSTM	30K	34.3
	Deep Convolutional 8/4	30K	34.6
Two-layer decoder	2-layer BiLSTM	30K	35.3
	Deep Convolutional 20/5	30K	35.7

Table 2: Accuracy on three WMT tasks, including results published in previous work. For deep convolutional encoders, we include the number of layers in CNN-a and CNN-c, respectively.

They use byte pair encoding (BPE) to achieve open-vocabulary translation and dropout in all components of the neural network to achieve 28.1 BLEU; we use the same pre-processing but no BPE (§4).

The results (Table 2) show that a deep convolutional encoder can perform competitively to the state of the art on this dataset (Sennrich et al., 2016a). Our bi-directional LSTM encoder baseline is 0.6 BLEU lower than the state of the art but uses only 512 hidden units compared to 1024. A single-layer convolutional encoder with embedding size 256 performs at 27.1 BLEU. Increasing the number of convolutional layers to 8 in CNN-a and 4 in CNN-c achieves 27.8 BLEU which outperforms our baseline and is competitive to the state of the art.

On WMT'15 English to German, we compare to a BiLSTM baseline and prior work: (Jean et al., 2015) introduce a large output vocabulary; the decoder of (Chung et al., 2016) operates on the character-level; (Yang et al., 2016) uses LSTMs instead of GRUs and feeds the conditional input to the output layer as well as to the decoder.

Our single-layer BiLSTM baseline is competitive to prior work and a two-layer BiLSTM encoder performs 0.6 BLEU better at 24.1 BLEU. Previous work also used multi-layer setups, e.g., (Chung et al., 2016) has two layers both in the encoder and the decoder with 1024 hidden units, and (Yang et al., 2016) use 1000 hidden units per LSTM. We use 512 hidden units for both LSTM and convolutional encoders. Our convolutional model with either 8 or 15 layers in CNN-a outperform the BiLSTM encoder with both a single decoder layer or two decoder layers.

Finally, we evaluate on the larger WMT'14 English-French corpus. On this dataset the recurrent architectures benefit from an additional layer both in the encoder and the decoder. For a single-layer decoder, a deep convolutional encoder outperforms the BiLSTM accuracy by 0.3 BLEU and for a two-layer decoder, our very deep convolutional encoder with up to 20 layers outperforms the BiLSTM by 0.4 BLEU. It has 40% fewer parameters than the BiLSTM due to the smaller embedding sizes. We also outperform several previous systems, including the very deep encoder-decoder model proposed by (Luong et al., 2015a). Our best result is just 0.2 BLEU below (Zhou et al., 2016) who use a very deep LSTM setup with a 9-layer encoder, a 7-layer decoder, shortcut connections and extensive regularization with dropout and L2 regularization.

5.3 Convolutional Encoder Architecture Details

We next motivate our design of the convolutional encoder (§3.2). We use the smaller IWSLT'14 German-English setup without unknown word replacement to enable fast experimental turn-around. BLEU results are averaged over three training runs initialized with different seeds.

Figure 2 shows accuracy for a different number of layers of both CNNs with and without residual connections. Our first observation is that computing the conditional input c_i directly over embeddings $\mathbf{e}$ (line "without CNN-c") is already working well at 28.3 BLEU with a single CNN-a layer and at 29.1 BLEU for CNN-a with 7 layers (Figure 2a). Increasing the number of CNN-c layers is beneficial up to three layers and beyond this we did not observe further improvements. Similarly, increasing the number of layers in CNN-a beyond six does not increase accuracy on this relatively small dataset. In general, choosing two to three times as many layers in CNN-a as in CNN-c is a good rule of thumb. Without residual connections, the model fails to utilize the increase in modeling power from additional layers, and performance drops significantly for deeper encoders (Figure 2b).

Our convolutional architecture relies on two sets of networks, CNN-a for attention score computation $\mathbf{a_i}$ and CNN-c for the conditional input c_i to be fed to the decoder. We found that using the same network for both tasks, similar to recurrent encoders, resulted in poor accuracy of 22.9 BLEU. This compares to 28.5 BLEU for separate single-layer networks, or 28.3 BLEU when aggregating embeddings for c_i. Increasing the number of layers in the single network setup did not help. Figure 2(a) suggests that the attention weights (CNN-a) need to integrate information from a wide context which can be done with a deep stack. At the same time, the vectors which are averaged (CNN-c) seem to benefit from a shallower, more local representation closer to the input words. Two stacks are an easy way to achieve these contradicting requirements.

In Appendix A we visualize attention scores and find that alignments for CNN encoders are less sharp compared to BiLSTMs, however, this does not affect the effectiveness of unknown word replacement once we adjust for shifted maxima. In Appendix B we investigate whether deep convolutional encoders are required for translating long sentences and observe that even relatively shallow encoders perform well on long sentences.

5.4 Training and Generation Speed

For training, we use the fast CuDNN LSTM implementation for layers without attention and experiment on IWSLT'14 with batch size 32. The single-layer BiLSTM model trains at 4300 target words/second, while the 6/3 deep convolutional encoder compares at 6400 words/second on an NVidia Tesla M40 GPU. We do not observe shorter overall training time since SGD converges slower than Adam which we use for BiLSTM models.

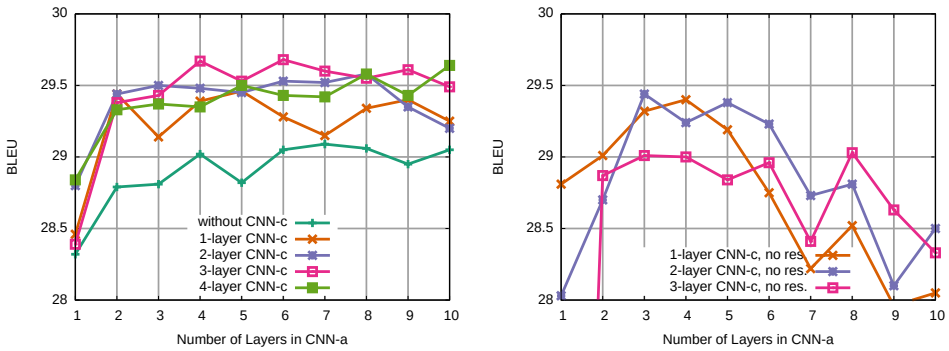
We measure generation speed on an Intel Haswell CPU clocked at 2.50GHz with a single thread for BLAS operations. We use vocabulary selection which can speed up generation by up to a factor of ten at no cost in accuracy via making the time to compute the final output layer negligible (Mi et al., 2016; L'Hostis et al., 2016). This shifts the focus from the efficiency of the encoder to the efficiency of the decoder. On IWSLT'14 (Table 3a) the convolutional encoder increases the speed of the overall model by a factor of 1.35 compared to the BiLSTM encoder while improving accuracy by 0.7 BLEU. In this setup both encoders models have the same hidden layer and embedding sizes.

On the larger WMT'15 English-German task (Table 3b) the convolutional encoder speeds up generation by 2.1 times compared to a two-layer BiLSTM. This corresponds to 231 source words/second with beam size 5. Our best model on this dataset generates 203 words/second but at slightly lower accuracy compared to the full vocabulary setting in Table 2. The recurrent encoder uses larger embeddings than the convolutional encoder which were required for the models to match in accuracy.

The smaller embedding size is not the only reason for the speed-up. In Table 3a (a), we compare a Conv 6/3 encoder and a BiLSTM with equal embedding sizes. The convolutional encoder is still 1.34x faster (at 0.7 higher BLEU) although it requires roughly 1.6x as many FLOPs. We believe that this is likely due to better cache locality for convolutional layers on CPUs: an LSTM with fused gates[7] requires two big matrix multiplications with different weights as well as additions, multiplications and non-linearities for each source word, while the output of each convolutional layer can be computed as whole with a single matrix multiply.

For comparison, the quantized deep LSTM-

[7] Our bi-directional LSTM implementation is based on torch rnnlib which uses fused LSTM gates (https://github.com/facebookresearch/torch-rnnlib/) and which we consider an efficient implementation.



(a) With residual connections

(b) Without residual connections

Figure 2: Effect of encoder depth on IWSLT'14 with and without residual connections. The x-axis varies the number of layers in CNN-a and curves show different CNN-c settings.

Encoder	Words/s	BLEU
BiLSTM	139.7	22.4
Deep Conv. 6/3	187.9	23.1

(a) IWSLT'14 German-English generation speed on *tst2013* with beam size 10.

Encoder	Words/s	BLEU
2-layer BiLSTM	109.9	23.6
Deep Conv. 8/4	231.1	23.7
Deep Conv. 15/5	203.3	24.0

(b) WMT'15 English-German generation speed on *newstest2015* with beam size 5.

Table 3: Generation speed in source words per second on a single CPU core using vocabulary selection.

based model in (Wu et al., 2016) processes 106.4 words/second for English-French on a CPU with 88 cores and 358.8 words/second on a custom TPU chip. The optimized RNNsearch model and C++ decoder described by (Junczys-Dowmunt et al., 2016) translates 265.3 words/s on a CPU with a similar vocabulary selection technique, computing 16 sentences in parallel, i.e., 16.6 words/s on a single core.

6 Conclusion

We introduced a simple encoder model for neural machine translation based on convolutional networks. This approach is more parallelizable than recurrent networks and provides a shorter path to capture long-range dependencies in the source. We find it essential to use source position embeddings as well as different CNNs for attention score computation and conditional input aggregation.

Our experiments show that convolutional encoders perform on par or better than baselines based on bi-directional LSTM encoders. In comparison to other recent work, our deep convolutional encoder is competitive to the best published results to date (WMT'16 English-Romanian) which are obtained with significantly more complex models (WMT'14 English-French) or stem from improvements that are orthogonal to our work (WMT'15 English-German). Our architecture also leads to large generation speed improvements: translation models with our convolutional encoder can translate twice as fast as strong baselines with bi-directional recurrent encoders.

Future work includes better training to enable faster convergence with the convolutional encoder to better leverage the higher processing speed. Our fast architecture is interesting for character level encoders where the input is significantly longer than for words. Also, we plan to investigate the effectiveness of our architecture on other sequence-to-sequence tasks, e.g. summarization, constituency parsing, dialog modeling.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

James Bradbury and Richard Socher. 2016. MetaMind Neural Machine Translation System for WMT 2016. In *Proc. of WMT*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proc. of IWSLT*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the Properties of Neural Machine Translation: Encoder-decoder Approaches. In *Proc. of SSST*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proc. of EMNLP*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A Character-level Decoder without Explicit Segmentation for Neural Machine Translation. *arXiv preprint arXiv:1603.06147* .

Ronan Collobert, Koray Kavukcuoglu, and Clement Farabet. 2011a. Torch7: A Matlab-like Environment for Machine Learning. In *BigLearn, NIPS Workshop*. http://torch.ch.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011b. Natural Language Processing (almost) from scratch. *JMLR* 12(Aug):2493–2537.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A Simple, Fast, and Effective Reparameterization of IBM Model 2. Proc. of ACL.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*.

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

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On Using Very Large Target Vocabulary for Neural Machine Translation. *arXiv preprint arXiv:1412.2007v2* .

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal Neural Machine Translation systems for WMT15. In *Proc. of WMT*. pages 134–140.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is Neural Machine Translation Ready for Deployment? A Case Study on 30 Translation Directions. *arXiv preprint arXiv:1610.01108* .

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proc. of EMNLP*.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural Machine Translation in Linear Time. *arXiv* .

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *Proc. of ICLR* .

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL*.

Andrew Lamb and Michael Xie. 2016. Convolutional Encoders for Neural Machine Translation. https://cs224d.stanford.edu/reports/LambAndrew.pdf. Accessed: 2010-10-31.

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary Selection Strategies for Neural Machine Translation. *arXiv preprint arXiv:1610.00072* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the Rare Word Problem in Neural Machine Translation. In *Proc. of ACL*.

Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. 2015. Encoding Source Language with Convolutional Neural Network for Machine Translation. In *Proc. of ACL*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Vocabulary Manipulation for Neural Machine Translation. *arXiv preprint arXiv:1605.03209* .

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the Difficulty of Training Recurrent Neural Networks. *ICML (3)* 28:1310–1318.

Ngoc-Quan Pham, Germn Kruszewski, and Gemma Boleda. 2016. Convolutional Neural Network Language Models. In *Proc. of EMNLP*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level Training with Recurrent Neural Networks. In *Proc. of ICLR*.

Holger Schwenk. 2014. http://www-lium.univ-lemans.fr/~schwenk/cslm_joint_paper/. Accessed: 2016-10-15.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural Machine Translation of Rare Words with Subword Units. In *Proc. of ACL*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent Neural Networks from overfitting. *JMLR* 15:1929–1958.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, and Arthur Szlam. 2015. End-to-end Memory Networks. In *Proc. of NIPS*. pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. of NIPS*. pages 3104–3112.

Zhaopeng Tu, Baotian Hu, Zhengdong Lu, and Hang Li. 2015. Context-dependent Translation selection using Convolutional Neural Network. In *Proc. of ACL-IJCNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* .

Zichao Yang, Zhiting Hu, Yuntian Deng, Chris Dyer, and Alex Smola. 2016. Neural Machine Translation with Recurrent Attention Modeling. *arXiv preprint arXiv:1607.05108* .

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation. *arXiv preprint arXiv:1606.04199* .

A Alignment Visualization

In Figure 4 and Figure 5, we plot attention scores for a sample WMT'15 English-German and WMT'14 English-French translation with BiLSTM and deep convolutional encoders. The translation is on the x-axis and the source sentence on the y-axis.

The attention scores of the BiLSTM output are sharp but do not necessarily represent a correct alignment. For CNN encoders the scores are less focused but still indicate an approximate source location, e.g., in Figure 4b, when moving the clause "over 1,000 people were taken hostage" to the back of the translation. For some models, attention maxima are consistently shifted by one token as both in Figure 4b and Figure 5b.

Interestingly, convolutional encoders tend to focus on the last token (Figure 4b) or both the first and last tokens (Figure 5b). Motivated by the hypothesis that the this may be due to the decoder depending on the length of the source sentence (which it cannot determine without position embeddings), we explicitly provided a distributed representation of the input length to the decoder and attention module. However, this did not cause a change in attention patterns nor did it improve translation accuracy.

B Performance by Sentence Length

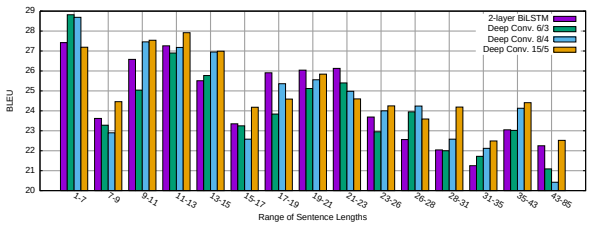
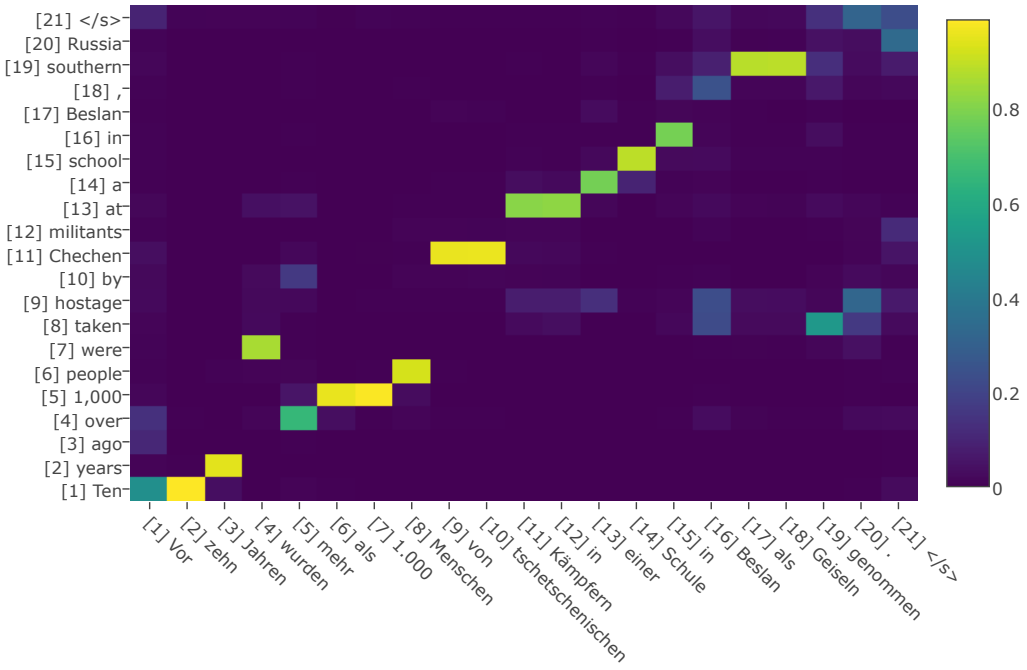


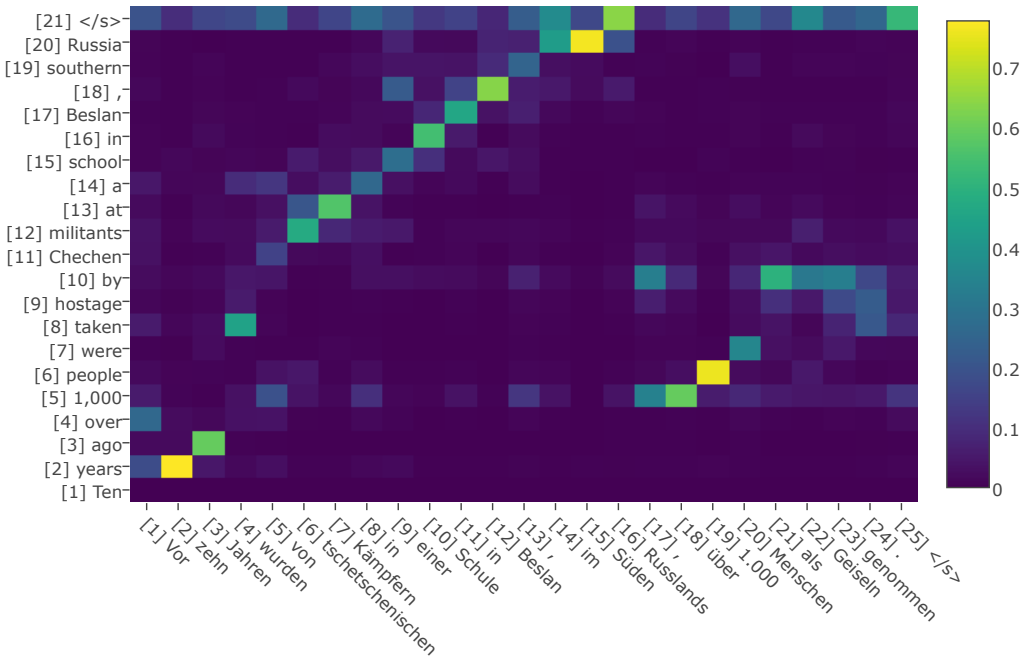
Figure 3: BLEU per sentence length on WMT'15 English-German *newstest2015*. The test set is partitioned into 15 equally-sized buckets according to source sentence length.

One characteristic of our convolutional encoder architecture is that the context over which outputs are computed depends on the number of layers. With bi-directional RNNs, every encoder output depends on the entire source sentence. In Figure 3, we evaluate whether limited context affects the translation quality on longer sentences of WMT'15 English-German which often requires moving verbs over long distances. We sort the *newstest2015* test set by source length, partition it into 15 equally-sized buckets, and compare the BLEU scores of models listed in Table 2 on a per-bucket basis.

There is no clear evidence for sub-par translations on sentences that are longer than the observable context per encoder output. We include a small encoder with a 6-layer CNN-c and a 3-layer CNN-a in the comparison which performs worse than a 2-layer BiLSTM (23.3 BLEU vs. 24.1). With 6 convolutional layers at kernel width 3, each encoder output contains information of 13 adjacent source words. Looking at the accuracy for sentences with 15 words or more, this relatively shallow CNN is either on par or better than the BiLSTM for 5 out of 10 buckets; the BiLSTM has access to the entire source context. Similar observations can be made for the deeper convolutional encoders.

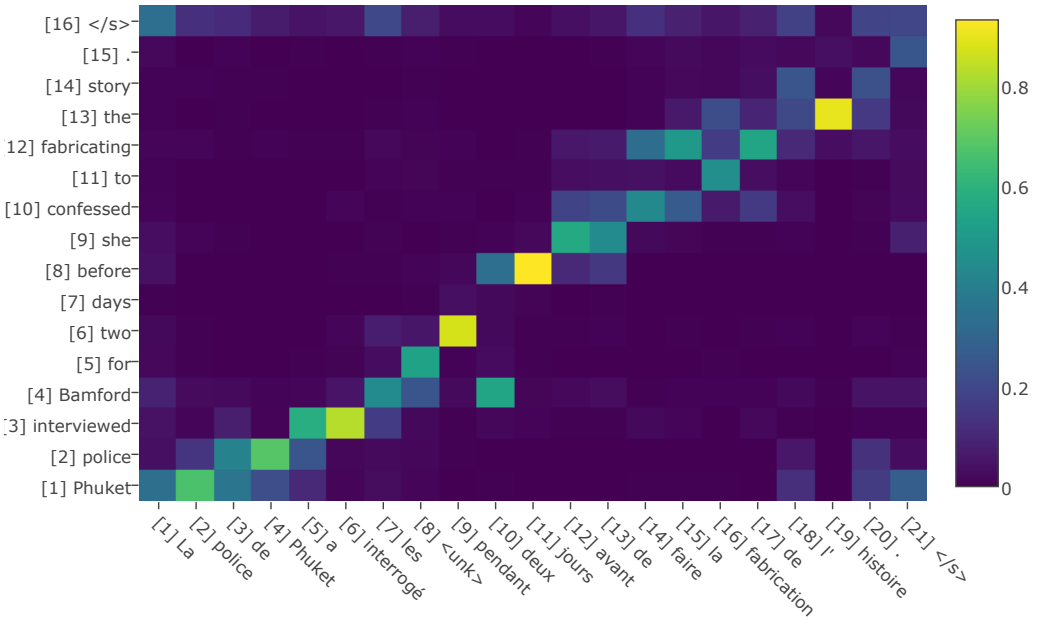


(a) 2-layer BiLSTM encoder.

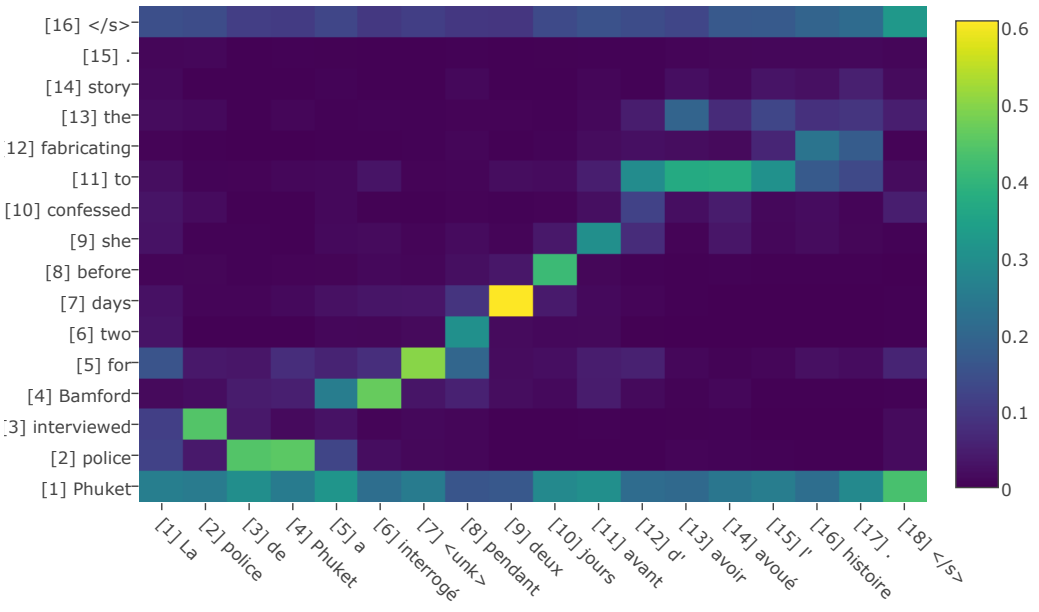


(b) Deep convolutional encoder with 15-layer CNN-a and 5-layer CNN-c.

Figure 4: Attention scores for WMT'15 English-German translation for a sentence of *newstest2015*.



(a) 2-layer BiLSTM encoder.



(b) Deep convolutional encoder with 20-layer CNN-a and 5-layer CNN-c.

Figure 5: Attention scores for WMT'14 English-French translation for a sentence of *ntst14*.

Deep Neural Machine Translation with Linear Associative Unit

Mingxuan Wang[1] **Zhengdong Lu**[2] **Jie Zhou**[2] **Qun Liu**[4,5]
[1]Mobile Internet Group, Tencent Technology Co., Ltd
wangmingxuan@ict.ac.cn
[2]DeeplyCurious.ai
[3] Insititute of Deep Learning Research, Baidu Co., Ltd
[4] Institute of Computing Technology, Chinese Academy of Sciences
[5]ADAPT Centre, School of Computing, Dublin City University

Abstract

Deep Neural Networks (DNNs) have provably enhanced the state-of-the-art Neural Machine Translation (NMT) with their capability in modeling complex functions and capturing complex linguistic structures. However NMT systems with deep architecture in their encoder or decoder RNNs often suffer from severe gradient diffusion due to the non-linear recurrent activations, which often make the optimization much more difficult. To address this problem we propose novel linear associative units (LAU) to reduce the gradient propagation length inside the recurrent unit. Different from conventional approaches (LSTM unit and GRU), LAUs utilizes linear associative connections between input and output of the recurrent unit, which allows unimpeded information flow through both space and time direction. The model is quite simple, but it is surprisingly effective. Our empirical study on Chinese-English translation shows that our model with proper configuration can improve by 11.7 BLEU upon Groundhog and the best reported results in the same setting. On WMT14 English-German task and a larger WMT14 English-French task, our model achieves comparable results with the state-of-the-art.

1 Introduction

Neural Machine Translation (NMT) is an end-to-end learning approach to machine translation which has recently shown promising results on multiple language pairs (Luong et al., 2015; Shen et al., 2015; Wu et al., 2016; Zhang et al., 2016; Tu et al., 2016; Zhang and Zong, 2016; Jean et al., 2015; Meng et al., 2015). Unlike conventional Statistical Machine Translation (SMT) systems (Koehn et al., 2003; Chiang, 2005; Liu et al., 2006; Xiong et al., 2006; Mi et al., 2008) which consist of multiple separately tuned components, NMT aims at building upon a single and large neural network to directly map input text to associated output text. Typical NMT models consists of two recurrent neural networks (RNNs), an encoder to read and encode the input text into a distributed representation and a decoder to generate translated text conditioned on the input representation (Sutskever et al., 2014; Bahdanau et al., 2014).

Driven by the breakthrough achieved in computer vision (He et al., 2015; Srivastava et al., 2015), research in NMT has recently turned towards studying Deep Neural Networks (DNNs). Wu et al. (2016) and Zhou et al. (2016) found that deep architectures in both the encoder and decoder are essential for capturing subtle irregularities in the source and target languages. However, training a deep neural network is not as simple as stacking layers. Optimization often becomes increasingly difficult with more layers. One reasonable explanation is the notorious problem of vanishing/exploding gradients which was first studied in the context of vanilla RNNs (Pascanu et al., 2013b). Most prevalent approaches to solve this problem rely on short-cut connections between adjacent layers such as residual or fast-forward connections (He et al., 2015; Srivastava et al., 2015; Zhou et al., 2016). Differ-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 136–145
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1013

ent from previous work, we choose to reduce the gradient path inside the recurrent units and propose a novel Linear Associative Unit (LAU) which creates a fusion of both linear and nonlinear transformations of the input. Through this design, information can flow across several steps both in time and in space with little attenuation. The mechanism makes it easy to train deep stack RNNs which can efficiently capture the complex inherent structures of sentences for NMT. Based on LAUs, we also propose a NMT model , called DEEPLAU, with deep architecture in both the encoder and decoder.

Although DEEPLAU is fairly simple, it gives remarkable empirical results. On the NIST Chinese-English task, DEEPLAU with proper settings yields the best reported result and also a 4.9 BLEU improvement over a strong NMT baseline with most known techniques (e.g, dropout) incorporated. On WMT English-German and English-French tasks, it also achieves performance superior or comparable to the state-of-the-art.

2 Neural machine translation

A typical neural machine translation system is a single and large neural network which directly models the conditional probability $p(\mathbf{y}|\mathbf{x})$ of translating a source sentence $\mathbf{x} = \{x_1, x_2, \cdots, x_{T_x}\}$ to a target sentence $\mathbf{y} = \{y_1, y_2, \cdots, y_{T_y}\}$.

Attention-based NMT, with RNNsearch as its most popular representative, generalizes the conventional notion of encoder-decoder in using an array of vectors to represent the source sentence and dynamically addressing the relevant segments of them during decoding. The process can be explicitly split into an encoding part, a decoding part and an attention mechanism. The model first encodes the source sentence $\mathbf{x}$ into a sequence of vectors $\mathbf{c} = \{h_1, h_2, \cdots, h_{T_x}\}$. In general, h_i is the annotation of x_i from a bi-directional RNN which contains information about the whole sentence with a strong focus on the parts of x_i. Then, the RNNsearch model decodes and generates the target translation $\mathbf{y}$ based on the context $\mathbf{c}$ and the partial traslated sequence $\mathbf{y}_{<t}$ by maximizing the probability of $p(y_i|y_{<i}, \mathbf{c})$. In the attention model, $\mathbf{c}$ is dynamically obtained according to the contribution of the source annotation made to the word prediction. This is called automatic alignment (Bahdanau et al., 2014) or attention mechanism (Luong et al., 2015), but it is essentially reading with content-based addressing defined in (Graves et al., 2014). With this addressing strategy the decoder can attend to the source representation that is most relevant to the stage of decoding.

Deep neural models have recently achieved a great success in a wide range of problems. In computer vision, models with more than 100 convolutional layers have outperformed shallow ones by a big margin on a series of image tasks (He et al., 2015; Srivastava et al., 2015). Following similar ideas of building deep CNNs, some promising improvements have also been achieved on building deep NMT systems. Zhou et al. (2016) proposed a new type of linear connections between adjacent layers to simplify the training of deeply stacked RNNs. Similarly, Wu et al. (2016) introduced residual connections to their deep neural machine translation system and achieve great improvements. However the optimization of deep RNNs is still an open problem due to the massive recurrent computation which makes the gradient propagation path extremely tortuous.

3 Model Description

In this section, we discuss Linear Associative Unit (LAU) to ease the training of deep stack of RNNs. Based on this idea, we further propose DEEPLAU, a neural machine translation model with a deep encoder and decoder.

3.1 Recurrent Layers

A recurrent neural network (Williams and Zipser, 1989) is a class of neural network that has recurrent connections and a state (or its more sophisticated memory-like extension). The past information is built up through the recurrent connections. This makes RNN applicable for sequential prediction tasks of arbitrary length. Given a sequence of vectors $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T\}$ as input, a standard RNN computes the sequence hidden states $\mathbf{h} = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T\}$ by iterating the following

equation from $t = 1$ to $t = T$:

$$\mathbf{h}_t = \phi(\mathbf{x}_t, \mathbf{h}_{t-1}) \quad (1)$$

ϕ is usually a nonlinear function such as composition of a logistic sigmoid with an affine transformation.

3.2 Gated Recurrent Unit

It is difficult to train RNNs to capture long-term dependencies because the gradients tend to either vanish (most of the time) or explode. The effect of long-term dependencies is dropped exponentially with respect to the gradient propagation length. The problem was explored in depth by (Hochreiter and Schmidhuber, 1997; Pascanu et al., 2013b). A successful approach is to design a more sophisticated activation function than a usual activation function consisting of gating functions to control the information flow and reduce the propagation path. There is a long thread of work aiming to solve this problem, with the long short-term memory units (LSTM) being the most salient examples and gated recurrent unit (GRU) being the most recent one (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). RNNs employing either of these recurrent units have been shown to perform well in tasks that require capturing long-term dependencies.

GRU can be viewed as a slightly more dramatic variation on LSTM with fewer parameters. The activation function is armed with two specifically designed gates called update and reset gates to control the flow of information inside each hidden unit. Each hidden state at time-step t is computed as follows

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \quad (2)$$

where $\odot$ is an element-wise product, $\mathbf{z}_t$ is the update gate, and $\tilde{\mathbf{h}}_t$ is the candidate activation.

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \quad (3)$$

where $\mathbf{r}_t$ is the reset gate. Both reset and update gates are computed as :

$$\mathbf{r}_t = \sigma(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1}) \quad (4)$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1}) \quad (5)$$

This procedure of taking a linear sum between the existing state and the newly computed state is similar to the LSTM unit.

3.3 Linear Associative Unit

GRU can actually be viewed as a non-linear activation function with gating mechanism. Here we propose LAU which extends GRU by having an additional linear transformation of the input in its dynamics. More formally, the state update function becomes

$$\begin{aligned}\mathbf{h}_t =&((1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t) \odot (1 - \mathbf{g}_t) \\ &+ \mathbf{g}_t \odot \mathbf{H}(\mathbf{x}_t).\end{aligned} \quad (6)$$

Here the updated $\mathbf{h}_t$ has three sources: 1) the direct transfer from previous state $\mathbf{h}_{t-1}$, 2) the candidate update $\tilde{\mathbf{h}}_t$, and 3) a direct contribution from the input $\mathbf{H}(\mathbf{x}_t)$. More specifically, $\tilde{\mathbf{h}}_t$ contains the nonlinear information of the input and the previous hidden state.

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{f}_t \odot (\mathbf{W}_{xh}\mathbf{x}_t) + \mathbf{r}_t \odot (\mathbf{W}_{hh}\mathbf{h}_{t-1})), \quad (7)$$

where $\mathbf{f}_t$ and $\mathbf{r}_t$ express how much of the nonlinear abstraction are produced by the input $\mathbf{x}_t$ and previous hidden state $\mathbf{h}_t$, respectively. For simplicity, we set $\mathbf{f}_t = 1 - \mathbf{r}_t$ in this paper and find that this works well in our experiments. The term $\mathbf{H}(\mathbf{x}_t)$ is usually an affine linear transformation of the input $\mathbf{x}_t$ to mach the dimensions of $\mathbf{h}_t$, where $\mathbf{H}(\mathbf{x}_t) = \mathbf{W}_x x_t$. The associated term $\mathbf{g}_t$ (the input gate) decides how much of the linear transformation of the input is carried to the hidden state and then the output. The gating function $\mathbf{r}_t$ (reset gate) and $\mathbf{z}_t$ (update gate) are computed following Equation (4) and (5) while $\mathbf{g}_t$ is computed as

$$\mathbf{g}_t = \sigma(\mathbf{W}_{xg}\mathbf{x}_t + \mathbf{W}_{hg}\mathbf{h}_{t-1}). \quad (8)$$

The term $\mathbf{g}_t \odot \mathbf{H}(\mathbf{x}_t)$ therefore offers a direct way for input $\mathbf{x}_t$ to go to later hidden layers, which can eventually lead to a path to the output layer when applied recursively. This mechanism is potentially very useful for translation where the input, no matter whether it is the source word or the attentive reading (context), should sometimes be directly carried to the next stage of processing without any substantial composition or nonlinear transformation. To understand this, imagine we want to translate a English sentence containing a relative rare entity name such as "Bahrain" to Chinese: LAU is potentially able to retain the embedding of this word in its hidden state, which

will otherwise be prone to serious distortion due to the scarcity of training instances for it.

3.4 DEEPLAU

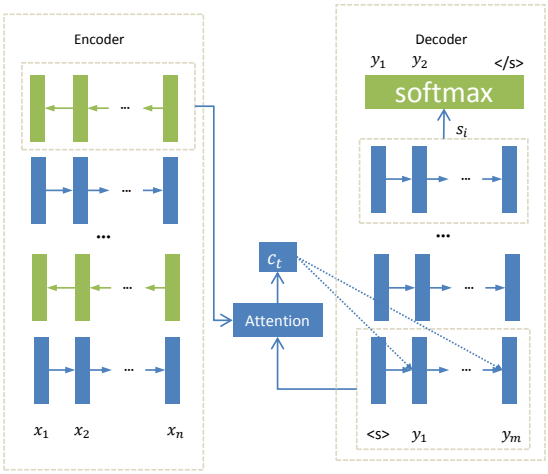


Figure 1: DEEPLAU: a neural machine translation model with deep encoder and decoder.

Graves et al. (2013) explored the advantages of deep RNNs for handwriting recognition and text generation. There are multiple ways of combining one layer of RNN with another. Pascanu et al. (2013a) introduced Deep Transition RNNs with Skip connections (DT(S)-RNNs). Kalchbrenner et al. (2015) proposed to make a full connection of all the RNN hidden layers. In this work we employ vertical stacking where only the output of the previous layer of RNN is fed to the current layer as input. The input at recurrent layer ℓ (denoted as $\mathbf{x}_t^\ell$) is exactly the output of the same time step at layer $\ell - 1$ (denoted as $\mathbf{h}_t^{\ell-1}$). Additionally, in order to learn more temporal dependencies, the sequences can be processed in different directions. More formally, given an input sequence $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_T)$, the output on layer ℓ is

$$\mathbf{h}_t^{(\ell)} = \begin{cases} \mathbf{x}_t, & \ell = 1 \\ \phi^\ell(\mathbf{h}_{t+d}^{(\ell)}, \mathbf{h}_t^{(\ell-1)}), & \ell > 1 \end{cases} \quad (9)$$

where

- $\mathbf{h}_t^{(\ell)}$ gives the output of layer ℓ at location t.
- ϕ is a recurrent function and we choose LAUs in this work.
- The directions are marked by a direction term $d \in \{-1, 1\}$. If we fixed d to -1, the input will be processed in forward direction, otherwise backward direction.

The deep architecture of DEEPLAU, as shown in Figure 1, consists of three parts: a stacked LAU-based encoder, a stacked LAU-based decoder and an improved attention model.

Encoder One shortcoming of conventional RNNs is that they are only able to make use of previous context. In machine translation, where whole source utterances are transcribed at once, there is no reason not to exploit future context as well. Thus bi-directional RNNs are proposed to integrate information from the past and the future. The typical bidirectional approach processes the raw input in backward and forward direction with two separate layers, and then concatenates them together. Following Zhou et al. (2016), we choose another bidirectional approach to process the sequence in order to learn more temporal dependencies in this work. Specifically, an RNN layer processes the input sequence in forward direction. The output of this layer is taken by an upper RNN layer as input, processed in reverse direction. Formally, following Equation (9), we set $d = (-1)^\ell$. This approach can easily build a deeper network with the same number of parameters compared to the classical approach. The final encoder consists of L_{enc} layers and produces the output $\mathbf{h}^{L_{\text{enc}}}$ to compute the conditional input $\mathbf{c}$ to the decoder.

Attention Model The alignment model $\alpha_{t,j}$ scores how well the output at position t matches the inputs around position j based on $\mathbf{s}_{t-1}^1$ and $\mathbf{h}_j^{L_{\text{enc}}}$ where $\mathbf{h}_j^{L_{\text{enc}}}$ is the top-most layer of the encoder at step j and $\mathbf{s}_{t-1}^1$ is the first layer of decoder at step $t - 1$. It is intuitively beneficial to exploit the information of y_{t-1} when reading from the source sentence representation, which is missing from the implementation of attention-based NMT in (Bahdanau et al., 2014). In this work, we build a more effective alignment path by feeding both the previous hidden state $\mathbf{s}_{t-1}^1$ and the context word y_{t-1} to the attention model, inspired by the recent implementation of attention-based

NMT[1]. The conditional input $\mathbf{c}_j$ is a weighted sum of attention score $\alpha_{t,j}$ and encoder output $\mathbf{h}^{L_{\text{enc}}}$. Formally, the calculation of $\mathbf{c}_j$ is

$$\mathbf{c}_j = \sum_{t=1}^{t=L_x} \alpha_{t,j} \mathbf{h}_t^{L_{\text{enc}}} \tag{10}$$

where

$$\begin{aligned} e_{t,j} &= \mathbf{v}_a^T \sigma(\mathbf{W}_a \mathbf{s}_{t-1}^1 + \mathbf{U}_a \mathbf{h}_j^{L_{\text{enc}}} + \mathbf{W}_y \mathbf{y}_{t-1}) \\ \alpha_{t,j} &= \text{softmax}(e_{t,j}). \end{aligned} \tag{11}$$

σ is a nonlinear function with the information of y_{t-1} (its word embedding being $\mathbf{y}_{t-1}$) added. In our preliminary experiments, we found that GRU works slightly better than tanh function, but we chose the latter for simplicity.

Decoder The decoder follows Equation (9) with fixed direction term $d = -1$. At the first layer, we use the following input:

$$\mathbf{x}_t = [\mathbf{c}_t, \mathbf{y}_{t-1}]$$

where $\mathbf{y}_{t-1}$ is the target word embedding at time step t, $\mathbf{c}_t$ is dynamically obtained follows Equation (10). There are L_{dec} layers of RNNs armed with LAUs in the decoder. At inference stage, we only utilize the top-most hidden states $\mathbf{s}^{L_{\text{dec}}}$ to make the final prediction with a softmax layer:

$$p(y_i | y_{<i}, \mathbf{x}) = \text{softmax}(\mathbf{W}_o \mathbf{s}_i^{L_{\text{dec}}}) \tag{12}$$

.

4 Experiments

4.1 Setup

We mainly evaluated our approaches on the widely used NIST Chinese-English translation task. In order to show the usefulness of our approaches, we also provide results on other two translation tasks: English-French, English-German. The evaluation metric is BLEU[2] (Papineni et al., 2002).

For Chinese-English, our training data consists of 1.25M sentence pairs extracted from LDC corpora[3], with 27.9M Chinese words and 34.5M English words respectively. We choose NIST 2002 (MT02) dataset as our development set, and the NIST 2003 (MT03), 2004 (MT04) 2005 (MT05) and 2006 (MT06) datasets as our test sets.

For English-German, to compare with the results reported by previous work (Luong et al., 2015; Zhou et al., 2016; Jean et al., 2015), we used the same subset of the WMT 2014 training corpus that contains 4.5M sentence pairs with 91M English words and 87M German words. The concatenation of news-test 2012 and news-test 2013 is used as the validation set and news-test 2014 as the test set.

To evaluate at scale, we also report the results of English-French. To compare with the results reported by previous work on end-to-end NMT (Sutskever et al., 2014; Bahdanau et al., 2014; Jean et al., 2015; Luong et al., 2014; Zhou et al., 2016), we used the same subset of the WMT 2014 training corpus that contains 12M sentence pairs with 304M English words and 348M French words. The concatenation of news-test 2012 and news-test 2013 serves as the validation set and news-test 2014 as the test set.

4.2 Training details

Our training procedure and hyper parameter choices are similar to those used by (Bahdanau et al., 2014). In more details, we limit the source and target vocabularies to the most frequent $30K$ words in both Chinese-English and English-French. For English-German, we set the source and target vocabularies size to $120K$ and $80K$, respectively.

For all experiments, the dimensions of word embeddings and recurrent hidden states are both set to 512. The dimension of c_t is also of size 512. Note that our network is more narrow than most previous work where hidden states of dimmention 1024 is used. we initialize parameters by sampling each element from the Gaussian distribution with mean 0 and variance 0.04^2.

Parameter optimization is performed using stochastic gradient descent. Adadelta (Zeiler,

[1] github.com/nyu-dl/dl4mt-tutorial/tree/master/session2

[2] For Chinese-English task, we apply case-insensitive NIST BLEU. For other tasks, we tokenized the reference and evaluated the performance with *multi-bleu.pl*. The metrics are exactly the same as in previous work.

[3] The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

SYSTEM	MT03	MT04	MT05	MT06	AVE.
Existing systems					
Moses	31.61	33.48	30.75	30.85	31.67
Groundhog	31.92	34.09	31.56	31.12	32.17
COVERAGE	34.49	38.34	34.91	34.25	35.49
MEMDEC	36.16	39.81	35.91	35.98	36.95
Our deep NMT systems					
DEEPGRU	33.21	36.76	33.05	33.30	34.08
DEEPLAU	**39.35**	**41.15**	**38.07**	**37.29**	**38.97**
DEEPLAU +Ensemble + PosUnk	42.21	43.85	44.75	42.58	43.35

Table 1: Case-insensitive BLEU scores on Chinese-English translation.

2012) is used to automatically adapt the learning rate of each parameter ($\epsilon = 10^{-6}$ and $\rho = 0.95$). To avoid gradient explosion, the gradients of the cost function which had ℓ_2 norm larger than a predefined threshold τ were normalized to the threshold (Pascanu et al., 2013a). We set τ to 1.0 at the beginning and halve the threshold until the BLEU score does not change much on the development set. Each SGD is a mini-batch of 128 examples. We train our NMT model with the sentences of length up to 80 words in the training data, while for the Moses system we use the full training data. Translations are generated by a beam search and log-likelihood scores are normalized by sentence length. We use a beam width of 10 in all the experiments. Dropout was also applied on the output layer to avoid over-fitting. The dropout rate is set to 0.5. Except when otherwise mentioned, NMT systems are have 4 layers encoders and 4 layers decoders.

4.3 Results on Chinese-English Translation

Table 1 shows BLEU scores on Chinese-English datasets. Clearly DEEPLAU leads to a remarkable improvement over their competitors. Compared to DEEPGRU, DEEPLAU is +4.89 BLEU score higher on average four test sets, showing the modeling power gained from the liner associative connections. We suggest it is because LAUs apply adaptive gate function conditioned on the input which make it able to automatically decide how much linear information should be transferred to the next step.

To show the power of DEEPLAU, we also make a comparison with previous work. Our best single model outperforms both a phrased-based MT system (Moses) as well as an open source attention-based NMT system (Groundhog) by +7.3 and +6.8 BLEU points respectively on average. The result is also better than some other state-of-the-art variants of attention-based NMT mode with big margins. After PosUnk and ensemble, DEEPLAU seizes another notable gain of +4.38 BLEU and outperform Moses by +11.68 BLEU.

4.4 Results on English-German Translation

The results on English-German translation are presented in Table 2. We compare our NMT systems with various other systems including the winning system in WMT14 (Buck et al., 2014), a phrase-based system whose language models were trained on a huge monolingual text, the Common Crawl corpus. For end-to-end NMT systems, to the best of our knowledge, Wu et al. (2016) is currently the SOTA system and about 4 BLEU points on top of previously best reported results even though Zhou et al. (2016) used a much deeper neural network[4].

Following Wu et al. (2016), the BLEU score represents the averaged score of 8 models we trained. Our approach achieves comparable results with SOTA system. As can be seen from the Table 2, DeepLAU performs better than the word based model and even not much worse than the best wordpiece models achieved by Wu et al. (2016). Note that DEEPLAU are sim-

[4]It is also worth mentioning that the result reported by Zhou et al. (2016) does not include PosUnk, and this comparison is not fair enough.

SYSTEM	Architecture	Voc.	BLEU
Existing systems			
Buck et al. (2014)	Winning WMT14 system phrase-based + large LM	-	20.7
Jean et al. (2015)	gated RNN with search + LV + PosUnk	$500K$	19.4
Luong et al. (2015)	LSTM with 4 layers + dropout + local att. + PosUnk	$80K$	20.9
Shen et al. (2015)	gated RNN with search + PosUnk + MRT	$80K$	20.5
Zhou et al. (2016)	LSTM with 16 layers + F-F connections	$80K$	20.6
Wu et al. (2016)	LSTM with 8 laysrs + RL-refined Word	$80K$	23.1
Wu et al. (2016)	LSTM with 8 laysrs + RL-refined WPM-32K	-	24.6
Wu et al. (2016)	LSTM with 8 laysrs + RL-refined WPM-32K + Ensemble	-	26.3
Our deep NMT systems			
this work	DEEPLAU	$80K$	$22.1(\pm 0.3)$
this work	DEEPLAU + PosUnk	$80K$	$23.8(\pm 0.3)$
this work	DEEPLAU + PosUnk + Ensemble 8 models	$80K$	26.1

Table 2: Case-sensitive BLEU scores on German-English translation.

ple and easy to implement, as opposed to previous models reported in Wu et al. (2016), which dependends on some external techniques to achieve their best performance, such as their introduction of length normalization, coverage penalty, fine-tuning and the RL-refined model.

4.5 Results on English-French Translation

SYSTEM	BLEU
Enc-Dec (Luong et al., 2014)	30.4
RNNsearch (Bahdanau et al., 2014)	28.5
RNNsearch-LV (Jean et al., 2015)	32.7
Deep-Att (Zhou et al., 2016)	35.9
DEEPLAU	35.1

Table 3: English-to-French task: BLEU scores of single neural models.

To evaluate at scale, we also show the results on an English-French task with $12M$ sentence pairs and $30K$ vocabulary in Table 3. Luong et al. (2014) achieves BLEU score of 30.4 with a six layers deep Encoder-Decoder model. The two attention models, RNNSearch and RNNsearch-LV achieve BLEU scores of 28.5 and 32.7 respectively. The previous best single NMT Deep-Att model with an 18 layers encoder and 7 layers decoder achieves BLEU score of 35.9. For DEEPLAU, we obtain the BLEU score of 35.1 with a 4 layers encoder and 4 layers decoder, which is on par with the SOTA system in terms of BLEU. Note that Zhou et al. (2016) utilize a much larger depth as well as external alignment model and extensive regularization to achieve their best results.

4.6 Analysis

Then we will study the main factors that influence our results on NIST Chinese-English translation task. We also compare our approach with two SOTA topologies which were used in building deep NMT systems.

- Residual Networks (ResNet) are among the pioneering works (Szegedy et al., 2016; He et al., 2016) that utilize extra identity connections to enhance information flow such that very deep neural networks can be effectively optimized. Share the similar idea, Wu et al. (2016) introduced to leverage residual connections to train deep RNNs.

- Fast Forward (F-F) connections were proposed to reduce the propagation path length which is the pioneer work to simplify the training of deep NMT model (Zhou et al., 2016). The work can be viewed as a parametric ResNet with short cut connections between adjacent layers. The procedure takes a linear sum between the input and the newly computed state.

LAU vs. GRU Table 4 shows the effect of the novel LAU. By comparing row 3 to row 7, we see that when L_{Enc} and L_{Dec} are set to 2,

SYSTEM	(L_{enc},L_{Dec})	width	AVE.
1 DEEPGRU	(2,1)	512	33.59
2 DEEPGRU	(2,2)	1024	34.68
3 DEEPGRU	(2,2)	512	34.91
4 DEEPGRU	(4,4)	512	34.08
5 4+ResNet	(4,4)	512	36.40
6 4+F-F	(4,4)	512	37.62
7 DEEPLAU	(2,2)	512	37.65
8 DEEPLAU	(4,4)	512	38.97
9 DEEPLAU	(8,6)	512	39.01
10 DEEPLAU	(8,6)	256	38.91

Table 4: BLEU scores of DEEPLAU and DEEPGRU with different model sizes.

the average BLEU scores achieved by DEEPGRU and DEEPLAU are 34.68 and 37.65, respectively. LAU can bring an improvement of 2.97 in terms of BLEU. After increasing the model depth to 4 (row 4 and row 6), the improvement is enlarged to 4.91. When DEEPGRU is trained with larger depth (say, 4), the training becomes more difficult and the performance falls behind its shallow partner. While for DEEPLAU, as can be see in row 9, with increasing the depth even to $L_{\text{Enc}} = 8$ and $L_{\text{Dec}} = 6$ we can still obtain growth by 0.04 BLEU score. Compared to previous shortcut connection methods (row 5 and row 6), The LAU still achieve meaningful improvements over F-F connections and Residual connections by $+1.35$ and $+2.57$ BLEU points respectively.

DEEPLAU introduces more parameters than DEEPGRU. In order to figure out the effect of DEEPLAU comparing models with the same parameter size, we increase the hidden size of DEEPGRU model. Row 3 shows that, after using a twice larger GRU layer, the BLEU score is 34.68, which is still worse than the corresponding DEEPLAU model with fewer parameters.

Depth vs. Width Next we will study the model size. In Table 4, starting from $L_{\text{Enc}} = 2$ and $L_{\text{Dec}} = 2$ and gradually increasing the model depth, we can achieve substantial improvements in terms of BLEU. With $L_{\text{Enc}} = 8$ and $L_{\text{Dec}} = 6$, our DEEPLAU model yields the best BLEU score. We tried to increase the model depth with the same hidden size but failed to see further improvements.

We then tried to increase the hidden size. By comparing row 2 and row 3, we find the improvements is relative small with a wider hidden size. It is also worth mentioning that a deep and thin network with fewer parameters can still achieve comparable results with its shallow partner. This suggests that depth plays a more important role in increasing the complexity of neural networks than width and our deliberately designed LAU benefit from the optimizing of such a deep model.

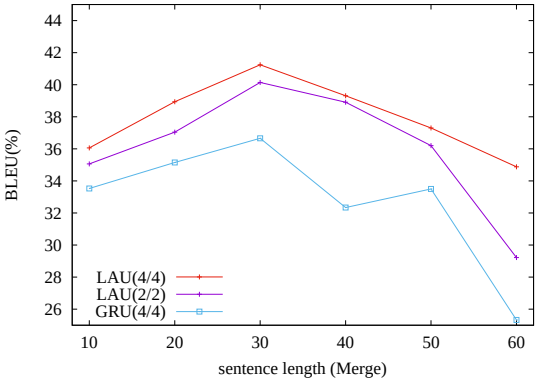


Figure 2: The BLEU scores of generated translations on the merged four test sets with respect to the lengths of source sentences.

About Length A more detailed comparison between DEEPLAU (4 layers encoder and 4 layers decoder), DEEPLAU(2 layer encoder and 2 layer decoder) and DEEPGRU (4 layers encoder and 4 layers decoder), suggest that with deep architectures are essential to the superior performance of our system. In particular, we test the BLEU scores on sentences longer than $\{10, 20, 30, 40, 50, 60\}$ on the merged test set. Clearly, in all curves, performance degrades with increased sentence length. However, DEEPLAU models yield consistently higher BLEU scores than the DEEPGRU model on longer sentences. These observations are consistent with our intuition that very deep RNN model is especially good at modeling the nested latent structures on relatively complicated sentences and LAU plays an important role on optimizing such a complex deep model.

5 Conclusion

We propose a Linear Associative Unit (LAU) which makes a fusion of both linear and non-linear transformation inside the recurrent unit. On this way, gradients decay much slower compared to the standard deep networks which enable us to build a deep neural network for machine translation. Our empirical study shows that it can significantly improve the performance of NMT.

6 acknowledge

We sincerely thank the anonymous reviewers for their thorough reviewing and valuable suggestions. Wang's work is partially supported by National Science Foundation for Deep Semantics Based Uighur to Chinese Machine Translation (ID 61662077). Qun Liu's work is partially supported by Science Foundation Ireland in the ADAPT Centre for Digital Content Technology (www.adaptcentre.ie) at Dublin City University funded under the SFI Research Centres Programme (Grant 13/RC/2106) co-funded under the European Regional Development Fund.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. 2014. N-gram counts and language models from the common crawl. In *LREC*. Citeseer, volume 2, page 4.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 263–270.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* .

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385* .

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 770–778.

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

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1–10. http://www.aclweb.org/anthology/P15-1001.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *arXiv preprint arXiv:1507.01526* .

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 48–54.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 609–616.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206* .

Fandong Meng, Zhengdong Lu, Zhaopeng Tu, Hang Li, and Qun Liu. 2015. Neural transformation machine: A new architecture for sequence-to-sequence learning. *CoRR* abs/1506.06442. http://arxiv.org/abs/1506.06442.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *ACL*. pages 192–199.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013a. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026* .

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013b. On the difficulty of training recurrent neural networks. *ICML (3)* 28:1310–1318.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433* .

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in neural information processing systems*. pages 2377–2385.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261* .

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *ArXiv eprints, January* .

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 521–528.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Biao Zhang, Deyi Xiong, and Jinsong Su. 2016. Variational neural machine translation. *arXiv preprint arXiv:1605.07869* .

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of EMNLP*.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199* .

Neural AMR: Sequence-to-Sequence Models for Parsing and Generation

Ioannis Konstas[†] **Srinivasan Iyer**[†] **Mark Yatskar**[†]
Yejin Choi[†] **Luke Zettlemoyer**[†‡]

[†]Paul G. Allen School of Computer Science & Engineering, Univ. of Washington, Seattle, WA
{ikonstas,sviyer,my89,yejin,lsz}@cs.washington.edu

[‡]Allen Institute for Artificial Intelligence, Seattle, WA
lukez@allenai.org

Abstract

Sequence-to-sequence models have shown strong performance across a broad range of applications. However, their application to parsing and generating text using Abstract Meaning Representation (AMR) has been limited, due to the relatively limited amount of labeled data and the non-sequential nature of the AMR graphs. We present a novel training procedure that can lift this limitation using millions of unlabeled sentences and careful preprocessing of the AMR graphs. For AMR parsing, our model achieves competitive results of 62.1 SMATCH, the current best score reported without significant use of external semantic resources. For AMR generation, our model establishes a new state-of-the-art performance of BLEU 33.8. We present extensive ablative and qualitative analysis including strong evidence that sequence-based AMR models are robust against ordering variations of graph-to-sequence conversions.

1 Introduction

Abstract Meaning Representation (AMR) is a semantic formalism to encode the meaning of natural language text. As shown in Figure 1, AMR represents the meaning using a directed graph while abstracting away the surface forms in text. AMR has been used as an intermediate meaning representation for several applications including machine translation (MT) (Jones et al., 2012), summarization (Liu et al., 2015), sentence compression (Takase et al., 2016), and event extraction (Huang et al., 2016). While AMR allows for rich semantic representation, annotating training data in AMR is expensive, which in turn limits the use

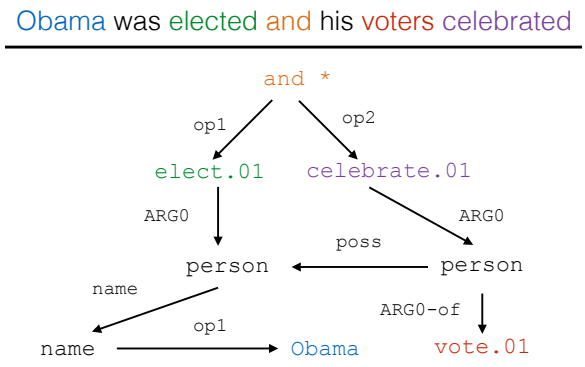


Figure 1: An example sentence and its corresponding Abstract Meaning Representation (AMR). AMR encodes semantic dependencies between entities mentioned in the sentence, such as "Obama" being the "arg0" of the verb "elected".

of neural network models (Misra and Artzi, 2016; Peng et al., 2017; Barzdins and Gosko, 2016).

In this work, we present the first successful sequence-to-sequence (seq2seq) models that achieve strong results for both text-to-AMR parsing and AMR-to-text generation. Seq2seq models have been broadly successful in many other applications (Wu et al., 2016; Bahdanau et al., 2015; Luong et al., 2015; Vinyals et al., 2015). However, their application to AMR has been limited, in part because effective *linearization* (encoding graphs as linear sequences) and data sparsity were thought to pose significant challenges. We show that these challenges can be easily overcome, by demonstrating that seq2seq models can be trained using *any* graph-isomorphic linearization and that unlabeled text can be used to significantly reduce sparsity.

Our approach is two-fold. First, we introduce a novel paired training procedure that enhances both the text-to-AMR parser and AMR-to-text generator. More concretely, first we use self-training to

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 146–157
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1014

bootstrap a high quality AMR parser from millions of unlabeled Gigaword sentences (Napoles et al., 2012) and then use the automatically parsed AMR graphs to pre-train an AMR generator. This paired training allows both the parser and generator to learn high quality representations of fluent English text from millions of weakly labeled examples, that are then fine-tuned using human annotated AMR data.

Second, we propose a preprocessing procedure for the AMR graphs, which includes anonymizing entities and dates, grouping entity categories, and encoding nesting information in concise ways, as illustrated in Figure 2(d). This preprocessing procedure helps overcoming the data sparsity while also substantially reducing the complexity of the AMR graphs. Under such a representation, we show that any depth first traversal of the AMR is an effective linearization, and it is even possible to use a different random order for each example.

Experiments on the LDC2015E86 AMR corpus (SemEval-2016 Task 8) demonstrate the effectiveness of the overall approach. For parsing, we are able to obtain competitive performance of 62.1 SMATCH without using any external annotated examples other than the output of a NER system, an improvement of over 10 points relative to neural models with a comparable setup. For generation, we substantially outperform previous best results, establishing a new state of the art of 33.8 BLEU. We also provide extensive ablative and qualitative analysis, quantifying the contributions that come from preprocessing and the paired training procedure.

2 Related Work

Alignment-based Parsing Flanigan et al. (2014) (JAMR) pipeline concept and relation identification with a graph-based algorithm. Zhou et al. (2016) extend JAMR by performing the concept and relation identification tasks jointly with an incremental model. Both systems rely on features based on a set of alignments produced using bi-lexical cues and hand-written rules. In contrast, our models train directly on parallel corpora, and make only minimal use of alignments to anonymize named entities.

Grammar-based Parsing Wang et al. (2016) (CAMR) perform a series of shift-reduce transformations on the output of an externally-trained dependency parser, similar to Damonte et al. (2017), Brandt et al. (2016), Puzikov et al. (2016), and Goodman et al. (2016). Artzi et al. (2015) use a grammar induction approach with Combinatory Categorical Grammar (CCG), which relies on pretrained CCGBank categories, like Bjerva et al. (2016). Pust et al. (2015) recast parsing as a string-to-tree Machine Translation problem, using unsupervised alignments (Pourdamghani et al., 2014), and employing several external semantic resources. Our neural approach is engineering lean, relying only on a large unannotated corpus of English and algorithms to find and canonicalize named entities.

Neural Parsing Recently there have been a few seq2seq systems for AMR parsing (Barzdins and Gosko, 2016; Peng et al., 2017). Similar to our approach, Peng et al. (2017) deal with sparsity by anonymizing named entities and typing low frequency words, resulting in a very compact vocabulary (2k tokens). However, we avoid reducing our vocabulary by introducing a large set of unlabeled sentences from an external corpus, therefore drastically lowering the out-of-vocabulary rate (see Section 6).

AMR Generation Flanigan et al. (2016) specify a number of tree-to-string transduction rules based on alignments and POS-based features that are used to drive a tree-based SMT system. Pourdamghani et al. (2016) also use an MT decoder; they learn a classifier that linearizes the input AMR graph in an order that follows the output sentence, effectively reducing the number of alignment crossings of the phrase-based decoder. Song et al. (2016) recast generation as a traveling salesman problem, after partitioning the graph into fragments and finding the best linearization order. Our models do not need to rely on a particular linearization of the input, attaining comparable performance even with a per example random traversal of the graph. Finally, all three systems intersect with a large language model trained on Gigaword. We show that our seq2seq model has the capacity to learn the same information as a language model, especially after pretraining on the external corpus.

Data Augmentation Our paired training procedure is largely inspired by Sennrich et al. (2016). They improve neural MT performance for low resource language pairs by using a back-translation MT system for a large monolingual corpus of the target language in order to create synthetic output,

and mixing it with the human translations. We instead pre-train on the external corpus first, and then fine-tune on the original dataset.

3 Methods

In this section, we first provide the formal definition of AMR parsing and generation (section 3.1). Then we describe the sequence-to-sequence models we use (section 3.2), graph-to-sequence conversion (section 3.3), and our paired training procedure (section 3.4).

3.1 Tasks

We assume access to a training dataset D where each example pairs a natural language sentence s with an AMR a. The AMR is a rooted directed acylical graph. It contains nodes whose names correspond to sense-identified verbs, nouns, or AMR specific concepts, for example `elect.01`, `Obama`, and `person` in Figure 1. One of these nodes is a distinguished root, for example, the node `and` in Figure 1. Furthermore, the graph contains labeled edges, which correspond to PropBank-style (Palmer et al., 2005) semantic roles for verbs or other relations introduced for AMR, for example, `arg0` or `op1` in Figure 1. The set of node and edge names in an AMR graph is drawn from a set of tokens C, and every word in a sentence is drawn from a vocabulary W.

We study the task of training an **AMR parser**, i.e., finding a set of parameters θ_P for model f, that predicts an AMR graph $\hat{a}$, given a sentence s:

$$\hat{a} = \underset{a}{\operatorname{argmax}} f\big(a|s;\theta_P\big) \quad (1)$$

We also consider the reverse task, training an **AMR generator** by finding a set of parameters θ_G, for a model f that predicts a sentence $\hat{s}$, given an AMR graph a:

$$\hat{s} = \underset{s}{\operatorname{argmax}} f\big(s|a;\theta_G\big) \quad (2)$$

In both cases, we use the same family of predictors f, sequence-to-sequence models that use global attention, but the models have independent parameters, θ_P and θ_G.

3.2 Sequence-to-sequence Model

For both tasks, we use a stacked-LSTM sequence-to-sequence neural architecture employed in neural machine translation (Bahdanau et al., 2015; Wu et al., 2016).[1] Our model uses a global attention decoder and unknown word replacement with small modifications (Luong et al., 2015).

The model uses a stacked bidirectional-LSTM encoder to encode an input sequence and a stacked LSTM to decode from the hidden states produced by the encoder. We make two modifications to the encoder: (1) we concatenate the forward and backward hidden states at every level of the stack instead of at the top of the stack, and (2) introduce dropout in the first layer of the encoder. The decoder predicts an attention vector over the encoder hidden states using previous decoder states. The attention is used to weigh the hidden states of the encoder and then predict a token in the output sequence. The weighted hidden states, the decoded token, and an attention signal from the previous time step (input feeding) are then fed together as input to the next decoder state. The decoder can optionally choose to output an unknown word symbol, in which case the predicted attention is used to copy a token directly from the input sequence into the output sequence.

3.3 Linearization

Our seq2seq models require that both the input and target be presented as a linear sequence of tokens. We define a linearization order for an AMR graph as any sequence of its nodes and edges. A linearization is defined as (1) a linearization order and (2) a rendering function that generates any number of tokens when applied to an element in the linearization order (see Section 4.2 for implementation details). Furthermore, for parsing, a valid AMR graph must be recoverable from the linearization.

3.4 Paired Training

Obtaining a corpus of jointly annotated pairs of sentences and AMR graphs is expensive and current datasets only extend to thousands of examples. Neural sequence-to-sequence models suffer from sparsity with so few training pairs. To reduce the effect of sparsity, we use an external unannotated corpus of sentences S_e, and a procedure which pairs the training of the parser and generator.

Our procedure is described in Algorithm 1, and first trains a parser on the dataset D of pairs of sentences and AMR graphs. Then it uses self-training

[1] We extended the Harvard NLP seq2seq framework from `http://nlp.seas.harvard.edu/code`.

Algorithm 1 Paired Training Procedure

Input: Training set of sentences and AMR graphs $(s, a) \in \mathcal{D}$, an unannotated external corpus of sentences S_e, a number of self training iterations, N, and an initial sample size k.
Output: Model parameters for AMR parser θ_P and AMR generator θ_G.

$\theta_P \leftarrow$ Train parser on D
▷ Self-train AMR parser.
$S_e^1 \leftarrow$ sample k sentences from S_e
for $i = 1$ to N **do**
    $A_e^i \leftarrow$ Parse S_e^i using parameters θ_P
    ▷ Pre-train AMR parser.
    $\theta_P \leftarrow$ Train parser on (A_e^i, S_e^i)
    ▷ Fine tune AMR parser.
    $\theta_P \leftarrow$ Train parser on D with initial parameters θ_P
    $S_e^{i+1} \leftarrow$ sample $k \cdot 10^i$ new sentences from S_e
end for
$S_e^N \leftarrow$ sample $k \cdot 10^N$ new sentences from S_e
▷ Pre-train AMR generator.
$A_e \leftarrow$ Parse S_e^N using parameters θ_P
$\theta_G \leftarrow$ Train generator on (A_e^N, S_e^N)
▷ Fine tune AMR generator.
$\theta_G \leftarrow$ Train generator on D using initial parameters θ_G
return θ_P, θ_G

to improve the initial parser. Every iteration of self-training has three phases: (1) parsing samples from a large, unlabeled corpus S_e, (2) creating a new set of parameters by training on S_e, and (3) fine-tuning those parameters on the original paired data. After each iteration, we increase the size of the sample from S_e by an order of magnitude. After we have the best parser from self-training, we use it to label AMRs for S_e and pre-train the generator. The final step of the procedure fine-tunes the generator on the original dataset D.

4 AMR Preprocessing

We use a series of preprocessing steps, including AMR linerization, anonymization, and other modifications we make to sentence-graph pairs. Our methods have two goals: (1) reduce the complexity of the linearized sequences to make learning easier while maintaining enough original information, and (2) address sparsity from certain open class vocabulary entries, such as named entities (NEs) and quantities. Figure 2(d) contains example inputs and outputs with all of our preprocessing techniques.

Graph Simplification In order to reduce the overall length of the linearized graph, we first remove variable names and the `instance-of` relation (/) before every concept. In case of re-entrant nodes we replace the variable mention with its co-referring concept. Even though this replacement incurs loss of information, often the surrounding context helps recover the correct realization, e.g., the possessive role `:poss` in the example of Figure 1 is strongly correlated with the surface form *his*. Following Pourdamghani et al. (2016) we also remove senses from all concepts for AMR generation only. Figure 2(a) contains an example output after this stage.

4.1 Anonymization of Named Entities

Open-class types including NEs, dates, and numbers account for 9.6% of tokens in the sentences of the training corpus, and 31.2% of vocabulary W. 83.4% of them occur fewer than 5 times in the dataset. In order to reduce sparsity and be able to account for new unseen entities, we perform extensive anonymization.

First, we anonymize sub-graphs headed by one of AMR's over 140 fine-grained entity types that contain a `:name` role. This captures structures referring to entities such as `person`, `country`, miscellaneous entities marked with `⋆-enitity`, and typed numerical values, `⋆-quantity`. We exclude `date` entities (see the next section). We then replace these sub-graphs with a token indicating fine-grained type and an index, i, indicating it is the ith occurrence of that type.[2] For example, in Figure 2 the sub-graph headed by `country` gets replaced with `country_0`.

On the training set, we use alignments obtained using the JAMR aligner (Flanigan et al., 2014) and the unsupervised aligner of Pourdamghani et al. (2014) in order to find mappings of anonymized subgraphs to spans of text and replace mapped text with the anonymized token that we inserted into the AMR graph. We record this mapping for use during testing of generation models. If a generation model predicts an anonymization token, we find the corresponding token in the AMR graph and replace the model's output with the most frequent mapping observed during training for the entity name. If the entity was never observed, we copy its name directly from the AMR graph.

Anonymizing Dates For dates in AMR graphs, we use separate anonymization tokens for year, month-number, month-name, day-number and day-name, indicating whether the date is mentioned by word or by number.[3] In AMR gener-

[2] In practice we only used three groups of ids: a different one for NEs, dates and constants/numbers.

[3] We also use three date format markers that appear in the text as: *YYYYMMDD*, *YYMMDD*, and *YYYY-MM-DD*.

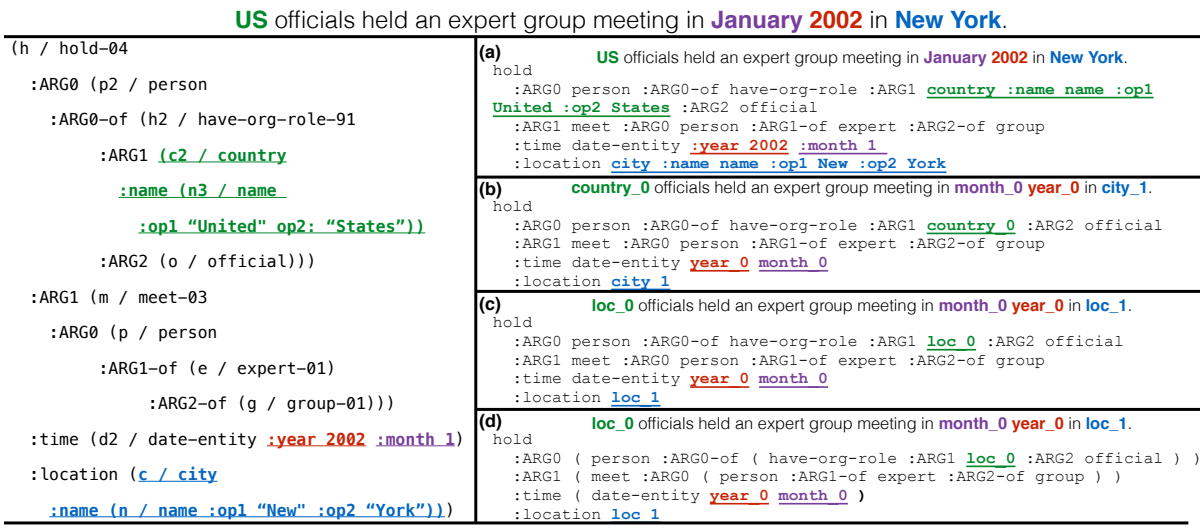


Figure 2: Preprocessing methods applied to sentence (top row) - AMR graph (left column) pairs. Sentence-graph pairs after (a) graph simplification, (b) named entity anonymization, (c) named entity clustering, and (d) insertion of scope markers.

ation, we render the corresponding format when predicted. Figure 2(b) contains an example of all preprocessing up to this stage.

Named Entity Clusters When performing AMR generation, each of the AMR fine-grained entity types is manually mapped to one of the four coarse entity types used in the Stanford NER system (Finkel et al., 2005): person, location, organization and misc. This reduces the sparsity associated with many rarely occurring entity types. Figure 2 (c) contains an example with named entity clusters.

NER for Parsing When parsing, we must normalize test sentences to match our anonymized training data. To produce fine-grained named entities, we run the Stanford NER system and first try to replace any identified span with a fine-grained category based on alignments observed during training. If this fails, we anonymize the sentence using the coarse categories predicted by the NER system, which are also categories in AMR. After parsing, we deterministically generate AMR for anonymizations using the corresponding text span.

4.2 Linearization

Linearization Order Our linearization order is defined by the order of nodes visited by depth first search, including backward traversing steps. For example, in Figure 2, starting at `meet` the order contains `meet`, `:ARG0`, `person`, `:ARG1-of`, `expert`, `:ARG2-of`, `group`, `:ARG2-of`, `:ARG1-of`, `:ARG0`.[4] The order traverses children in the sequence they are presented in the AMR. We consider alternative orderings of children in Section 7 but always follow the pattern demonstrated above.

Rendering Function Our rendering function marks scope, and generates tokens following the pre-order traversal of the graph: (1) if the element is a node, it emits the type of the node. (2) if the element is an edge, it emits the type of the edge and then recursively emits a bracketed string for the (concept) node immediately after it. In case the node has only one child we omit the scope markers (denoted with left "(", and right ")" parentheses), thus significantly reducing the number of generated tokens. Figure 2(d) contains an example showing all of the preprocessing techniques and scope markers that we use in our full model.

5 Experimental Setup

We conduct all experiments on the AMR corpus used in SemEval-2016 Task 8 (LDC2015E86), which contains 16,833/1,368/1,371 train/dev/test examples. For the paired training procedure of Algorithm 1, we use Gigaword as our external corpus and sample sentences that only contain words from the AMR corpus vocabulary W. We subsampled the original sentence to ensure there is no overlap with the AMR training or test sets. Table 2

[4] Sense, `instance-of` and variable information has been removed at the point of linearization.

	Dev			Test		
Model	Prec	Rec	F1	Prec	Rec	F1
SBMT (Pust et al., 2015)	-	-	69.0	-	-	67.1
CAMR (Wang et al., 2016)	72.3	61.4	66.6	70.4	63.1	66.5
CCG* (Artzi et al., 2015)	67.2	65.1	66.1	66.8	65.7	66.3
JAMR (Flanigan et al., 2014)	-	-	-	64.0	53.0	58.0
GIGA-20M	62.2	66.0	64.4	59.7	64.7	62.1
GIGA-2M	61.9	64.8	63.3	60.2	63.6	61.9
GIGA-200k	59.7	62.9	61.3	57.8	60.9	59.3
AMR-ONLY	54.9	60.0	57.4	53.1	58.1	55.5
SEQ2SEQ (Peng et al., 2017)	-	-	-	55.0	50.0	52.0
CHAR-LSTM (Barzdins and Gosko, 2016)	-	-	-	-	-	43.0

Table 1: SMATCH scores for AMR Parsing. *Reported numbers are on the newswire portion of a previous release of the corpus (LDC2014T12).

summarizes statistics about the original dataset and the extracted portions of Gigaword. We evaluate AMR parsing with SMATCH (Cai and Knight, 2013), and AMR generation using BLEU (Papineni et al., 2002)[5].

We validated word embedding sizes and RNN hidden representation sizes by maximizing AMR development set performance (Algorithm 1 – line 1). We searched over the set $\{128, 256, 500, 1024\}$ for the best combinations of sizes and set both to 500. Models were trained by optimizing cross-entropy loss with stochastic gradient descent, using a batch size of 100 and dropout rate of 0.5. Across all models when performance does not improve on the AMR dev set, we decay the learning rate by 0.8.

For the initial parser trained on the AMR corpus, (Algorithm 1 – line 1), we use a single stack version of our model, set initial learning rate to 0.5 and train for 60 epochs, taking the best performing model on the development set. All subsequent models benefited from increased depth and we used 2-layer stacked versions, maintaining the same embedding sizes. We set the initial Gigaword sample size to $k = 200,000$ and executed a maximum of 3 iterations of self-training. For pre-training the parser and generator, (Algorithm 1 – lines 4 and 9), we used an initial learning rate of 1.0, and ran for 20 epochs. We attempt to fine-tune the parser and generator, respectively, after every epoch of pre-training, setting the initial learning rate to 0.1. We select the best performing model on the development set among all of these fine-tuning attempts. During prediction we perform decoding using beam search and set the beam size to 5 both for parsing and generation.

[5]We use the multi-BLEU script from the MOSES decoder suite (Koehn et al., 2007).

Corpus	Examples	OOV@1	OOV@5
AMR	16833	44.7	74.9
GIGA-200k	200k	17.5	35.3
GIGA-2M	2M	11.2	19.1
GIGA-20M	20M	8.0	12.7

Table 2: LDC2015E86 AMR training set, GIGA-200k, GIGA-2M and GIGA-20M statistics; OOV@1 and OOV@5 are the out-of-vocabulary rates on the NL side with thresholds of 1 and 5, respectively. Vocabulary sizes are 13027 tokens for the AMR side, and 17319 tokens for the NL side.

Model	Dev	Test
GIGA-20M	33.1	33.8
GIGA-2M	31.8	32.3
GIGA-200k	27.2	27.4
AMR-ONLY	21.7	22.0
PBMT* (Pourdamghani et al., 2016)	27.2	26.9
TSP (Song et al., 2016)	21.1	22.4
TREETOSTR (Flanigan et al., 2016)	23.0	23.0

Table 3: BLEU results for AMR Generation. *Model has been trained on a previous release of the corpus (LDC2014T12).

6 Results

Parsing Results Table 1 summarizes our development results for different rounds of self-training and test results for our final system, self-trained on 200k, 2M and 20M unlabeled Gigaword sentences. Through every round of self-training, our

parser improves. Our final parser outperforms comparable seq2seq and character LSTM models by over 10 points. While much of this improvement comes from self-training, our model without Gigaword data outperforms these approaches by 3.5 points on F1. We attribute this increase in performance to different handling of preprocessing and more careful hyper-parameter tuning. All other models that we compare against use semantic resources, such as WordNet, dependency parsers or CCG parsers (models marked with * were trained with less data, but only evaluate on newswire text; the rest evaluate on the full test set, containing text from blogs). Our full models outperform JAMR, a graph-based model but still lags behind other parser-dependent systems (CAMR[6]), and resource heavy approaches (SBMT).

Generation Results Table 3 summarizes our AMR generation results on the development and test set. We outperform all previous state-of-the-art systems by the first round of self-training and further improve with the next rounds. Our final model trained on GIGA-20M outperforms TSP and TREETOSTR trained on LDC2015E86, by over 9 BLEU points.[7] Overall, our model incorporates less data than previous approaches as all reported methods train language models on the whole Gigaword corpus. We leave scaling our models to all of Gigaword for future work.

Sparsity Reduction Even after anonymization of open class vocabulary entries, we still encounter a great deal of sparsity in vocabulary given the small size of the AMR corpus, as shown in Table 2. By incorporating sentences from Gigaword we are able to reduce vocabulary sparsity dramatically, as we increase the size of sampled sentences: the out-of-vocabulary rate with a threshold of 5 reduces almost 5 times for GIGA-20M.

Preprocessing Ablation Study We consider the contribution of each main component of our preprocessing stages while keeping our linearization order identical. Figure 2 contains examples for each setting of the ablations we evaluate on. First we evaluate using linearized graphs without parentheses for indicating scope, Figure 2(c), then without named entity clusters, Figure 2(b), and additionally without any anonymization, Figure 2(a).

[6]Since we are currently not using any Wikipedia resources for the prediction of named entities, we compare against the no-wikification version of the CAMR system.

[7]We also trained our generator on GIGA-2M and fine-tuned on LDC2014T12 in order to have a direct comparison with PBMT, and achieved a BLEU score of 29.7, i.e., 2.8 points of improvement.

Model	BLEU
FULL	21.8
FULL - SCOPE	19.7
FULL - SCOPE - NE	19.5
FULL - SCOPE - NE - ANON	18.7

Table 4: BLEU scores for AMR generation ablations on preprocessing (DEV set).

Model	Prec	Rec	F1
FULL	54.9	60.0	57.4
FULL - ANON	22.7	54.2	32.0

Table 5: SMATCH scores for AMR parsing ablations on preprocessing (DEV set).

Tables 4 summarizes our evaluation on the AMR generation. Each components is required, and scope markers and anonymization contribute the most to overall performance. We suspect without scope markers our seq2seq models are not as effective at capturing long range semantic relationships between elements of the AMR graph. We also evaluated the contribution of anonymization to AMR parsing (Table 5). Following previous work, we find that seq2seq-based AMR parsing is largely ineffective without anonymization (Peng et al., 2017).

7 Linearization Evaluation

In this section we evaluate three strategies for converting AMR graphs into sequences in the context of AMR generation and show that our models are largely agnostic to linearization orders. Our results argue, unlike SMT-based AMR generation methods (Pourdamghani et al., 2016), that seq2seq models can learn to ignore artifacts of the conversion of graphs to linear sequences.

7.1 Linearization Orders

All linearizations we consider use the pattern described in Section 4.2, but differ on the order in which children are visited. Each linearization generates anonymized, scope-marked output (see Section 4), of the form shown in Figure 2(d).

Human The proposal traverses children in the order presented by human authored AMR annotations exactly as shown in Figure 2(d).

Linearization Order	BLEU
HUMAN	21.7
GLOBAL-RANDOM	20.8
RANDOM	20.3

Table 6: BLEU scores for AMR generation for different linearization orders (DEV set).

Global-Random We construct a random global ordering of all edge types appearing in AMR graphs and re-use it for every example in the dataset. We traverse children based on the position in the global ordering of the edge leading to a child.

Random For each example in the dataset we traverse children following a different random order of edge types.

7.2 Results

We present AMR generation results for the three proposed linearization orders in Table 6. Random linearization order performs somewhat worse than traversing the graph according to Human linearization order. Surprisingly, a per example random linearization order performs nearly identically to a global random order, arguing seq2seq models can learn to ignore artifacts of the conversion of graphs to linear sequences.

Human-authored AMR leaks information The small difference between random and global-random linearizations argues that our models are largely agnostic to variation in linearization order. On the other hand, the model that follows the human order performs better, which leads us to suspect it carries extra information not apparent in the graphical structure of the AMR.

To further investigate, we compared the relative ordering of edge pairs under the same parent to the relative position of children nodes derived from those edges in a sentence, as reported by JAMR alignments. We found that the majority of pairs of AMR edges (57.6%) always occurred in the same relative order, therefore revealing no extra generation order information.[8] Of the examples corresponding to edge pairs that showed variation, 70.3% appeared in an order consistent with the order they were realized in the sentence. The relative ordering of some pairs of AMR edges was particularly indicative of generation order. For example, the relative ordering of edges with types `location` and `time`, was 17% more indicative of the generation order than the majority of generated locations before `time`.[9]

To compare to previous work we still report results using human orderings. However, we note that any practical application requiring a system to generate an AMR representation with the intention to realize it later on, e.g., a dialog agent, will need to be trained either using consistent, or random-derived linearization orders. Arguably, our models are agnostic to this choice.

Error Type	%
Coverage	29
Disfluency	23
Anonymization	14
Sparsity	13
Attachment	12
Other	10

Table 7: Error analysis for AMR generation on a sample of 50 examples from the development set.

8 Qualitative Results

Figure 3 shows example outputs of our full system. The generated text for the first graph is nearly perfect with only a small grammatical error due to anonymization. The second example is more challenging, with a deep right-branching structure, and a coordination of the verbs `stabilize` and `push` in the subordinate clause headed by `state`. The model omits some information from the graph, namely the concepts `terrorist` and `virus`. In the third example there are greater parts of the graph that are missing, such as the whole sub-graph headed by `expert`. Also the model makes wrong attachment decisions in the last two sub-graphs (it is the `evidence` that is *unimpeachable* and *irrefutable*, and not the `equipment`), mostly due to insufficient annotation (`thing`) thus making their generation harder.

Finally, Table 7 summarizes the proportions of error types we identified on 50 randomly selected examples from the development set. We found that the generator mostly suffers from coverage issues,

[8] This is consistent with constraints encoded in the annotation tool used to collect AMR. For example, `:ARG0` edges are always ordered before `:ARG1` edges.

[9] Consider the sentences *"She went to school in New York two years ago"*, and *"Two years ago, she went to school in New York"*, where *"two year ago"* is the time modifying constituent for the verb *went* and *"New York"* is the location modifying constituent of *went*.

an inability to mention all tokens in the input, followed by fluency mistakes, as illustrated above. Attachment errors are less frequent, which supports our claim that the model is robust to graph linearization, and can successfully encode long range dependency information between concepts.

9 Conclusions

We applied sequence-to-sequence models to the tasks of AMR parsing and AMR generation, by carefully preprocessing the graph representation and scaling our models via pretraining on millions of unlabeled sentences sourced from Gigaword corpus. Crucially, we avoid relying on resources such as knowledge bases and externally trained parsers. We achieve competitive results for the parsing task (SMATCH 62.1) and state-of-the-art performance for generation (BLEU 33.8).

For future work, we would like to extend our work to different meaning representations such as the Minimal Recursion Semantics (MRS; Copestake et al. (2005)). This formalism tackles certain linguistic phenomena differently from AMR (e.g., negation, and co-reference), contains explicit annotation on concepts for number, tense and case, and finally handles multiple languages[10] (Bender, 2014). Taking a step further, we would like to apply our models on Semantics-Based Machine Translation using MRS as an intermediate representation between pairs of languages, and investigate the added benefit compared to directly translating the surface strings, especially in the case of distant language pairs such as English and Japanese (Siegel, 2000).

Acknowledgments

The research was supported in part by DARPA under the DEFT program through AFRL (FA8750-13-2-0019) and the CwC program through ARO (W911NF-15-1-0543), the ARO (W911NF-16-1-0121), the NSF (IIS-1252835, IIS-1562364, IIS-1524371), an Allen Distinguished Investigator Award, and gifts by Google and Facebook. The authors thank Rik Koncel-Kedziorski and the UW NLP group for helpful discussions, and the anonymous reviewers for their thorough and helpful comments.

[10] A list of actively maintained languages can be found here: http://moin.delph-in.net/GrammarCatalogue

```
limit
 :arg0 ( treaty :arg0-of ( control :arg1 arms ) )
 :arg1 ( number
 :arg1 ( weapon :mod conventional
  :arg1-of ( deploy
   :arg2 ( relative-pos :op1 loc_0 :dir west )
  :arg1-of possible ) ) )
```

REF: the arms control treaty limits the number of conventional weapons that can be deployed west of the Ural Mountains .

SYS: the arms control treaty limits the number of conventional weapons that can be deployed west of Ural Mountains .

COMMENT: disfluency

```
state
 :arg0 report
 :arg1 ( obligate :arg1 ( government-organization
   :arg0-of ( govern :arg1 loc_0 ) )
   :arg2 ( help :arg1 ( and
    :op1 ( stabilize :arg1 ( state :mod weak ) )
    :op2 ( push :arg1 ( regulate
      :mod international :arg0-of ( stop
       :arg1 terrorist
       :arg2 ( use
        :arg1 ( information
         :arg2-of ( available :arg3-of free ))
        :arg2 ( and
         :op1 ( create :arg1 ( form
          :domain ( warfare
           :mod biology :example ( version
             :arg1-of modify :poss other_1 ) )
           :mod new ) )
         :op2 ( unleash :arg1 form  )
      ) ) ) ) ) ) ) )
```

REF: the report stated British government must help to stabilize weak states and push for international regulations that would stop terrorists using freely available information to create and unleash new forms of biological warfare such as a modified version of the influenza virus .

SYS: the report stated that the Britain government must help stabilize the weak states and push international regulations to stop the use of freely available information to create a form of new biological warfare such as the modified version of the influenza .

COMMENT: coverage , disfluency, attachment

```
state
 :arg0 ( person
  :arg0-of ( have-org-role
   :arg1 ( committee :mod technical )
   :arg3 ( expert
    :arg1 person
    :arg2 missile
    :mod loc_0 ) ) )
 :arg1 ( evidence
  :arg0 equipment
  :arg1 ( plan :arg1 ( transfer :arg1 ( contrast
   :arg1 ( missile :mod ( just :polarity - ) )
    :arg2 ( capable
     :arg1 thing
     :arg2 ( make :arg1 missile ) ) ) ) )
  :mod ( impeach :polarity - :arg1 thing )
  :mod ( refute :polarity - :arg1 thing ) )
```

REF: a technical committee of Indian missile experts stated that the equipment was unimpeachable and irrefutable evidence of a plan to transfer not just missiles but missile-making capability.

SYS: a technical committee expert on the technical committee stated that the equipment is not impeach , but it is not refutes .

COMMENT: coverage , disfluency, attachment

Figure 3: Linearized AMR after preprocessing, reference sentence, and output of the generator. We mark with colors common error types: disfluency, coverage (missing information from the input graph), and attachment (implying a semantic relation from the AMR between incorrect entities).

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1699–1710. http://aclweb.org/anthology/D15-1198.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*. CBLS, San Diego, California. http://arxiv.org/abs/1409.0473.

Guntis Barzdins and Didzis Gosko. 2016. RIGA at SemEval-2016 Task 8: Impact of Smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1143–1147. http://www.aclweb.org/anthology/S16-1176.

Emily M. Bender. 2014. Language CoLLAGE: Grammatical description with the LinGO grammar matrix. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*. Reykjavik, Iceland, pages 2447–2451.

Johannes Bjerva, Johan Bos, and Hessel Haagsma. 2016. The Meaning Factory at SemEval-2016 Task 8: Producing AMRs with Boxer. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1179–1184. http://www.aclweb.org/anthology/S16-1182.

Lauritz Brandt, David Grimm, Mengfei Zhou, and Yannick Versley. 2016. ICL-HD at SemEval-2016 Task 8: Meaning representation parsing - augmenting AMR parsing with a preposition semantic role labeling neural network. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1160–1166. http://www.aclweb.org/anthology/S16-1179.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sofia, Bulgaria, pages 748–752. http://www.aclweb.org/anthology/P13-2131.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal Recursion Semantics: An introduction. *Research on Language and Computation* 3(2):281–332. https://doi.org/10.1007/s11168-006-6327-9.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain, pages 536–546. http://www.aclweb.org/anthology/E17-1051.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 363–370. https://doi.org/10.3115/1219840.1219885.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, San Diego, California, pages 731–739. http://www.aclweb.org/anthology/N16-1087.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Baltimore, Maryland, pages 1426–1436. http://www.aclweb.org/anthology/P14-1134.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. UCL+Sheffield at SemEval-2016 Task 8: Imitation learning for AMR parsing with an alpha-bound. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1167–1172. http://www.aclweb.org/anthology/S16-1180.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 258–268. http://www.aclweb.org/anthology/P16-1025.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *Proceedings of the 2012 International Conference on Computational Linguistics*. Bombay, India, pages 1359–1376. http://www.aclweb.org/anthology/C12-1083.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine

translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 177–180. http://dl.acm.org/citation.cfm?id=1557769.1557821.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Denver, Colorado, pages 1077–1086. http://www.aclweb.org/anthology/N15-1114.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Dipendra Kumar Misra and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1775–1786. https://aclweb.org/anthology/D16-1183.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, Montréal, Canada, pages 95–100. http://www.aclweb.org/anthology/W12-3018.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1):71–106. http://www.cs.rochester.edu/ gildea/palmer-propbank-cl.pdf.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural AMR parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain, pages 366–375. http://www.aclweb.org/anthology/E17-1035.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Doha, Qatar, pages 425–429. http://www.aclweb.org/anthology/D14-1048.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating English from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference*. Association for Computational Linguistics, Edinburgh, UK, pages 21–25. http://anthology.aclweb.org/W16-6603.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1143–1154. https://aclweb.org/anthology/D/D15/D15-1136.

Yevgeniy Puzikov, Daisuke Kawahara, and Sadao Kurohashi. 2016. M2L at SemEval-2016 Task 8: AMR parsing with neural networks. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1154–1159. http://www.aclweb.org/anthology/S16-1178.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 86–96. http://www.aclweb.org/anthology/P16-1009.

Melanie Siegel. 2000. *HPSG Analysis of Japanese*, Springer Berlin Heidelberg, pages 264–279.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. AMR-to-text generation as a traveling salesman problem. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2084–2089. https://aclweb.org/anthology/D16-1224.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1054–1059. https://aclweb.org/anthology/D16-1112.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the*

28th International Conference on Neural Information Processing Systems, MIT Press, pages 2773–2781. http://papers.nips.cc/paper/5635-grammar-as-a-foreign-language.pdf.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at SemEval-2016 Task 8: An extended transition-based AMR parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, pages 1173–1178. http://www.aclweb.org/anthology/S16-1181.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 680–689. https://aclweb.org/anthology/D16-1065.

Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems

Wang Ling[♠] **Dani Yogatama**[♠] **Chris Dyer**[♠] **Phil Blunsom**[♠♢]
♠DeepMind ♢University of Oxford
{lingwang,dyogatama,cdyer,pblunsom}@google.com

Abstract

Solving algebraic word problems requires executing a series of arithmetic operations—a program—to obtain a final answer. However, since programs can be arbitrarily complicated, inducing them directly from question-answer pairs is a formidable challenge. To make this task more feasible, we solve these problems by generating *answer rationales*, sequences of natural language and human-readable mathematical expressions that derive the final answer through a series of small steps. Although rationales do not explicitly specify programs, they provide a scaffolding for their structure via intermediate milestones. To evaluate our approach, we have created a new 100,000-sample dataset of questions, answers and rationales. Experimental results show that indirect supervision of program learning via answer rationales is a promising strategy for inducing arithmetic programs.

1 Introduction

Behaving intelligently often requires mathematical reasoning. Shopkeepers calculate change, tax, and sale prices; agriculturists calculate the proper amounts of fertilizers, pesticides, and water for their crops; and managers analyze productivity. Even determining whether you have enough money to pay for a list of items requires applying addition, multiplication, and comparison. Solving these tasks is challenging as it involves recognizing how goals, entities, and quantities in the real-world map onto a mathematical formalization, computing the solution, and mapping the solution back onto the world. As a proxy for the richness of the real world, a series of papers have used natural language specifications of algebraic word problems, and solved these by either learning to fill in templates that can be solved with equation solvers (Hosseini et al., 2014; Kushman et al., 2014) or inferring and modeling operation sequences (programs) that lead to the final answer (Roy and Roth, 2015).

In this paper, we learn to solve algebraic word problems by inducing and modeling programs that generate not only the answer, but an **answer rationale**, a natural language explanation interspersed with algebraic expressions justifying the overall solution. Such rationales are what examiners require from students in order to demonstrate understanding of the problem solution; they play the very same role in our task. Not only do natural language rationales enhance model interpretability, but they provide a coarse guide to the structure of the arithmetic programs that must be executed. In fact the learner we propose (which relies on a heuristic search; §4) fails to solve this task without modeling the rationales—the search space is too unconstrained.

This work is thus related to models that can explain or rationalize their decisions (Hendricks et al., 2016; Harrison et al., 2017). However, the use of rationales in this work is quite different from the role they play in most prior work, where interpretation models are trained to generate plausible sounding (but not necessarily accurate) post-hoc descriptions of the decision making process they used. In this work, the rationale is generated as a latent variable that gives rise to the answer—it is thus a more faithful representation of the steps used in computing the answer.

This paper makes three contributions. First, we have created a new dataset with more than 100,000 algebraic word problems that includes both answers and natural language answer rationales (§2). Figure 1 illustrates three representative instances

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 158–167
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1015

Problem 1:
Question: Two trains running in opposite directions cross a man standing on the platform in 27 seconds and 17 seconds respectively and they cross each other in 23 seconds. The ratio of their speeds is:
Options: A) 3/7 B) 3/2 C) 3/88 D) 3/8 E) 2/2
Rationale: Let the speeds of the two trains be x m/sec and y m/sec respectively. Then, length of the first train = 27x meters, and length of the second train = 17 y meters. (27x + 17y) / (x + y) = 23 → 27x + 17y = 23x + 23y → 4x = 6y → x/y = 3/2.
Correct Option: B

Problem 2:
Question: From a pack of 52 cards, two cards are drawn together at random. What is the probability of both the cards being kings?
Options: A) 2/1223 B) 1/122 C) 1/221 D) 3/1253 E) 2/153
Rationale: Let s be the sample space.
Then n(s) = 52C2 = 1326
E = event of getting 2 kings out of 4
n(E) = 4C2 = 6
P(E) = 6/1326 = 1/221
Answer is C
Correct Option: C

Problem 3:
Question: For which of the following does $p(a)-p(b) = p(ab)$ for all values of a and b?
Options:A) $p(x) = x^2$, B) $p(x) = x/2$, C) $p(x) = x + 5$, D) $p(x) = 2x1$, E) $p(x) = |x|$
Rationale: To solve this easiest way is just put the value and see that if it equals or not.
with option A. $p(a) = a^2$ and $p(b) = b^2$
so L.H.S = $a^2 - b^2$
and R.H.S = $(a - b)^2 \rightarrow a^2 + b^2 - 2ab$.
so L.H.S not equal to R.H.S
with option B. $p(a) = a/2$ and $p(b) = b/2$
L.H.S = $a/2 - b/2 \rightarrow 1/2(a - b)$
R.H.S = $(a - b)/2$
so L.H.S = R.H.S which is the correct answer.
answer:B
Correct Option: B

Figure 1: Examples of solved math problems.

from the dataset. Second, we propose a sequence to sequence model that generates a sequence of instructions that, when executed, generates the rationale; only after this is the answer chosen (§3). Since the target program is not given in the training data (most obviously, its specific form will depend on the operations that are supported by the program interpreter); the third contribution is thus a technique for inferring programs that generate a rationale and, ultimately, the answer. Even constrained by a text rationale, the search space of possible programs is quite large, and we employ a heuristic search to find plausible next steps to guide the search for programs (§4). Empirically, we are able to show that state-of-the-art sequence to sequence models are unable to perform above chance on this task, but that our model doubles the accuracy of the baseline (§6).

2 Dataset

We built a dataset with 100,000 problems with the annotations shown in Figure 1. Each question is decomposed in four parts, two inputs and two outputs: the description of the problem, which we will denote as the **question**, and the possible (multiple choice) answer options, denoted as **options**. Our goal is to generate the description of the rationale used to reach the correct answer, denoted as **rationale** and the **correct option** label. Problem 1 illustrates an example of an algebra problem, which must be translated into an expression (i.e., $(27x + 17y)/(x + y) = 23$) and then the desired quantity (x/y) solved for. Problem 2 is an example that could be solved by multi-step arithmetic operations proposed in (Roy and Roth, 2015). Finally, Problem 3 describes a problem that is solved by testing each of the options, which has not been addressed in the past.

2.1 Construction

We first collect a set of 34,202 seed problems that consist of multiple option math questions covering a broad range of topics and difficulty levels. Examples of exams with such problems include the GMAT (Graduate Management Admission Test) and GRE (General Test). Many websites contain example math questions in such exams, where the answer is supported by a rationale.

Next, we turned to crowdsourcing to generate new questions. We create a task where users are presented with a set of 5 questions from our seed dataset. Then, we ask the Turker to choose one of the questions and write a similar question. We also force the answers and rationale to differ from the original question in order to avoid paraphrases of the original question. Once again, we manually check a subset of the jobs for each Turker for quality control. The type of questions generated using this method vary. Some turkers propose small changes in the values of the questions (e.g., changing the equality $p(a)p(b) = p(ab)$ in Problem 3 to a different equality is a valid question, as long as the rationale and options are rewritten to reflect the change). We designate these as replica problems as the natural language used in the question and rationales tend to be only minimally unaltered. Others propose new problems in the same topic where the generated questions tend to differ more radically from existing ones. Some Turkers also copy math problems available on the web, and we

		Question	Rationale
Training Examples		100,949	
Dev Examples		250	
Test Examples		250	
Numeric	Average Length	9.6	16.6
	Vocab Size	21,009	14,745
Non-Numeric	Average Length	67.8	89.1
	Vocab Size	17,849	25,034
All	Average Length	77.4	105.7
	Vocab Size	38,858	39,779

Table 1: Descriptive statistics of our dataset.

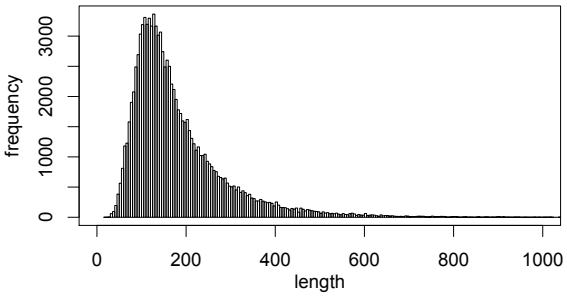


Figure 2: Distribution of examples per length.

define in the instructions that this is not allowed, as it will generate multiple copies of the same problem in the dataset if two or more Turkers copy from the same resource. These Turkers can be detected by checking the nearest neighbours within the collected datasets as problems obtained from online resources are frequently submitted by more than one Turker. Using this method, we obtained 70,318 additional questions.

2.2 Statistics

Descriptive statistics of the dataset is shown in Figure 1. In total, we collected 104,519 problems (34,202 seed problems and 70,318 crowdsourced problems). We removed 500 problems as heldout set (250 for development and 250 for testing). As replicas of the heldout problems may be present in the training set, these were removed manually by listing for each heldout instance the closest problems in the training set in terms of character-based Levenstein distance. After filtering, 100,949 problems remained in the training set.

We also show the average number of tokens (total number of tokens in the question, options and rationale) and the vocabulary size of the questions and rationales. Finally, we provide the same statistics exclusively for tokens that are numeric values and tokens that are not.

Figure 2 shows the distribution of examples based on the total number of tokens. We can see that most examples consist of 30 to 500 tokens, but there are also extremely long examples with more than 1000 tokens in our dataset.

3 Model

Generating rationales for math problems is challenging as it requires models that learn to perform math operations at a finer granularity as each step within the solution must be explained. For instance, in Problem 1, the equation $(27x + 17y)/(x + y) = 23$ must be solved to obtain the answer. In previous work (Kushman et al., 2014), this could be done by feeding the equation into an expression solver to obtain $x/y = 3/2$. However, this would skip the intermediate steps $27x + 17y = 23x + 23y$ and $4x = 6y$, which must also be generated in our problem. We propose a model that jointly learns to generate the text in the rationale, and to perform the math operations required to solve the problem. This is done by generating a program, containing both instructions that generate output and instructions that simply generate intermediate values used by following instructions.

3.1 Problem Definition

In traditional sequence to sequence models (Sutskever et al., 2014; Bahdanau et al., 2014), the goal is to predict the output sequence $\boldsymbol{y} = y_1, \ldots, y_{|\boldsymbol{y}|}$ from the input sequence $\boldsymbol{x} = x_1, \ldots, x_{|\boldsymbol{x}|}$, with lengths $|\boldsymbol{y}|$ and $|\boldsymbol{x}|$.

In our particular problem, we are given the problem and the set of options, and wish to predict the rationale and the correct option. We set $\boldsymbol{x}$ as the sequence of words in the problem, concatenated with words in each of the options separated by a special tag. Note that knowledge about the possible options is required as some problems are solved by the process of elimination or by testing each of the options (e.g. Problem 3). We wish to generate $\boldsymbol{y}$, which is the sequence of words in the rationale. We also append the correct option as the last word in $\boldsymbol{y}$, which is interpreted as the chosen option. For example, $\boldsymbol{y}$ in Problem 1 is "Let the ... = 3/2 . ⟨EOR⟩ B ⟨EOS⟩", whereas in Problem 2 it is "Let s be ... Answer is C ⟨EOR⟩ C ⟨EOS⟩", where "⟨EOS⟩" is the end of sentence symbol and "⟨EOR⟩" is the end of rationale symbol.

i	$\boldsymbol{x}$	$\boldsymbol{z}$	$\boldsymbol{v}$	$\boldsymbol{r}$
1	From	Id("Let")	*Let*	y_1
2	a	Id("s")	*s*	y_2
3	pack	Id("be")	*be*	y_3
4	of	Id("the")	*the*	y_4
5	52	Id("sample")	*sample*	y_5
6	cards	Id("space")	*space*	y_6
7	,	Id(".")	*.*	y_7
8	two	Id("\n")	*\n*	y_8
9	cards	Id("Then")	*Then*	y_9
10	are	Id("n")	*n*	y_{10}
11	drawn	Id("(")	*(*	y_{11}
12	together	Id("s")	*s*	y_{12}
13	at	Id(")")	*)*	y_{13}
14	random	Id("=")	=	y_{14}
15	.	Str_to_Float(x_5)	**52**	$\underline{m_1}$
16	What	Float_to_Str(m_1)	*52*	$\underline{y_{15}}$
17	is	Id("C")	*C*	y_{16}
18	the	Id("2")	*2*	y_{17}
19	probability	Id("=")	=	y_{18}
20	of	Str_to_Float(y_{17})	**2**	$\underline{m_2}$
21	both	Choose(m_1,m_2)	**1326**	$\underline{m_3}$
22	cards	Float_to_Str(m_3)	*1326*	y_{19}
23	being	Id("E")	*E*	y_{20}
24	kings	Id("=")	=	y_{21}
25	?	Id("event")	*event*	y_{22}
26	<O>	Id("of")	*of*	y_{23}
27	A)	Id("getting")	*getting*	y_{24}
28	2/1223	Id("2")	*2*	y_{25}
29	<O>	Id("kings")	*kings*	y_{26}
30	B)	Id("out")	*out*	y_{27}
31	1/122	Id("of")	*of*	y_{28}
...	...	...	...	...
$\lvert\boldsymbol{z}\rvert$		Id("⟨EOS⟩")	*⟨EOS⟩*	$y_{\lvert\boldsymbol{y}\rvert}$

Table 2: Example of a program $\boldsymbol{z}$ that would generate the output $\boldsymbol{y}$. In $\boldsymbol{v}$, *italics* indicates string types; ***bold*** indicates float types. Refer to §3.3 for description of variable names.

3.2 Generating Programs to Generate Rationales

We wish to generate a latent sequence of **program instructions**, $\boldsymbol{z} = z_1, \ldots, z_{|\boldsymbol{z}|}$, with length $|\boldsymbol{z}|$, that will generate $\boldsymbol{y}$ when executed.

We express $\boldsymbol{z}$ as a program that can access $\boldsymbol{x}$, $\boldsymbol{y}$, and the memory buffer $\boldsymbol{m}$. Upon finishing execution we expect that the sequence of output tokens to be placed in the output vector $\boldsymbol{y}$.

Table 2 illustrates an example of a sequence of instructions that would generate an excerpt from Problem 2, where columns $\boldsymbol{x}$, $\boldsymbol{z}$, $\boldsymbol{v}$, and $\boldsymbol{r}$ denote the input sequence, the instruction sequence (program), the values of executing the instruction, and where each value v_i is written (i.e., either to the output or to the memory). In this example, instructions from indexes 1 to 14 simply fill each position with the observed output $y_1, \ldots, y_{14}$ with a string, where the `Id` operation simply returns its parameter without applying any operation. As such, running this operation is analogous to generating a word by sampling from a softmax over a vocabulary. However, instruction z_{15} reads the input word x_5, 52, and applies the operation `Str_to_Float`, which converts the word 52 into a floating point number, and the same is done for instruction z_{20}, which reads a previously generated output word y_{17}. Unlike, instructions $z_1, \ldots, z_{14}$, these operations write to the external memory $\boldsymbol{m}$, which stores intermediate values. A more sophisticated instruction—which shows some of the power of our model—is $z_{21} = \texttt{Choose}(m_1, m_2) \rightarrow m_3$ which evaluates $\binom{m_1}{m_2}$ and stores the result in m_3. This process repeats until the model generates the end-of-sentence symbol. The last token of the program as said previously must generate the correct option value, from "A" to "E".

By training a model to generate instructions that can manipulate existing tokens, the model benefits from the additional expressiveness needed to solve math problems within the generation process. In total we define 22 different operations, 13 of which are frequently used operations when solving math problems. These are: `Id`, `Add`, `Subtract`, `Multiply`, `Divide`, `Power`, `Log`, `Sqrt`, `Sine`, `Cosine`, `Tangent`, `Factorial`, and `Choose` (number of combinations). We also provide 2 operations to convert between `Radians` and `Degrees`, as these are needed for the sine, cosine and tangent operations. There are 6 operations that convert floating point numbers into strings and vice-versa. These include the `Str_to_Float` and `Float_to_Str` operations described previously, as well as operations which convert between floating point numbers and fractions, since in many math problems the answers are in the form "3/4". For the same reason, an operation to convert between a floating point number and number grouped in thousands is also used (e.g. 1000000 to "1,000,000" or "1.000.000"). Finally, we define an operation (`Check`) that given the input string, searches through the list of options and returns a string with the option index in {"A", "B", "C", "D", "E"}. If the input value does not match any of the options, or more than one option contains that value, it cannot be applied. For instance, in Problem 2, once the correct probability "1/221" is generated, by applying the check operation to this number we can obtain correct option "C".

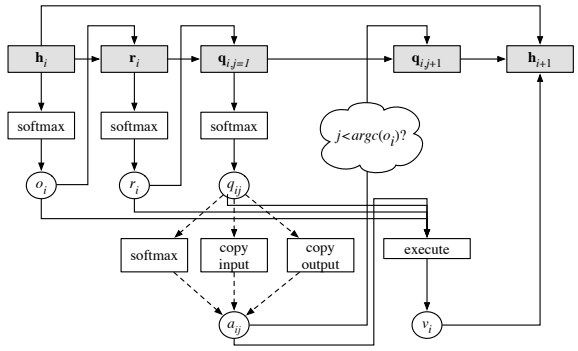


Figure 3: Illustration of the generation process of a single instruction tuple at timestamp i.

3.3 Generating and Executing Instructions

In our model, programs consist of sequences of instructions, $\boldsymbol{z}$. We turn now to how we model each z_i, conditional on the text program specification, and the program's history. The instruction z_i is a tuple consisting of an operation (o_i), an ordered sequence of its arguments ($\boldsymbol{a}_i$), and a decision about where its results will be placed (r_i) (is it appended in the output $\boldsymbol{y}$ or in a memory buffer $\boldsymbol{m}$?), and the result of applying the operation to its arguments (v_i). That is, $z_i = (o_i, r_i, \boldsymbol{a}_i, v_i)$.

Formally, o_i is an element of the pre-specified set of operations $\mathcal{O}$, which contains, for example `add`, `div`, `Str_to_Float`, etc. The number of arguments required by o_i is given by $argc(o_i)$, e.g., $argc(\texttt{add}) = 2$ and $argc(\texttt{log}) = 1$. The arguments are $\boldsymbol{a}_i = a_{i,1}, \ldots, a_{i,argc(o_i)}$. An instruction will generate a return value v_i upon execution, which will either be placed in the output $\boldsymbol{y}$ or hidden. This decision is controlled by r_i. We define the instruction probability as:

$$
\begin{aligned}
p(o_i, \boldsymbol{a}_i, r_i, v_i \mid \boldsymbol{z}_{<i}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{m}) = \\
p(o_i \mid \boldsymbol{z}_{<i}, \boldsymbol{x}) \times p(r_i \mid \boldsymbol{z}_{<i}, \boldsymbol{x}, o_i) \times \\
\prod_{j=1}^{argc(o_i)} p(a_{i,j} \mid \boldsymbol{z}_{<i}, \boldsymbol{x}, o_i, \boldsymbol{m}, \boldsymbol{y}) \times \\
[v_i = \text{apply}(o_i, \boldsymbol{a})],
\end{aligned}
$$

where $[p]$ evaluates to 1 if p is true and 0 otherwise, and $\text{apply}(f, \boldsymbol{x})$ evaluates the operation f with arguments $\boldsymbol{x}$. Note that the apply function is not learned, but pre-defined.

The network used to generate an instruction at a given timestamp i is illustrated in Figure 3. We first use the recurrent state $\mathbf{h}_i$ to generate $p(o_i \mid \boldsymbol{z}_{<i}, \boldsymbol{x}) = \underset{o_i \in \mathcal{O}}{\text{softmax}}(\mathbf{h}_i)$, using a softmax over the set of available operations $\mathcal{O}$.

In order to predict r_i, we generate a new hidden state $\mathbf{r}_i$, which is a function of the current program context $\mathbf{h}_i$, and an embedding of the current predicted operation, o_i. As the output can either be placed in the memory $\boldsymbol{m}$ or the output $\boldsymbol{y}$, we compute the probability $p(r_i = \text{OUTPUT} \mid \boldsymbol{z}_{<i}, \boldsymbol{x}, o_i) = \sigma(\mathbf{r}_i \cdot \mathbf{w}_r + b_r)$, where σ is the logistic sigmoid function. If $r_i = \text{OUTPUT}$, v_i is appended to the output $\boldsymbol{y}$; otherwise it is appended to the memory $\boldsymbol{m}$.

Once we generate r_i, we must predict $\boldsymbol{a}_i$, the $argc(o_i)$-length sequence of arguments that operation o_i requires. The jth argument $a_{i,j}$ can be either generated from a softmax over the vocabulary, copied from the input vector $\boldsymbol{x}$, or copied from previously generated values in the output $\boldsymbol{y}$ or memory $\boldsymbol{m}$. This decision is modeled using a latent predictor network (Ling et al., 2016), where the control over which method used to generate $a_{i,j}$ is governed by a latent variable $q_{i,j} \in \{\text{SOFTMAX}, \text{COPY-INPUT}, \text{COPY-OUTPUT}\}$. Similar to when predicting r_i, in order to make this choice, we also generate a new hidden state for each argument slot j, denoted by $\mathbf{q}_{i,j}$ with an LSTM, feeding the previous argument in at each time step, and initializing it with $\mathbf{r}_i$ and by reading the predicted value of the output r_i.

- If $q_{i,j} = \text{SOFTMAX}$, $a_{i,j}$ is generated by sampling from a softmax over the vocabulary $\mathcal{Y}$,

  $$p(a_{i,j} \mid q_{i,j} = \text{SOFTMAX}) = \underset{a_{i,j} \in \mathcal{Y}}{\text{softmax}}(\mathbf{q}_{i,j}).$$

  This corresponds to a case where a string is used as argument (e.g. y_1="Let").

- If $q_{i,j} = \text{COPY-INPUT}$, $a_{i,j}$ is obtained by copying an element from the input vector with a pointer network (Vinyals et al., 2015) over input words $x_1, \ldots, x_{|\boldsymbol{x}|}$, represented by their encoder LSTM state $\mathbf{u}_1, \ldots, \mathbf{u}_{|\boldsymbol{x}|}$. As such, we compute the probability distribution over input words as:

  $$p(a_{i,j} \mid q_{i,j} = \text{COPY-INPUT}) = \underset{a_{i,j} \in x_1, \ldots, x_{|\boldsymbol{x}|}}{\text{softmax}} \left(f(\mathbf{u}_{a_{i,j}}, \mathbf{q}_{i,j})\right) \quad (1)$$

  Function f computes the affinity of each token $x_{a_{i,j}}$ and the current output context $\mathbf{q}_{i,j}$. A common implementation of f, which we follow, is to apply a linear projection from $[\mathbf{u}_{a_{i,j}}; \mathbf{q}_{i,j}]$ into a fixed size vector (where $[\mathbf{u}; \mathbf{v}]$ is vector concatenation), followed by a $\tanh$ and a linear projection into a single value.

- If $q_{i,j} =$ COPY-OUTPUT, the model copies from either the output $\boldsymbol{y}$ or the memory $\boldsymbol{m}$. This is equivalent to finding the instruction z_i, where the value was generated. Once again, we define a pointer network that points to the output instructions and define the distribution over previously generated instructions as:

$$p(a_{i,j} \mid q_{i,j} = \text{COPY-OUTPUT}) = \underset{a_{i,j} \in z_1, \ldots, z_{i-1}}{\text{softmax}} \left(f(\mathbf{h}_{a_{i,j}}, \mathbf{q}_{i,j}) \right)$$

  Here, the affinity is computed using the decoder state $\mathbf{h}_{a_{i,j}}$ and the current state $\mathbf{q}_{i,j}$.

Finally, we embed the argument $a_{i,j}$[1] and the state $\mathbf{q}_{i,j}$ to generate the next state $\mathbf{q}_{i,j+1}$. Once all arguments for o_i are generated, the operation is executed to obtain v_i. Then, the embedding of v_i, the final state of the instruction $\mathbf{q}_{i,|\boldsymbol{a}_i|}$ and the previous state $\mathbf{h}_i$ are used to generate the state at the next timestamp $\mathbf{h}_{i+1}$.

4 Inducing Programs while Learning

The set of instructions $\boldsymbol{z}$ that will generate $\boldsymbol{y}$ is unobserved. Thus, given $\boldsymbol{x}$ we optimize the marginal probability function:

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}} p(\boldsymbol{y} \mid \boldsymbol{z}) p(\boldsymbol{z} \mid \boldsymbol{x}) = \sum_{\boldsymbol{z} \in \mathcal{Z}(y)} p(\boldsymbol{z} \mid \boldsymbol{x}),$$

where $p(\boldsymbol{y} \mid \boldsymbol{z})$ is the Kronecker delta function $\delta_{e(\boldsymbol{z}),\boldsymbol{y}}$, which is 1 if the execution of $\boldsymbol{z}$, denoted as $e(\boldsymbol{z})$, generates y and 0 otherwise. Thus, we can redefine $p(\boldsymbol{y}|\boldsymbol{x})$, the marginal over all programs $\mathcal{Z}$, as a marginal over programs that would generate $\boldsymbol{y}$, defined as $\mathcal{Z}(\boldsymbol{y})$. As marginalizing over $\boldsymbol{z} \in \mathcal{Z}(\boldsymbol{y})$ is intractable, we approximate the marginal by generating samples from our model. Denote the set of samples that are generated by $\hat{\mathcal{Z}}(\boldsymbol{y})$. We maximize $\sum \boldsymbol{z} \in \hat{\mathcal{Z}}(\boldsymbol{y}) p(\boldsymbol{z}|\boldsymbol{x})$.

However, generating programs that generate $\boldsymbol{y}$ is not trivial, as randomly sampling from the RNN distribution over instructions at each timestamp is unlikely to generate a sequence $\boldsymbol{z} \in \mathcal{Z}(\boldsymbol{y})$.

This is analogous to the question answering work in Liang et al. (2016), where the query that generates the correct answer must be found during inference, and training proved to be difficult without supervision. In Roy and Roth (2015) this problem is also addressed by adding prior knowledge to constrain the exponential space.

In our work, we leverage the fact that we are generating rationales, where there is a sense of progression within the rationale. That is, we assume that the rationale solves the problem step by step. For instance, in Problem 2, the rationale first describes the number of combinations of two cards in a deck of 52 cards, then describes the number of combinations of two kings, and finally computes the probability of drawing two kings. Thus, while generating the final answer without the rationale requires a long sequence of latent instructions, generating each of the tokens of the rationale requires far less operations.

More formally, given the sequence $z_1, \ldots, z_{i-1}$ generated so far, and the possible values for z_i given by the network, denoted $\mathcal{Z}_i$, we wish to filter $\mathcal{Z}_i$ to $\mathcal{Z}_i(y_k)$, which denotes a set of possible options that contain at least one path capable of generating the next token at index k. Finding the set $\mathcal{Z}_i(y_k)$ is achieved by testing all combinations of instructions that are possible with at most one level of indirection, and keeping those that can generate y_k. This means that the model can only generate one intermediate value in memory (not including the operations that convert strings into floating point values and vice-versa).

Decoding. During decoding we find the most likely sequence of instructions $\boldsymbol{z}$ given $\boldsymbol{x}$, which can be performed with a stack-based decoder. However, it is important to refer that each generated instruction $z_i = (o_i, r_i, a_{i,1}, \ldots, a_{i,|a_i|}, v_i)$ must be executed to obtain v_i. To avoid generating unexecutable code—e.g., log(0)—each hypothesis instruction is executed and removed if an error occurs. Finally, once the "⟨EOR⟩" tag is generated, we only allow instructions that would generate one of the option "A" to "E" to be generated, which guarantees that one of the options is chosen.

5 Staged Back-propagation

As it is shown in Figure 2, math rationales with more than 200 tokens are not uncommon, and with additional intermediate instructions, the size $\boldsymbol{z}$ can easily exceed 400. This poses a practical challenge for training the model.

For both the attention and copy mechanisms,

[1] The embeddings of a given argument $a_{i,j}$ and the return value v_i are obtained with a lookup table embedding and two flags indicating whether it is a string and whether it is a float. Furthermore, if the the value is a float we also add its numeric value as a feature.

for each instruction z_i, the model needs to compute the probability distribution between all the attendable units $\boldsymbol{c}$ conditioned on the previous state $\mathbf{h}_{i-1}$. For the attention model and input copy mechanisms, $\boldsymbol{c} = \boldsymbol{x_{0,i-1}}$ and for the output copy mechanism $\boldsymbol{c} = \boldsymbol{z}$. These operations generally involve an exponential number of matrix multiplications as the size of $\boldsymbol{c}$ and $\boldsymbol{z}$ grows. For instance, during the computation of the probabilities for the input copy mechanism in Equation 1, the affinity function f between the current context $\boldsymbol{q}$ and a given input $\boldsymbol{u}_k$ is generally implemented by projecting $\boldsymbol{u}$ and $\boldsymbol{q}$ into a single vector followed by a non-linearity, which is projected into a single affinity value. Thus, for each possible input $\boldsymbol{u}$, 3 matrix multiplications must be performed. Furthermore, for RNN unrolling, parameters and intermediate outputs for these operations must be replicated for each timestamp. Thus, as $\boldsymbol{z}$ becomes larger the attention and copy mechanisms quickly become a memory bottleneck as the computation graph becomes too large to fit on the GPU. In contrast, the sequence-to-sequence model proposed in (Sutskever et al., 2014), does not suffer from these issues as each timestamp is dependent only on the previous state $\mathbf{h}_{i-1}$.

To deal with this, we use a training method we call **staged back-propagation** which saves memory by considering slices of K tokens in $\boldsymbol{z}$, rather than the full sequence. That is, to train on a mini-batch where $|z| = 300$ with $K = 100$, we would actually train on 3 mini-batches, where the first batch would optimize for the first $\boldsymbol{z}_{1:100}$, the second for $\boldsymbol{z}_{101:200}$ and the third for $\boldsymbol{z}_{201:300}$. The advantage of this method is that memory intensive operations, such as attention and the copy mechanism, only need to be unrolled for K steps, and K can be adjusted so that the computation graph fits in memory.

However, unlike truncated back-propagation for language modeling, where context outside the scope of K is ignored, sequence-to-sequence models require global context. Thus, the sequence of states $\boldsymbol{h}$ is still built for the whole sequence $\boldsymbol{z}$. Afterwards, we obtain a slice $\boldsymbol{h}_{j:j+K}$, and compute the attention vector.[2] Finally, the prediction of the instruction is conditioned on the LSTM state and the attention vector.

[2]This modeling strategy is sometimes known as late fusion, as the attention vector is not used for state propagation, it is incorporated "later".

6 Experiments

We apply our model to the task of generating rationales for solutions to math problems, evaluating it on both the quality of the rationale and the ability of the model to obtain correct answers.

6.1 Baselines

As the baseline we use the attention-based sequence to sequence model proposed by Bahdanau et al. (2014), and proposed augmentations, allowing it to copy from the input (Ling et al., 2016) and from the output (Merity et al., 2016).

6.2 Hyperparameters

We used a two-layer LSTM with a hidden size of $H = 200$, and word embeddings with size 200. The number of levels that the graph $\mathcal{G}$ is expanded during sampling D is set to 5. Decoding is performed with a beam of 200. As for the vocabulary of the softmax and embeddings, we keep the most frequent 20,000 word types, and replace the rest of the words with an unknown token. During training, the model only learns to predict a word as an unknown token, when there is no other alternative to generate the word.

6.3 Evaluation Metrics

The evaluation of the rationales is performed with average sentence level perplexity and BLEU-4 (Papineni et al., 2002). When a model cannot generate a token for perplexity computation, we predict unknown token. This benefits the baselines as they are less expressive. As the perplexity of our model is dependent on the latent program that is generated, we force decode our model to generate the rationale, while maximizing the probability of the program. This is analogous to the method used to obtain sample programs described in Section 4, but we choose the most likely instructions at each timestamp instead of sampling. Finally, the correctness of the answer is evaluated by computing the percentage of the questions, where the chosen option matches the correct one.

6.4 Results

The test set results, evaluated on perplexity, BLEU, and accuracy, are presented in Table 3.

Perplexity. In terms of perplexity, we observe that the regular sequence to sequence model fares poorly on this dataset, as the model requires the generation of many values that tend to be

Model	Perplexity	BLEU	Accuracy
Seq2Seq	524.7	8.57	20.8
+Copy Input	46.8	21.3	20.4
+Copy Output	45.9	20.6	20.2
Our Model	**28.5**	**27.2**	**36.4**

Table 3: Results over the test set measured in Perplexity, BLEU and Accuracy.

sparse. Adding an input copy mechanism greatly improves the perplexity as it allows the generation process to use values that were mentioned in the question. The output copying mechanism improves perplexity slightly over the input copy mechanism, as many values are repeated after their first occurrence. For instance, in Problem 2, the value "1326" is used twice, so even though the model cannot generate it easily in the first occurrence, the second one can simply be generated by copying the first one. We can observe that our model yields significant improvements over the baselines, demonstrating that the ability to generate new values by algebraic manipulation is essential in this task. An example of a program that is inferred is shown in Figure 4. The graph was generated by finding the most likely program $\boldsymbol{z}$ that generates $\boldsymbol{y}$. Each node isolates a value in $\boldsymbol{x}$, $\boldsymbol{m}$, or $\boldsymbol{y}$, where arrows indicate an operation executed with the outgoing nodes as arguments and incoming node as the return of the operation. For simplicity, operations that copy or convert values (e.g. from string to float) were not included, but nodes that were copied/converted share the same color. Examples of tokens where our model can obtain the perplexity reduction are the values "0.025", "0.023", "0.002" and finally the answer "E" , as these cannot be copied from the input or output.

BLEU. We observe that the regular sequence to sequence model achieves a low BLEU score. In fact, due to the high perplexities the model generates very short rationales, which frequently consist of segments similar to "Answer should be D", as most rationales end with similar statements. By applying the copy mechanism the BLEU score improves substantially, as the model can define the variables that are used in the rationale. Interestingly, the output copy mechanism adds no further improvement in the perplexity evaluation. This is because during decoding all values that can be copied from the output are values that could have been generated by the model either from the softmax or the input copy mechanism. As such, adding an output copying mechanism adds little to the expressiveness of the model during decoding.

Finally, our model can achieve the highest BLEU score as it has the mechanism to generate the intermediate and final values in the rationale.

Accuracy. In terms of accuracy, we see that all baseline models obtain values close to chance (20%), indicating that they are completely unable to solve the problem. In contrast, we see that our model can solve problems at a rate that is significantly higher than chance, demonstrating the value of our program-driven approach, and its ability to learn to generate programs.

In general, the problems we solve correctly correspond to simple problems that can be solved in one or two operations. Examples include questions such as "Billy cut up each cake into 10 slices, and ended up with 120 slices altogether. How many cakes did she cut up? A) 9 B) 7 C) 12 D) 14 E) 16", which can be solved in a single step. In this case, our model predicts "120 / 10 = 12 cakes. Answer is C" as the rationale, which is reasonable.

6.5 Discussion.

While we show that our model can outperform the models built up to date, generating complex rationales as those shown in Figure 1 correctly is still an unsolved problem, as each additional step adds complexity to the problem both during inference and decoding. Yet, this is the first result showing that it is possible to solve math problems in such a manner, and we believe this modeling approach and dataset will drive work on this problem.

7 Related Work

Extensive efforts have been made in the domain of math problem solving (Hosseini et al., 2014; Kushman et al., 2014; Roy and Roth, 2015), which aim at obtaining the correct answer to a given math problem. Other work has focused on learning to map math expressions into formal languages (Roy et al., 2016). We aim to generate natural language rationales, where the bindings between variables and the problem solving approach are mixed into a single generative model that attempts to solve the problem while explaining the approach taken.

Our approach is strongly tied with the work on sequence to sequence transduction using the encoder-decoder paradigm (Sutskever et al., 2014;

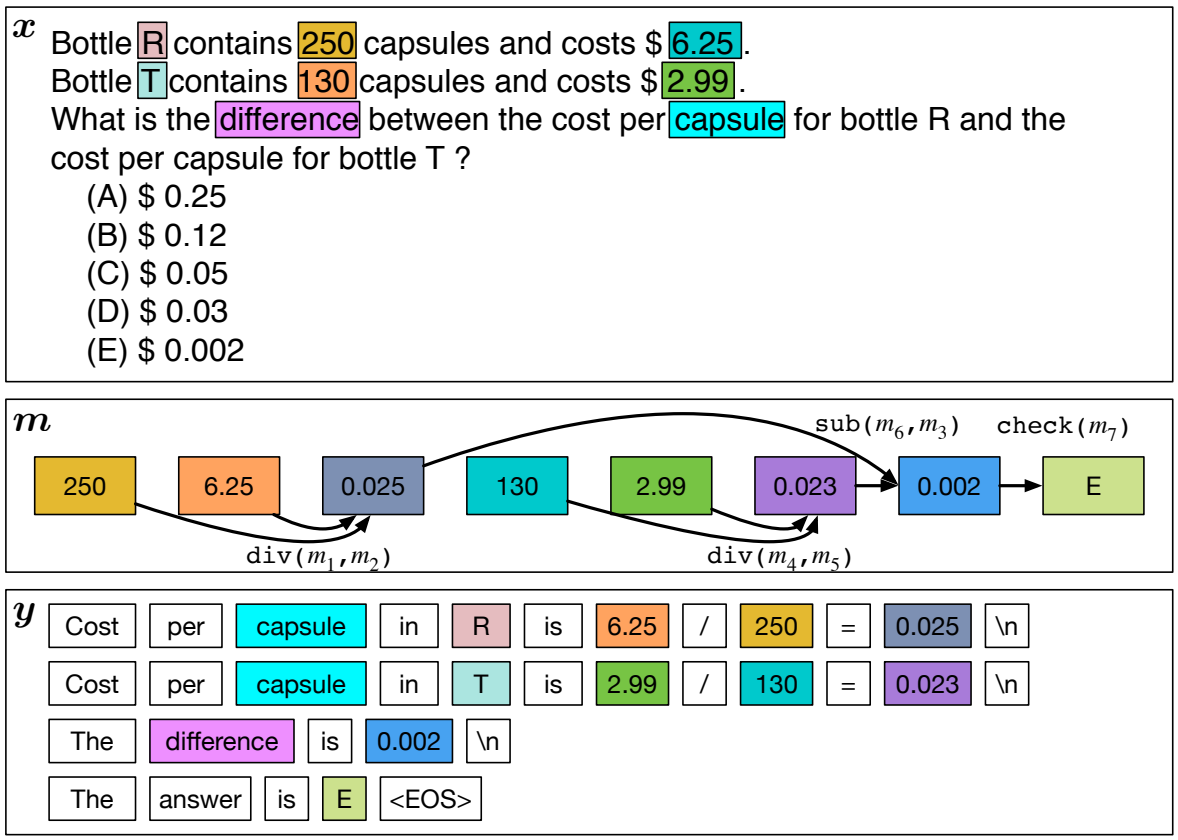


Figure 4: Illustration of the most likely latent program inferred by our algorithm to explain a held-out question-rationale pair.

Bahdanau et al., 2014; Kalchbrenner and Blunsom, 2013), and inherits ideas from the extensive literature on semantic parsing (Jones et al., 2012; Berant et al., 2013; Andreas et al., 2013; Quirk et al., 2015; Liang et al., 2016; Neelakantan et al., 2016) and program generation (Reed and de Freitas, 2016; Graves et al., 2016), namely, the usage of an external memory, the application of different operators over values in the memory and the copying of stored values into the output sequence.

Providing textual explanations for classification decisions has begun to receive attention, as part of increased interest in creating models whose decisions can be interpreted. Lei et al. (2016), jointly modeled both a classification decision, and the selection of the most relevant subsection of a document for making the classification decision. Hendricks et al. (2016) generate textual explanations for visual classification problems, but in contrast to our model, they first generate an answer, and then, conditional on the answer, generate an explanation. This effectively creates a post-hoc justification for a classification decision rather than a program for deducing an answer. These papers, like ours, have jointly modeled rationales and answer predictions; however, we are the first to use rationales to guide program induction.

8 Conclusion

In this work, we addressed the problem of generating rationales for math problems, where the task is to not only obtain the correct answer of the problem, but also generate a description of the method used to solve the problem. To this end, we collect 100,000 question and rationale pairs, and propose a model that can generate natural language and perform arithmetic operations in the same decoding process. Experiments show that our method outperforms existing neural models, in both the fluency of the rationales that are generated and the ability to solve the problem.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proc. of ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Ben-

gio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv* 1409.0473.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proc. of EMNLP*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwiska, Sergio Gmez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adri Puigdomnech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538(7626):471–476.

Brent Harrison, Upol Ehsan, and Mark O. Riedl. 2017. Rationalization: A neural machine translation approach to generating natural language explanations. *CoRR* abs/1702.07826.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating visual explanations. In *Proc. ECCV*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proc. of EMNLP*.

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proc. of ACL*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proc. of EMNLP*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proc. of ACL*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proc. of EMNLP*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv* 1611.00020.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. In *Proc. of ACL*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv* 1609.07843.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *Proc. ICLR*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. of ACL*.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proc. of ACL*.

Scott E. Reed and Nando de Freitas. 2016. Neural programmer-interpreters. In *Proc. of ICLR*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proc. of EMNLP*.

Subhro Roy, Shyam Upadhyay, and Dan Roth. 2016. Equation parsing: Mapping sentences to grounded equations. In *Proc. of EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *arXiv* 1409.3215.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proc. of NIPS*.

Automatically Generating Rhythmic Verse with Neural Networks

Jack Hopkins
Computer Laboratory
University of Cambridge
jack.hopkins@me.com

Douwe Kiela
Facebook AI Research
dkiela@fb.com

Abstract

We propose two novel methodologies for the automatic generation of rhythmic poetry in a variety of forms. The first approach uses a neural language model trained on a phonetic encoding to learn an implicit representation of both the *form* and *content* of English poetry. This model can effectively learn common poetic devices such as rhyme, rhythm and alliteration. The second approach considers poetry generation as a constraint satisfaction problem where a generative neural language model is tasked with learning a representation of content, and a discriminative weighted finite state machine constrains it on the basis of form. By manipulating the constraints of the latter model, we can generate coherent poetry with arbitrary forms and themes. A large-scale extrinsic evaluation demonstrated that participants consider machine-generated poems to be written by humans 54% of the time. In addition, participants rated a machine-generated poem to be the most human-like amongst all evaluated.

1 Introduction

Poetry is an advanced form of linguistic communication, in which a message is conveyed that satisfies both aesthetic and semantic constraints. As poetry is one of the most expressive forms of language, the *automatic* creation of texts recognisable as poetry is difficult. In addition to requiring an understanding of many aspects of language including phonetic patterns such as rhyme, rhythm and alliteration, poetry composition also requires a deep understanding of the meaning of language.

Poetry generation can be divided into two subtasks, namely the problem of *content*, which is concerned with a poem's semantics, and the problem of *form*, which is concerned with the aesthetic rules that a poem follows. These rules may describe aspects of the literary devices used, and are usually highly prescriptive. Examples of different forms of poetry are limericks, ballads and sonnets. Limericks, for example, are characterised by their strict rhyme scheme (AABBA), their rhythm (two unstressed syllables followed by one stressed syllable) and their shorter third and fourth lines. Creating such poetry requires not only an understanding of the language itself, but also of how it sounds when spoken aloud.

Statistical text generation usually requires the construction of a generative language model that explicitly learns the probability of any given word given previous context. Neural language models (Schwenk and Gauvain, 2005; Bengio et al., 2006) have garnered signficant research interest for their ability to learn complex syntactic and semantic representations of natural language (Mikolov et al., 2010; Sutskever et al., 2014; Cho et al., 2014; Kim et al., 2015). Poetry generation is an interesting application, since performing this task automatically requires the creation of models that not only focus on *what* is being written (content), but also on *how* it is being written (form).

We experiment with two novel methodologies for solving this task. The first involves training a model to learn an implicit representation of content and form through the use of a phonological encoding. The second involves training a generative language model to represent content, which is then constrained by a discriminative pronunciation model, representing form. This second model is of particular interest because poetry with arbitrary rhyme, rhythm, repetition and themes can be generated by tuning the pronunciation model.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 168–178
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1016

2 Related Work

Automatic poetry generation is an important task due to the significant challenges involved. Most systems that have been proposed can loosely be categorised as rule-based expert systems, or statistical approaches.

Rule-based poetry generation attempts include case-based reasoning (Gervás, 2000), template-based generation (Colton et al., 2012), constraint satisfaction (Toivanen et al., 2013; Barbieri et al., 2012) and text mining (Netzer et al., 2009). These approaches are often inspired by how humans might generate poetry.

Statistical approaches, conversely, make no assumptions about the creative process. Instead, they attempt to extract statistical patterns from existing poetry corpora in order to construct a language model, which can then be used to generate new poetic variants (Yi et al., 2016; Greene et al., 2010). Neural language models have been increasingly applied to the task of poetry generation. The work of Zhang and Lapata (2014) is one such example, where they were able to outperform all other classical Chinese poetry generation systems with both manual and automatic evaluation. Ghazvininejad et al. (2016) and Goyal et al. (2016) apply neural language models with regularising finite state machines. However, in the former case the rhythm of the output cannot be defined at sample time, and in the latter case the finite state machine is not trained on rhythm at all, as it is trained on dialogue acts. McGregor et al. (2016) construct a phonological model for generating prosodic texts, however there is no attempt to embed semantics into this model.

3 Phonetic-level Model

Our first model is a pure neural language model, trained on a phonetic encoding of poetry in order to represent both form and content. Phonetic encodings of language represent information as sequences of around 40 basic acoustic symbols. Training on phonetic symbols allows the model to learn effective representations of pronunciation, including rhyme and rhythm.

However, just training on a large corpus of poetry data is not enough. Specifically, two problems need to be overcome. 1) Phonetic encoding results in information loss: words that have the same pronunciation (homophones) cannot be perfectly reconstructed from the corresponding phonemes. This means that we require an additional probabilistic model in order to determine the most likely word given a sequence of phonemes. 2) The variety of poetry and poetic devices one can use—e.g., rhyme, rhythm, repetition—means that poems sampled from a model trained on *all* poetry would be unlikely to maintain internal consistency of meter and rhyme. It is therefore important to train the model on poetry which has its own internal consistency.

Thus, the model comprises three steps: transliterating an orthographic sequence to its phonetic representation, training a neural language model on the phonetic encoding, and decoding the generated sequence back from phonemes to orthographic symbols.

Phonetic encoding To solve the first step, we apply a combination of word lookups from the CMU pronunciation dictionary (Weide, 2005) with letter-to-sound rules for handling out-of-vocabulary words. These rules are based on the CART techniques described by Black et al. (1998), and are represented with a simple Finite State Transducer[1]. The number of letters and number of phones in a word are rarely a one-to-one match: letters may match with up to three phones. In addition, virtually all letters can, in some contexts, map to zero phones, which is known as 'wild' or epsilon. Expectation Maximisation is used to compute the probability of a single letter matching a single phone, which is maximised through the application of Dynamic Time Warping (Myers et al., 1980) to determine the most likely position of epsilon characters.

Although this approach offers full coverage over the training corpus—even for abbreviated words like *ask'd* and archaic words like *renewest*—it has several limitations. Irregularities in the English language result in difficulty determining general letter-to-sound rules that can manage words with unusual pronunciations such as "colonel" and "receipt" [2].

In addition to transliterating words into phoneme sequences, we also represent word break characters as a specific symbol. This makes

[1] Implemented using FreeTTS (Walker et al., 2010)

[2] An evaluation of models in American English, British English, German and French was undertaken by Black et al. (1998), who reported an externally validated per token accuracy on British English as low as 67%. Although no experiments were carried out on corpora of early-modern English, it is likely that this accuracy would be significantly lower.

decipherment, when converting back into an orthographic representation, much easier. Phonetic transliteration allows us to construct a phonetic poetry corpus comprising 1,046,536 phonemes.

Neural language model We train a Long-Short Term Memory network (Hochreiter and Schmidhuber, 1997) on the phonetic representation of our poetry corpus. The model is trained using stochastic gradient descent to predict the next phoneme given a sequence of phonemes. Specifically, we maximize a multinomial logistic regression objective over the final softmax prediction. Each phoneme is represented as a 256-dimensional embedding, and the model consists of two hidden layers of size 256. We apply backpropagation-through-time (Werbos, 1990) for 150 timesteps, which roughly equates to four lines of poetry in sonnet form. This allows the network to learn features like rhyme even when spread over multiple lines. Training is preemptively stopped at 25 epochs to prevent overfitting.

Orthographic decoding When decoding from phonemes back to orthographic symbols, the goal is to compute the most likely word corresponding to a sequence of phonemes. That is, we compute the most probable hypothesis word W given a phoneme sequence ρ:

$$arg\,max_i\ P\ (\ W_i \mid \rho\ ) \qquad (1)$$

We can consider the phonetic encoding of plaintext to be a *homophonic cipher*; that is, a cipher in which each symbol can correspond to one or more possible decodings. The problem of homophonic decipherment has received significant research attention in the past; with approaches utilising Expectation Maximisation (Knight et al., 2006), Integer Programming (Ravi and Knight, 2009) and A* search (Corlett and Penn, 2010).

Transliteration from phonetic to an orthographic representation is done by constructing a Hidden Markov Model using the CMU pronunciation dictionary (Weide, 2005) and an n-gram language model. We calculate the transition probabilities (using the n-gram model) and the emission matrix (using the CMU pronunciation dictionary) to determine pronunciations that correspond to a single word. All pronunciations are naively considered equiprobable. We perform Viterbi decoding to find the most likely sequence of words. This means finding the most likely word w_{t+1} given a previous word sequence $(w_{t-n}, ..., w_t)$.

$$arg\,max_{w_{t+1}}\ P\ (\ w_{t+1} \mid w_1,\ \ldots\ ,\ w_t\ ) \qquad (2)$$

If a phonetic sequence does not map to any word, we apply the heuristic of artificially breaking the sequence up into two subsequences at index *n*, such that *n* maximises the n-gram frequency of the subsequences.

And humble and their fit *flees* are wits size
but that one made and made thy step me lies

Cool light the golden dark in any way
the birds a *shade* a laughter turn away

Then adding wastes retreating white as thine
She watched what eyes are breathing awe what shine

But sometimes shines so covered how the beak
Alone in pleasant skies no more to seek

Figure 1: Example output of the phonetic-level model trained on Iambic Pentameter poetry (grammatical errors are emphasised).

Output A popular form of poetry with strict internal structure is the sonnet. Popularised in English by Shakespeare, the sonnet is characterised by a strict rhyme scheme and exactly fourteen lines of Iambic Pentameter (Greene et al., 2010). Since the 17,134 word tokens in Shakespeare's 153 sonnets are insufficient to train an effective model, we augment this corpus with poetry taken from the website *sonnets.org*, yielding a training set of 288,326 words and 1,563,457 characters.

An example of the output when training on this sonnets corpus is provided in Figure 1. Not only is it mostly in strict Iambic Pentameter, but the grammar of the output is mostly correct and the poetry contains rhyme.

4 Constrained Character-level Model

As the example shows, phonetic-level language models are effective at learning poetic form, despite small training sets and relatively few parameters. However, the fact that they require training data with internal poetic consistency implies that they do not generalise to other forms of poetry. That is, in order to generate poetry in Dactylic Hexameter (for example), a phonetic model must be trained on a corpus of Dactylic poetry. Not only is this impractical, but in many cases no corpus of

adequate size even exists. Even when such poetic corpora are available, a new model must be trained for each type of poetry. This precludes tweaking the form of the output, which is important when generating poetry automatically.

We now explore an alternative approach. Instead of attempting to represent both form and content in a single model, we construct a pipeline containing a generative language model representing *content*, and a discriminative model representing *form*. This allows us to represent the problem of creating poetry as a constraint satisfaction problem, where we can modify constraints to restrict the types of poetry we generate.

Character Language Model Rather than train a model on data representing features of *both* content and form, we now use a simple character-level model (Sutskever et al., 2011) focused solely on content. This approach offers several benefits over the word-level models that are prevalent in the literature. Namely, their more compact vocabulary allows for more efficient training; they can learn common prefixes and suffixes to allow us to sample words that are not present in the training corpus and can learn effective language representations from relatively small corpora; and they can handle archaic and incorrect spellings of words.

As we no longer need the model to explicitly represent the form of generated poetry, we can loosen our constraints when choosing a training corpus. Instead of relying on poetry only in sonnet form, we can instead construct a generic corpus of poetry taken from online sources. This corpus is composed of 7.56 million words and 34.34 million characters, taken largely from 20^{th} Century poetry books found online. The increase in corpus size facilitates a corresponding increase in the number of permissible model parameters. This allows us to train a 3-layer LSTM model with 2048-dimensional hidden layers, with embeddings in 128 dimensions. The model was trained to predict the next character given a sequence of characters, using stochastic gradient descent. We attenuate the learning rate over time, and by 20 epochs the model converges.

Rhythm Modeling Although a character-level language model trained on a corpus of generic poetry allows us to generate interesting text, internal irregularities and noise in the training data prevent the model from learning important features such as rhythm. Hence, we require an additional classifier to constrain our model by either accepting or rejecting sampled lines based on the presence or absence of these features. As the presence of *meter* (rhythm) is the most characteristic feature of poetry, it therefore must be our primary focus.

Pronunciation dictionaries have often been used to determine the syllabic stresses of words (Colton et al., 2012; Manurung et al., 2000; Misztal and Indurkhya, 2014), but suffer from some limitations for constructing a classifier. All word pronunciations are considered equiprobable, including archaic and uncommon pronunciations, and pronunciations are provided context free, despite the importance of context for pronunciation[3]. Furthermore, they are constructed from American English, meaning that British English may be misclassified.

These issues are circumvented by applying lightly supervised learning to determine the contextual stress pattern of any word. That is, we exploit the latent structure in our corpus of sonnet poetry, namely, the fact that sonnets are composed of lines in rigid Iambic Pentameter, and are therefore exactly ten syllables long with alternating syllabic stress. This allows us to derive a *syllable-stress distribution*. Although we use the sonnets corpus for this, it is important to note that any corpus with such a latent structure could be used.

We represent each line of poetry as a cascade of Weighted Finite State Transducers (WFST). A WFST is a finite-state automaton that maps between two sets of symbols. It is defined as an eight-tuple where $\langle Q, \Sigma, \rho, I, F, \Delta, \lambda, p \rangle$:

Q : A set of states

Σ : An input alphabet of symbols

ρ : An output alphabet of symbols

I : A set of initial states

F : A set of final states, or sinks

Δ : A transition function mapping pairs of states and symbols to sets of states

λ : A set of weights for initial states

P : A set of weights for final states

[3]For example, the independent probability of stressing the single syllable word *at* is 40%, but this increases to 91% when the following word is *the* (Greene et al., 2010)

A WFST assigns a probability (or weight, in the general case) to each path through it, going from an initial state to an end state. Every path corresponds to an input and output label sequence, and there can be many such paths for each sequence.

WFSTs are often used in a cascade, where a number of machines are executed in series, such that the output tape of one machine is the input tape for the next. Formally, a cascade is represented by the functional composition of several machines.

$$W(x, z) = A(x|y) \circ B(y|z) \circ C(z) \quad (3)$$

Where $W(x, z)$ is defined as the $\oplus$ sum of the path probabilities through the cascade, and x and z are an input sequence and output sequence respectively. In the real semiring (where the product of probabilities are taken in series, and the sum of the probabilities are taken in parallel), we can rewrite the definition of weighted composition to produce the following:

$$W(x, z) = \bigoplus_y A(x \mid y) \otimes B(y \mid z) \otimes C(z) \quad (4)$$

As we are dealing with probabilities, this can be rewritten as:

$$P(x, z) = \sum_y P(x \mid y)P(y \mid z)P(z) \quad (5)$$

We can perform Expectation Maximisation over the poetry corpus to obtain a probabilistic classifier which enables us to determine the most likely stress patterns for each word. Every word is represented by a single transducer.

In each cascade, a sequence of input words is mapped onto a sequence of stress patterns $\langle \times, / \rangle$ where each pattern is between 1 and 5 syllables in length[4]. We initially set all transition probabilities equally, as we make no assumptions about the stress distributions in our training set. We then iterate over each line of the sonnet corpus, using Expectation Maximisation to train the cascades. In practice, there are several de facto variations of Iambic meter which are permissible, as shown in Figure 2. We train the rhythm classifier by converging the cascades to whatever output is the most likely given the line.

[4]Words of more than 5 syllables comprise less than 0.1% of the lexicon (Aoyama and Constable, 1998).

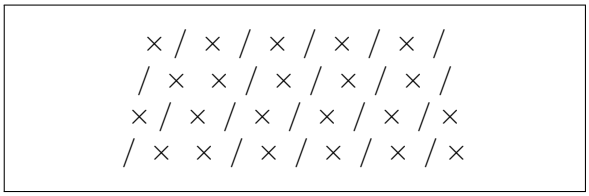
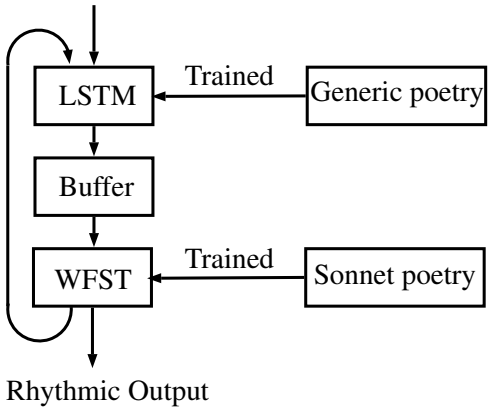


Figure 2: Permissible variations of Iambic Pentameter in Shakespeare's sonnets.



Constraining the model To generate poetry using this model, we sample sequences of characters from the character-level language model. To impose rhythm constrains on the language model, we first represent these sampled characters at the word level and pool sampled characters into word tokens in an intermediary buffer. We then apply the separately trained word-level WFSTs to construct a cascade of this buffer and perform Viterbi decoding over the cascade. This defines the distribution of stress-patterns over our word tokens.

We can represent this cascade as a probabilistic classifier, and accept or reject the buffered output based on how closely it conforms to the desired meter. While sampling sequences of words from this model, the entire generated sequence is passed to the classifier each time a new word is sampled. The pronunciation model then returns the probability that the entire line is within the specified meter. If a new word is rejected by the classifier, the state of the network is rolled back to the last formulaically acceptable state of the line, removing the rejected word from memory. The constraint on rhythm can be controlled by adjusting the acceptability threshold of the classifier. By increasing the threshold, output focuses on form over content. Conversely, decreasing the criterion puts greater emphasis on content.

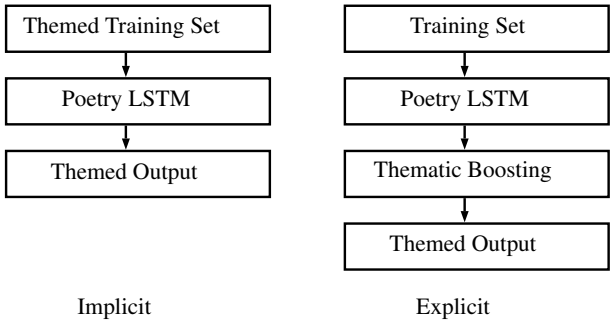


Figure 3: Two approaches for generating themed poetry.

Errors per line	1	2	3	4	Total
Phonetic Model	11	2	3	1	28
Character Model + WFST	6	5	1	1	23
Character Model	3	8	7	7	68

Table 1: Number of lines with n errors from a set of 50 lines generated by each of the three models.

4.1 Themes and Poetic devices

It is important for any generative poetry model to include themes and poetic devices. One way to achieve this would be by constructing a corpus that exhibits the desired themes and devices. To create a themed corpus about 'love', for instance, we would aggregate love poetry to train the model, which would thus learn an implicit representation of love. However, this forces us to generate poetry according to discrete themes and styles from pre-trained models, requiring a new training corpus for each model. In other words, we would suffer from similar limitations as with the phonetic-level model, in that we require a dedicated corpus. Alternatively, we can manipulate the language model by boosting character probabilities at sample time to increase the probability of sampling thematic words like 'love'. This approach is more robust, and provides us with more control over the final output, including the capacity to vary the inclusion of poetic devices in the output.

Themes In order to introduce thematic content, we heuristically boost the probability of sampling words that are semantically related to a theme word from the language model. First, we compile a list of similar words to a *key* theme word by retrieving its semantic neighbours from a distributional semantic model (Mikolov et al., 2013). For example, the theme *winter* might include thematic words *frozen*, *cold*, *snow* and *frosty*. We represent these semantic neighbours at the character level, and heuristically boost their probability by multiplying the sampling probability of these character strings by their cosine similarity to the key word, plus a constant. Thus, the likelihood of sampling a thematically related word is artificially increased, while still constraining the model rhythmically.

Poetic devices A similar method may be used for poetic devices such as assonance, consonance and alliteration. Since these devices can be orthographically described by the repetition of identical sequences of characters, we can apply the same heuristic to boost the probability of sampling character strings that have previously been sampled. That is, to sample a line with many instances of alliteration (multiple words with the same initial sound) we record the historical frequencies of characters sampled at the beginning of each previous word. After a word break character, we boost the probability that those characters will be sampled again in the softmax. We only keep track of frequencies for a fixed number of time steps. By increasing or decreasing the size of this window, we can manipulate the prevalence of alliteration. Variations of this approach are applied to invoke consonance (by boosting intra-word consonants) and assonance (by boosting intra-word vowels). An example of two sampled lines with high degrees of alliteration, assonance and consonance is given in Figure 4c.

5 Evaluation

In order to examine how effective our methodologies for generating poetry are, we evaluate the proposed models in two ways. First, we perform an intrinsic evaluation where we examine the quality of the models and the generated poetry. Second, we perform an extrinsic evaluation where we evaluate the generated output using human annotators, and compare it to human-generated poetry.

5.1 Intrinsic evaluation

To evaluate the ability of both models to generate formulaic poetry that adheres to rhythmic rules, we compared sets of fifty sampled lines from each model. The first set was sampled from the phonetic-level model trained on Iambic poetry. The second set was sampled from the character-level model, constrained to Iambic form. For com-

	Word	Line	Coverage
Wikipedia	64.84%	83.35%	97.53%
Sonnets	85.95%	80.32%	99.36%

Table 2: Error when transliterating text into phonemes and reconstructing back into text.

parison, and to act as a baseline, we also sampled from the *unconstrained* character model.

We created gold-standard syllabic classifications by recording each line spoken-aloud, and marking each syllable as either stressed or unstressed. We then compared these observations to loose Iambic Pentameter (containing all four variants), to determine how many syllabic misclassifications existed on each line. This was done by speaking each line aloud, and noting where the speaker put stresses.

As Table 1 shows, the constrained character level model generated the most formulaic poetry. Results from this model show that 70% of lines had zero mistakes, with frequency obeying an inverse power-law relationship with the number of errors. We can see that the phonetic model performed similarly, but produced more subtle mistakes than the constrained character model: many of the errors were single mistakes in an otherwise correct line of poetry.

In order to investigate this further, we examined to what extent these errors are due to transliteration (i.e., the phonetic encoding and orthographic decoding steps). Table 2 shows the reconstruction accuracy per word and per line when transliterating either Wikipedia or Sonnets to phonemes using the CMU pronunciation dictionary and subsequently reconstructing English text using the n-gram model[5]. Word accuracy reflects the frequency of *perfect* reconstruction, whereas per line tri-gram similarity (Kondrak, 2005) reflects the *overall* reconstruction. Coverage captures the percentage of in-vocabulary items. The relatively low per-word accuracy achieved on the Wikipedia corpus is likely due to the high frequency of out-of-vocabulary words. The results show that a significant number of errors in the phonetic-level model are likely to be caused by transliteration mistakes.

[5]Obviously, calculating this value for the character-level model makes no sense, since no transliteration occurs in that case.

5.2 Extrinsic evaluation

We conducted an indistinguishability study with a selection of automatically generated poetry and human poetry. As extrinsic evaluations are expensive and the phonetic model was unlikely to do well (as illustrated in Figure 4e: the model generates good Iambic form, but not very good English), we only evaluate on the constrained character-level model. Poetry was generated with a variety of themes and poetic devices (see supplementary material).

The aim of the study was to determine whether participants could distinguish between human and machine-generated poetry, and if so to what extent. A set of 70 participants (of whom 61 were English native speakers) were each shown a selection of randomly chosen poetry segments, and were invited to classify them as either human or generated. Participants were recruited from friends and people within poetry communities within the University of Cambridge, with an age range of 17 to 80, and a mean age of 29. Our participants were not financially incentivised, perceiving the evaluation as an intellectual challenge.

In addition to the classification task, each participant was also invited to rate each poem on a 1-5 scale with respect to three criteria, namely readability, form and evocation (how much emotion did a poem elicit). We naively consider the overall quality of a poem to be the mean of these three measures. We used a custom web-based environment, built specifically for this evaluation[6], which is illustrated in Figure 5. Based on human judgments, we can determine whether the models presented in this work can produce poetry of a similar quality to humans.

To select appropriate human poetry that could be meaningfully compared with the machine-generated poetry, we performed a comprehension test on all poems used in the evaluation, using the Dale-Chall readability formula (Dale and Chall, 1948). This formula represents readability as a function of the complexity of the input words. We selected nine machine-generated poems with a high readability score. The generated poems produced an average score of 7.11, indicating that readers over 15 years of age should easily be able to comprehend them.

For our human poems, we focused explicitly on poetry where greater consideration is placed on

[6]http://neuralpoetry.getforge.io/

(a)
The crow crooked on more beautiful and free,
He journeyed off into the quarter sea.
his radiant ribs girdled empty and very -
least beautiful as dignified to see.

(b)
Is that people like things
(are the way we to figure it
out) and I thought of you
reading and then is your
show or you know we will
finish along will you play.

(c)
Man with the broken blood blue glass and gold.
Cheap chatter chants to be a lover do.

(d)
How dreary to be somebody,
How public like a frog
To tell one's name the livelong day
To an admiring bog.

(e)
The son still streams and strength and spirit.
The ridden souls of which the fills of.

Figure 4: Examples of automatically generated and human generated poetry. (a) Character-level model - Strict rhythm regularisation - Iambic - No Theme. (b) Character-level model - Strict rhythm regularisation - Anapest. (c) Character-level model - Boosted alliteration/assonance. (d) Emily Dickinson - I'm nobody, who are you? (e) Phonetic-level model - Nonsensical Iambic lines.

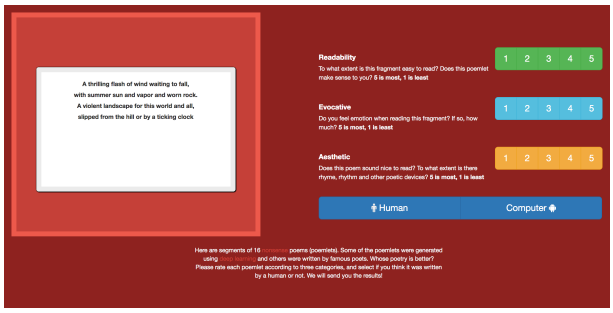


Figure 5: The experimental environment for asking participants to distinguish between automatically generated and human poetry.

prosodic elements like rhythm and rhyme than semantic content (known as "nonsense verse"). We randomly selected 30 poems belonging to that category from the website *poetrysoup.com*, of which eight were selected for the final comparison based on their comparable readability score. The selected poems were segmented into passages of between four and six lines, to match the length of the generated poetry segments. An example of such a segment is shown in Figure 4d. The human poems had an average score of 7.52, requiring a similar level of English aptitude to the generated texts.

The performance of each human poem, alongside the aggregated scores of the generated poems, is illustrated in Table 3. For the *human* poems, our group of participants guessed correctly that they were human 51.4% of the time. For the *generated* poems, our participants guessed correctly 46.2% of the time that they were machine generated. To determine whether our results were statistically significant, we performed a Chi^2 test. This resulted in a p-value of 0.718. This indicates that our participants were unable to tell the difference between human and generated poetry in any significant way. Although our participants generally considered the human poems to be of marginally higher quality than our generated poetry, they were unable to effectively distinguish between them. Interestingly, our results seem to suggest that our participants consider the generated poems to be more 'human-like' than those actually written by humans. In addition, the poem with the highest overall quality rating is a machine generated one. This shows that our approach was effective at generating high-quality rhythmic verse.

It should be noted that the poems that were most 'human-like' and most aesthetic respectively were generated by the neural character model. Generally the set of poetry produced by the neural character model was slightly less readable and emotive than the human poetry, but had above average form. All generated poems included in this evaluation can be found in the supplementary material, and our code is made available online[7].

[7]https://github.com/JackHopkins/ACLPoetry

Poet	Title	Human	Readability	Emotion	Form
Generated	Best	0.66	0.60	-0.77	0.90
G. M. Hopkins	Carrion Comfort	0.62	-1.09	1.39	-1.55
J. Thornton	Delivery of Death	0.60	0.26	-1.38	-0.65
Generated	*Mean*	*0.54*	*-0.28*	*-0.30*	*0.23*
M. Yvonne	Intricate Weave	0.53	2.38	0.94	-1.67
E. Dickinson	I'm Nobody	0.52	-0.46	0.92	0.44
G. M. Hopkins	The Silver Jubilee	0.52	0.71	-0.33	0.65
R. Dryden	Mac Flecknoe	0.51	-0.01	0.35	-0.78
A. Tennyson	Beautiful City	0.48	-1.05	0.97	-1.26
W. Shakespeare	A Fairy Song	0.45	0.65	1.30	1.18

Table 3: Proportion of people classifying each poem as 'human', as well as the relative qualitative scores of each poem as deviations from the mean.

6 Conclusions

Our contributions are twofold. First, we developed a neural language model trained on a phonetic transliteration of poetic form and content. Although example output looked promising, this model was limited by its inability to generalise to novel forms of verse. We then proposed a more robust model trained on unformed poetic text, whose output form is constrained at sample time. This approach offers greater control over the style of the generated poetry than the earlier method, and facilitates themes and poetic devices.

An indistinguishability test, where participants were asked to classify a randomly selected set of human "nonsense verse" and machine-generated poetry, showed generated poetry to be indistinguishable from that written by humans. In addition, the poems that were deemed most 'humanlike' and most aesthetic were both machine-generated.

In future work, it would be useful to investigate models based on morphemes, rather than characters, which offers potentially superior performance for complex and rare words (Luong et al., 2013), which are common in poetry.

References

Hideaki Aoyama and John Constable. 1998. Word length frequency and distribution in english: Observations, theory, and implications for the construction of verse lines. *arXiv preprint cmp-lg/9808004* .

Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. 2012. Markov constraints for generating lyrics with style. In *Proceedings of the 20th European Conference on Artificial Intelligence*. IOS Press, pages 115–120.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, Springer, pages 137–186.

Alan W Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Proceedings of ICLR* .

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full face poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*. pages 95–102.

Eric Corlett and Gerald Penn. 2010. An exact a* method for deciphering letter-substitution ciphers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1040–1047.

Edgar Dale and Jeanne S Chall. 1948. A formula for predicting readability: Instructions. *Educational research bulletin* pages 37–54.

Pablo Gervás. 2000. Wasp: Evaluation of different strategies for the automatic generation of spanish verse. In *Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects of AI*. pages 93–100.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1183–1191.

Raghav Goyal, Marc Dymetman, and Eric Gaussier. 2016. Natural language generation through character-based rnns with finite-state prior knowledge. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan, pages 1083–1092.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 524–533.

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615* .

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL on Main conference poster sessions*. Association for Computational Linguistics, pages 499–506.

Grzegorz Kondrak. 2005. N-gram similarity and distance. In *String processing and information retrieval*. Springer, pages 115–126.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*. pages 104–113.

Hisar Manurung, Graeme Ritchie, and Henry Thompson. 2000. Towards a computational model of poetry generation. Technical report, The University of Edinburgh.

Stephen McGregor, Matthew Purver, and Geraint Wiggins. 2016. Process based evaluation of computer generated poetry. In *The INLG 2016 Workshop on Computational Creativity in Natural Language Generation*. page 51.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*. volume 2, page 3.

Joanna Misztal and Bipin Indurkhya. 2014. Poetry generation system with an emotional personality. In *Proceedings of the Fourth International Conference on Computational Creativity*.

Cory Myers, Lawrence R Rabiner, and Aaron E Rosenberg. 1980. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on* 28(6):623–635.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku: Generating haiku with word associations norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*. Association for Computational Linguistics, pages 32–39.

Sujith Ravi and Kevin Knight. 2009. Learning phoneme mappings for transliteration without parallel data. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 37–45.

Holger Schwenk and Jean-Luc Gauvain. 2005. Training neural network language models on very large corpora. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 201–208.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Jukka M Toivanen, Matti Järvisalo, Hannu Toivonen, et al. 2013. Harnessing constraint programming for poetry composition. In *Proceedings of the Fourth International Conference on Computational Creativity*. page 160.

Willie Walker, Paul Lamere, and Philip Kwok. 2010. Freetts 1.2: A speech synthesizer written entirely in the java programming language.

R Weide. 2005. The carnegie mellon pronouncing dictionary [cmudict. 0.6].

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

Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. 2016. Generating chinese classical poems with rnn encoder-decoder. *arXiv preprint arXiv:1604.01537* .

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *EMNLP*. pages 670–680.

Creating Training Corpora for NLG Micro-Planning

Claire Gardent **Anastasia Shimorina**
CNRS, LORIA, UMR 7503
Vandoeuvre-lès-Nancy, F-54500, France

Shashi Narayan **Laura Perez-Beltrachini**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, UK

{claire.gardent,anastasia.shimorina}@loria.fr {shashi.narayan,lperez}@ed.ac.uk

Abstract

In this paper, we present a novel framework for semi-automatically creating linguistically challenging micro-planning data-to-text corpora from existing Knowledge Bases. Because our method pairs data of varying size and shape with texts ranging from simple clauses to short texts, a dataset created using this framework provides a challenging benchmark for microplanning. Another feature of this framework is that it can be applied to any large scale knowledge base and can therefore be used to train and learn KB verbalisers. We apply our framework to DBpedia data and compare the resulting dataset with Wen et al. (2016)'s. We show that while Wen et al.'s dataset is more than twice larger than ours, it is less diverse both in terms of input and in terms of text. We thus propose our corpus generation framework as a novel method for creating challenging data sets from which NLG models can be learned which are capable of handling the complex interactions occurring during in micro-planning between lexicalisation, aggregation, surface realisation, referring expression generation and sentence segmentation. To encourage researchers to take up this challenge, we recently made available a dataset created using this framework in the context of the WEBNLG shared task.

1 Introduction

To train Natural Language Generation (NLG) systems, various input-text corpora have been developed which associate (numerical, formal, linguistic) input with text. As discussed in detail in Section 2, these corpora can be classified into three main types namely, (i) domain specific corpora, (ii) benchmarks constructed from "Expert" Linguistic Annotations and (iii) crowdsourced benchmarks.[1]

In this paper, we focus on how to create data-to-text corpora which can support the learning of micro-planners i.e., data-to-text generation systems that can handle the complex interactions occurring between lexicalisation (mapping data to words), aggregation (exploiting linguistic constructs such as ellipsis and coordination to avoid repetition), surface realisation (using the appropriate syntactic constructs to build sentences), sentence segmentation and referring expression generation.

We start by reviewing the main existing types of NLG benchmarks and we argue for a crowdsourcing approach in which (i) data units are automatically built from an existing Knowledge Base (KB) and (ii) text is crowdsourced from the data (Section 2). We then propose a generic framework for semi-automatically creating training corpora for NLG (Section 3) from existing knowledge bases. In Section 4, we apply this framework to DBpedia data and we compare the resulting dataset with the dataset of Wen et al. (2016) using various metrics to evaluate the linguistic and computational adequacy of both datasets. By applying these metrics, we show that while Wen et al.'s dataset is more than twice larger than ours, it is less diverse both in terms of input and in terms of text. We also com-

[1] We ignore here (Lebret et al., 2016)'s dataset which was created fully automatically from Wikipedia by associating infoboxes with text because this dataset fails to ensure an adequate match between data and text. We manually examined 50 input/output pairs randomly extracted from this dataset and did not find a single example where data and text matched. As such, this dataset is ill-suited for training micro-planners. Moreover, since its texts contain both missing and additional information, it cannot be used to train joint models for content selection and micro-planning either.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 179–188
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1017

pare the performance of a sequence-to-sequence model (Vinyals et al., 2015) on both datasets to estimate the complexity of the learning task induced by each dataset. We show that the performance of this neural model is much lower on the new data set than on the existing ones. We thus propose our corpus generation framework as a novel method for creating challenging data sets from which NLG models can be learned which are capable of generating complex texts from KB data.

2 NLG Benchmarks

Domain specific benchmarks. Several domain specific data-text corpora have been built by researchers to train and evaluate NLG systems. In the sports domain, Chen and Mooney (2008) constructed a dataset mapping soccer games events to text which consists of 1,539 data-text pairs and a vocabulary of 214 words. For weather forecast generation, the dataset of Liang et al. (2009) includes 29,528 data-text pairs with a vocabulary of 345 words. For the air travel domain, Ratnaparkhi (2000) created a dataset consisting of 5,426 data-text pairs with a richer vocabulary (927 words) and in the biology domain, the KBGen shared task (Banik et al., 2013) made available 284 data-text pairs where the data was extracted from an existing knowledge base and the text was authored by biology experts.

An important limitation of these datasets is that, because they are domain specific, systems learned from them are restricted to generating domain specific, often strongly stereotyped text (e.g., weather forecast or soccer game commentator reports). Arguably, training corpora for NLG should support the learning of more generic systems capable of handling a much wider range of linguistic interactions than is present in stereotyped texts. By nature however, domain specific corpora restrict the lexical and often the syntactic coverage of the texts to be produced and thereby indirectly limit the expressivity of the generators trained on them.

Benchmarks constructed from "expert" linguistic annotations. NLG benchmarks have also been proposed where the input data is either derived from dependency parse trees (SR'11 task, Belz et al. 2011) or constructed through manual annotation (AMR Corpus, Banarescu et al. 2012). Contrary to the domain-specific data sets just mentioned, these corpora have a wider coverage and are large enough for training systems that can generate linguistically sophisticated text.

One main drawback of these benchmarks however is that their construction required massive manual annotation of text with complex linguistic structures (parse trees for the SR task and Abstract Meaning Representation for the AMR corpus). Moreover because these structures are complex, the annotation must be done by experts. It cannot be delegated to the crowd. In short, the creation of such benchmark is costly both in terms of time and in terms of expertise.

Another drawback is that, because the input representation derived from a text is relatively close to its surface form[2], the NLG task is mostly restricted to surface realisation (mapping input to sentences). That is, these benchmarks give very limited support for learning models that can handle the interactions between micro-planning subtasks.

Crowdsourced benchmarks. More recently, data-to-text benchmarks have also been created by associating data units with text using crowdsourcing.

Wen et al. (2016) first created data by enumerating all possible combinations of 14 dialog act types (e.g., *request, inform*) and attribute-value pairs present in four small-size, hand-written ontologies about TVs, laptops, restaurants and hotels. They then use crowdsourcing to associate each data unit with a text. The resulting dataset is both large and varied (4 domains) and was successfully exploited to train neural and imitation learning data-to-text generator (Wen et al., 2016; Lampouras and Vlachos, 2016). Similarly, Novikova and Rieser (2016) described a framework for collecting data-text pairs using automatic quality control measures and evaluating how the type of the input representations (text vs pictures) impacts the quality of crowdsourced text.

The crowdsourcing approach to creating input-text corpora has several advantages.

First, it is low cost in that the data is produced automatically and the text is authored by a crowdworker. This is in stark contrast with the previous approach where expert linguists are required to align text with data.

[2]For instance, the input structures made available by the shallow track of the SR task contain all the lemmas present in the corresponding text. In this case, the generation task is limited to determining (i) the linear ordering and (ii) the full form of the word in the input.

Second, because the text is crowd-sourced from the data (rather than the other way round), there is an adequate match between text and data both semantically (the text expresses the information contained in the data) and computationally (the data is sufficiently different from the text to require the learning of complex generation operations such as sentence segmentation, aggregation and referring expression generation).

Third, by exploiting small hand-written ontologies to quickly construct meaningful artificial data, the crowdsourcing approach allows for the easy creation of a large dataset with data units of various size and bearing on different domains. This, in turn, allows for better linguistic coverage and for NLG tasks of various complexity since typically, inputs of larger size increases the need for complex microplanning operations.

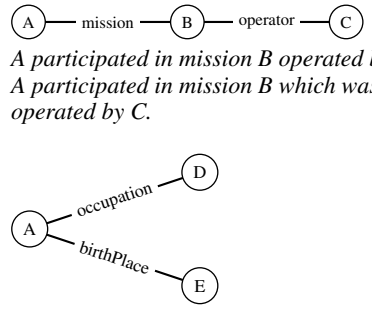
3 The WebNLG Framework for Creating Data-to-Text, Micro-Planning Benchmarks

While as just noted, the crowdsourcing approach presented by Wen et al. (2016) has several advantages, it also has a number of shortcomings.

One important drawback is that it builds on artificial rather than "real" data i.e., data that would be extracted from an existing knowledge base. As a result, the training corpora built using this method cannot be used to train KB verbalisers i.e., generation systems that can verbalise KB fragments.

Another limitation concerns the shape of the input data. Wen et al.'s data can be viewed as trees of depth one (a set of attributes-value pairs describing a single entity e.g., a restaurant or a laptop). As illustrated in Figure 1 however, there is a strong correlation between the shape of the input and the syntactic structure of the corresponding sentence. The path structure T_1 where B is shared by two predicates (*mission* and *operator*) will favour the use of a participial or a passive subject relative clause. In contrast, the branching structure T_2 will favour the use of a new clause with a pronominal subject or a coordinated VP. More generally, allowing for trees of deeper depth is necessary to indirectly promote the introduction in the benchmark of a more varied set of syntactic constructs to be learned by generators.

To address these issues, we introduce a novel method for creating data-to-text corpora from large knowledge bases such as DBPedia. Our



T_1
$S_{1.1}$ *A participated in mission B operated by C.*
$S_{1.2}$ *A participated in mission B which was operated by C.*

T_2
$S_{2.1}$ *A was born in E. She worked as an engineer.*
$S_{2.2}$ *A was born in E and worked as an engineer.*

Figure 1: Input shape and linguistic structures (A = Susan Helms, B = STS 78, C = NASA, D = engineer, E = Charlotte, North Carolina).

method combines (i) a content selection module designed to extract varied, relevant and coherent data units from DBPedia with (ii) a crowdsourcing process for associating data units with human authored texts that correctly capture their meaning. Example 1 shows a data/text unit created by our method using DBPedia as input KB.

(1) a. *(John_E_Blaha birthDate 1942_08_26)*
(John_E_Blaha birthPlace San_Antonio)
(John_E_Blaha occupation Fighter_pilot)

b. *John E Blaha, born in San Antonio on 1942-08-26, worked as a fighter pilot*

Our method has the following features.

First, it can be used to create a data-to-text corpus from any knowledge base where entities are categorised and there is a large number of entities belonging to the same category. As noted above, this means that the resulting corpus can be used to train KB verbalisers i.e., generators that are able to verbalise fragments of existing knowledge bases. It could be used for instance, to verbalise fragments of e.g., MusicBrainz[3], FOAF[4] or LinkedGeoData.[5]

Second, as crowdworkers are required to enter text that matches the data and a majority vote validation process is used to eliminate mis-matched pairs, there is a direct match between text and data. This allows for a clear focus on the non content selection part of generation known as microplanning.

Third, because data of increasing size is matched with texts ranging from simple clauses to

[3] https://musicbrainz.org/
[4] http://www.foaf-project.org/
[5] http://linkedgeodata.org/

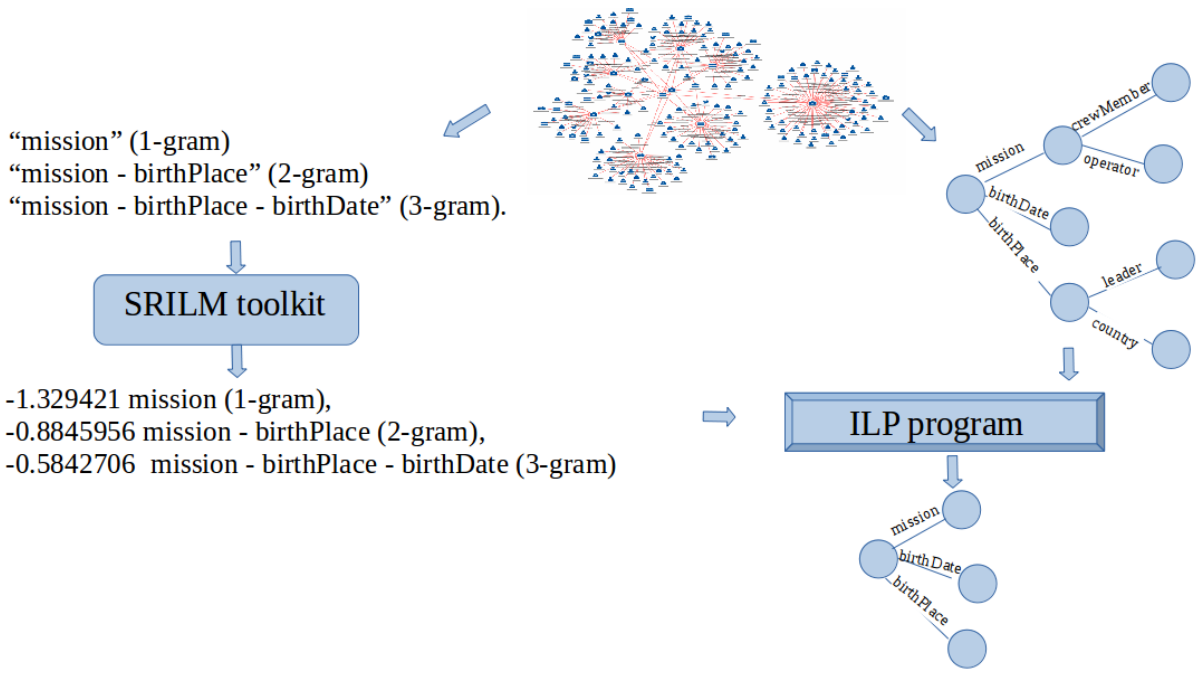


Figure 2: Extracting data units from DBPedia.

short texts consisting of several sentences, the resulting benchmark is appropriate for exercising the main subtasks of microplanning. For instance, in Example (1) above, given the input shown in (1a), generating (1b) involves lexicalising the *occupation* property as the phrase *worked as* (lexicalisation); using PP coordination (*born in San Antonio on 1942-08-26*) to avoid repeating the word *born* (aggregation); and verbalising the three triples using a single complex sentence including an apposition, a PP coordination and a transitive verb construction (sentence segmentation and surface realisation).

3.1 DBPedia

To illustrate the functioning of our benchmark creation framework, we apply it to DBPedia. DBPedia is a multilingual knowledge base that was built from various kinds of structured information contained in Wikipedia (Mendes et al., 2012). This data is stored as RDF (Resource Description Format) triples of the form *(subject, property, object)* where the subject is a URI (Uniform Resource Identifier), the property is a binary relation and the object is either a URI or a literal value such as a string, a date or a number. We use an English version of the DBPedia knowledge base which encompasses 6.2M entities, 739 classes, 1,099 properties with reference values and 1,596 properties with typed literal values.[6]

3.2 Selecting Content

To create data units, we adapted the procedure outlined by Perez-Beltrachini et al. (2016) and sketched in Figure 2. This method can be summarised as follows.

First, DBPedia *category graphs* are extracted from DBPedia by retrieving up to 500 *entity graphs* for entities of the same category.[7] For example, we build a category graph for the Astronaut category by collecting, graphs of depth five for 500 entities of types astronaut.

Next, category graphs are used to learn bi-gram models of DBPedia properties which specify the probability of two properties co-occuring together. Three types of bi-gram models are extracted from category graphs using the SRILM toolkit (Stolcke, 2002): one model (*S*-Model) for bigrams occurring in sibling triples (triples with a shared subject); one model (*C*-Model) for bigrams occurring in chained triples (the object of one triple is the subject of the other); and one model (*M*-Model) which is a linear interpolation of the sibling and the chain model. The intuition is that these sib-

[6] http://wiki.dbpedia.org/dbpedia-dataset-version-2015-10

[7] An entity graph for some entity e is a graph obtained by traversing the DBPedia graph starting in e and stopping at depth five.

ling and chain models capture different types of coherence, namely, topic-based coherence for the *S*-Model and discourse-based coherence for the *C*-Model.

Finally, the content selection task is formulated as an Integer Linear Programming (ILP) problem to select, for a given entity of category C and its entity graph G_e, subtrees of G_e with maximal bi-gram probability and varying size (between 1 and 7 RDF triples).

Category	A	B	M	U	S	W
#Inputs	663	1220	333	508	1137	1207
#I. Patterns	546	369	300	432	184	277
#Properties	38	46	30	41	32	50
#Entities	74	278	47	75	264	224

Table 1: Data statistics from content selection (A:Astronaut, B:Building, M:Monument, U:University, W:Written work, S:Sports team).

We applied this content selection procedure to the DBPedia categories Astronaut (A), Building (B), Monument (M), University (U), Sports team (S) and Written work (W), using the three bi-gram models (*S*-Model, *C*-Model, *M*-Model) and making the number of triples required by the ILP constraint to occur in the output solutions vary between 1 and 7. The results are shown in Table 1. An input is a set of triples produced by the content selection module. The number of input (#Inputs) is thus the number of distinct sets of triples produced by this module. In contrast, input patterns are inputs where subject and object have been abstracted over. That is, the number of input patterns (#I. Patterns) is the number of distinct sets of properties present in the set of inputs. The number of properties (#Properties) is the number of distinct RDF properties occurring in the dataset. Similarly, the number of entities (#Entities) is the number of distinct RDF subjects and objects occurring in each given dataset.

3.3 Associating Content with Text

We associate data with text using the Crowdflower platform.[8] We do this in four main steps as follows.

1. Clarifying properties. One difficulty when collecting texts verbalising sets of DBPedia triples is that the meaning of DBPedia properties may be unclear. We therefore first manually clarified for each category being worked on, those properties which have no obvious lexicalisations (e.g., *crew1up* was replaced by *commander*).

2. Getting verbalisations for single triples. Next, we collected three verbalisations for data units of size one, i.e. single triples consisting of a subject, a property and an object. For each such input, crowdworkers were asked to produce a sentence verbalising its content. We used both *a priori* automatic checks to prevent spamming and *a posteriori* manual checks to remove incorrect verbalisations. We also monitored crowdworkers as they entered their input and banned those who tried to circumvent our instructions and validators. The automatic checks comprise 12 custom javascript validators implemented in the CrowdFlower platform to block contributor answers which fail to meet requirements such as the minimal time a contributor should stay on page, the minimal length of the text produced, the minimal match of tokens between a triple and its verbalisation and various format restrictions used to detect invalid input. The exact match between a triple and its verbalisation was also prohibited. In addition, after data collection was completed, we manually checked each data-text pair and eliminated from the data set any pair where the text either did not match the information conveyed by the triple or was not a well-formed English sentence.

3. Getting verbalisations for input containing more than one triple. The verbalisations collected for single triples were used to construct input with bigger size. Thus, for input with a number of triples more than one, the crowd was asked to merge the sentences corresponding to each triple (obtained in step 2) into a natural sounding text. In such a way, we diminish the risk of having misinterpretations of the original semantics of a data unit. Contributors were also encouraged to change the order, and the wording of sentences, while writing their texts. For each data unit, we collected three verbalisations.

4. Verifying the quality of the collected texts. The verbalisations obtained in Step 3 were verified through crowdsourcing. Each verbalisation collected in Step 3 was displayed to CrowdFlower contributors together with the corresponding set of triples. Then the crowd was asked to assess its fluency, semantic adequacy, and grammaticality. Those criteria were checked by asking the follow-

[8] http://www.crowdflower.com

# Triples	1	2	3	4	5	6	7
# Tokens	4/30/10.48	11/45/22.97	7/37/16.96	17/60/36.38	14/53/29.61	29/80/49.14	24/73/42.95
# Sentences	1/2/1.00	1/4/1.23	1/3/1.02	1/5/2.05	1/4/1.64	1/6/2.85	1/5/2.42

Table 2: Text statistics from crowdsourcing for triple sets of varying sizes (min/max/avg).

ing three questions:

Does the text sound fluent and natural?
Does the text contain all and only the information from the data?
Is the text good English (no spelling or grammatical mistakes)?

We collected five answers per verbalisation. A verbalisation was considered bad, if it received three negative answers in at least one criterion. After the verification step, the total corpus loss was of 8.7%. An example of rejected verbalisation can be found in Example (2). The verbalisation was dropped due to the lack of fluency (awkward lexicalisation of the property *club*).

(2) *(AEK_Athens_F.C. manager Gus_Poyet)*
(Gus_Poyet club Chelsea_F.C.)
AEK Athens F.C. are managed by Gus Poyet, who is in Chelsea_F.C.

Table 2 shows some statistics about the texts obtained using our crowdsourcing procedure for triple sets of size one to seven.

4 Comparing Benchmarks

We now compare a dataset created using our dataset creation framework (henceforth WEBNLG) with the dataset of Wen et al. (2016)[9] (henceforth, RNNLG). Example 3 shows a sample data-text pair taken from the RNNLG dataset.

(3) Dialog Moves
recommend(name=caerus 33;type=television;
screensizerange=medium;family=t5;hasusbport=true)
The caerus 33 is a medium television in the T5 family that's USB-enabled.

As should be clear from the discussion in Section 2 and 3, both datasets are similar in that, in both cases, data is built from ontological information and text is crowdsourced from the data. An important difference between the two datasets is that, while the RNNLG data was constructed by enumerating possible combinations of dialog act types and attribute-value pairs, the WEBNLG data is created using a sophisticated content selection procedure geared at producing sets of data units that are relevant for a given ontological category and that are varied in terms of size, shape and content. We now investigate the impact of this difference on the two datasets (WEBNLG and RNNLG). To assess the degree to which both datasets support the generation of linguistically varied text requiring complex micro-planning operations, we examine a number of data and text related metrics. We also compare the results of an out-of-the-box sequence-to-sequence model as a way to estimate the complexity of the learning task induced by each dataset.

	WEBNLG	RNNLG
Nb. Input	5068	22225
Nb. Data-Text Pairs	13339	30842
Nb. Domains	6	4
Nb. Attributes	172	108
Nb. Input Patterns	2108	2155
Nb. Input / Nb Input Pattern	2.40	10.31
Nb. Input Shapes	58	6

Table 3: Comparing WEBNLG and RNNLG datasets. Attributes are properties in RDF triples or slots in dialog acts.

4.1 Data Comparison

Terminology. The attributes in the RNNLG dataset can be viewed as binary relations between the object talked about (a restaurant, a laptop, a TV or a hotel) and a value. Similarly, in the WEBNLGdataset, DBpedia RDF properties relate a subject entity to an object which can be either an entity or a datatype value. In what follows, we refer to both as *attributes*.

Table 3 shows several statistics which indicate that, while the RNNLG dataset is larger than WEBNLG, WEBNLG is much more diverse in terms of attributes, input patterns and input shapes.

Number of attributes. As illustrated in Example (4) below, different attributes can be lexicalised using different parts of speech. A dataset with a larger number of attributes is therefore more likely to induce texts with greater syntactic variety.

(4) Verb: *X title Y* / *X served as Y*
Relational noun: *X nationality Y* / *X's nationality is Y*
Preposition: *X country Y* / *X is in Y*
Adjective: *X nationality USA* / *X is American*

[9] https://github.com/shawnwun/RNNLG

As shown in Table 3, WEBNLG has a more diverse attribute set than RNNLG both in absolute (172 attributes in WEBNLG against 108 in RNNLG) and in relative terms (RNNLG is a little more than twice as large as WEBNLG).

Number of input patterns. Since attributes may give rise to lexicalisation with different parts of speech, the sets of attributes present in an input (input pattern)[10] indirectly determine the syntactic realisation of the corresponding text. Hence a higher number of input patterns will favour a higher number of syntactic realisations. This is exemplified in Example (5) where two inputs with the same number of attributes give rise to texts with different syntactic forms. While in Example (5a), the attribute set {*country, location, startDate*} is realised by a passive (*is located*), an apposition (*Australia*) and a deverbal nominal (*its construction*), in Example (5b), the attribute set {*almaMater, birthPlace, selection*} induced a passive (*was born*) and two VP coordinations (*graduated and joined*).

(5) a. *('108_St_Georges_Terrace location Perth', 'Perth country Australia', '108_St_Georges_Terrace startDate 1981')*
country, location, startDate
108 St. Georges Terrace is located in Perth, Australia. Its construction began in 1981.
passive, apposition, deverbal nominal

b. *('William_Anders selection 1963', 'William_Anders birthPlace British_Hong_Kong', 'William_Anders almaMater "AFIT, M.S. 1962"')*
almaMater, birthPlace, selection
William Anders was born in British Hong Kong, graduated from AFIT in 1962, and joined NASA in 1963.
passive, VP coordination, VP coordination

Again, despite the much larger size of the RNNLG dataset, the number of input patterns in both datasets is almost the same. That is, the relative variety in input patterns is higher in WEBNLG.

Number of input / Number of input patterns. The ratio between number of inputs and the number of input patterns has an important impact both in terms of linguistic diversity and in terms of learning complexity. A large ratio indicates a "repetitive dataset" where the same pattern is instantiated a high number of times. While this facilitates learning, this also reduces linguistic coverage (less combinations of structures can be learned) and may induce over-fitting. Note that because datasets are typically delexicalised when training NLG models (cf. e.g., Wen et al. 2015 and Lampouras and Vlachos 2016), at training time, different instantiations of the same input pattern reduce to identical input.

The two datasets markedly differ on this ratio which is five times lower in WEBNLG. While in WEBNLG, the same pattern is instantiated in average 2.40 times, it is instantiated 10.31 times in average in RNNLG. From a learning perspective, this means that the RNNLG dataset facilitates learning but also makes it harder to assess how well systems trained on it can generalise to handle unseen input.

Input shape. As mentioned in Section 3, in the RNNLG dataset, all inputs can be viewed as trees of depth one while in the WEBNLG dataset, input may have various shapes. As a result, RNNLG texts will be restricted to syntactic forms which permit expressing such multiple predications of the same entity e.g., subject relative clause, VP and sentence coordination etc. In contrast, the trees extracted by the WEBNLG content selection procedure may be of depth five and therefore allow for further syntactic constructs such as object relative clause and passive participles (cf. Figure 1).

We can show this empirically as well that WEBNLG is far more diverse than RNNLG in terms of input shapes. The RNNLG dataset has only 6 distinct shapes and all of them are of depth 1, i.e., all (attribute, value) pairs in an input are siblings to each other. In contrast, the WEBNLG dataset has 58 distinct shapes, out of which only 7 shapes are with depth 1, all others have depth more than 1 and they cover 49.6% of all inputs.

4.2 Text Comparison

Table 4 gives some statistics about the texts contained in each dataset.

(6) *(Alan_Bean birthDate "1932-03-15")*
Alan Bean was born on March 15, 1932.

(7) *('Alan_Bean nationality United_States', 'Alan_Bean birthDate "1932-03-15"', 'Alan_Bean almaMater "UT Austin, B.S. 1955"', 'Alan_Bean birthPlace Wheeler,_Texas', 'Alan_Bean selection 1963')*
Alan Bean was an American astronaut, born on March 15, 1932 in Wheeler, Texas. He received a Bachelor of Science degree at the University of Texas at Austin in 1955 and was chosen by NASA in 1963.

[10] Recall from section 3 that input patterns are inputs where subjects and objects have been remove thus, in essence, an input pattern is the set of all the attributes occurring in a given input.

As illustrated by the contrast between Examples (6) and (7) above, text length (number of tokens per text) and the number of sentences per text are strong indicators of the complexity of the generation task. We use the Stanford Part-Of-Speech Tagger and Parser version 3.5.2 (dated 2015-04-20, Manning et al. 2014) to tokenize and to perform sentence segmentation on text. As shown in Table 4, WEBNLG's texts are longer both in terms of tokens and in terms of number of sentences per text. Another difference between the two datasets is that WEBNLG contains a higher number of text per input thereby providing a better basis for learning paraphrases.

	WEBNLG	RNNLG
Nb. Text / Input	2.63	1.38
Text Length (avg/median/min/max)	24.36/23/4/80	18.37/19/1/76
Nb. Sentence / Text (avg/median/min/max)	1.45/1/1/6	1.25/1/1/6
Nb. Tokens	290479	531871
Nb. Types	2992	3524
Lexical Sophistication	0.69	0.54
CTTR	3.93	3.42

Table 4: Text statistics from WEBNLG and RNNLG.

The size and the content of the vocabulary is another important factor in ensuring the learning of wide coverage generators. While a large vocabulary makes the learning problem harder, it also allows for larger coverage. WEBNLG exhibits a higher corrected type-token ratio (CTTR), which indicates greater lexical variety, and higher lexical sophistication (LS). Lexical sophistication measures the proportion of relatively unusual or advanced word types in the text. In practice, LS is the proportion of lexical word types (lemma) which are not in the list of 2,000 most frequent words generated from the British National Corpus[11]. Type-token ratio (TTR) is a measure of diversity defined as the ratio of the number of word types to the number of words in a text. To address the fact that this ratio tends to decrease with the size of the corpus, corrected TTR can be used to control for corpus size. It is defined as $T/\sqrt{2N}$, where T is the number of types and N the number of tokens.

Overall, the results shown in Table 4 indicate that WEBNLG texts are both lexically more diverse (higher corrected type/token ratio) and more sophisticated (higher proportion of unfrequent words) than RNNLG's. They also show a proportionately larger vocabulary for WEBNLG (2,992 types for 290,479 tokens in WEBNLG against 3,524 types for 531,871 tokens in RNNLG).

4.3 Neural Generation

Richer and more varied datasets are harder to learn from. As a proof-of-concept study of the comparative difficulty of the two datasets with respect to machine learning, we compare the performance of a sequence-to-sequence model for generation on both datasets.

We use the multi-layered sequence-to-sequence model with attention mechanism described in (Vinyals et al., 2015).[12] The model was trained with 3-layer LSTMs with 512 units each with a batch size of 64 and a learning rate of 0.5.

To allow for a fair comparison, we use a similar amount of data (13K data-text pairs) for both datasets. As RNNLG is bigger in size than WEBNLG, we constructed a balanced sample of RNNLG which included equal number of instances per category (*tv, laptop*, etc). We use a 3:1:1 ratio for training, developement and testing. The training was done in two delexicalisation modes: *fully* and *name only*. In case of fully delexicalisation, all entities were replaced by their generic terms, whereas in *name only* mode only subjects were modified in that way. For instance, the triple (*FC Köln manager Peter Stöger*) was delexicalised as (*SportsTeam manager Manager*) in the first mode, and as (*SportsTeam manager Peter Stöger*) in the second mode. The delexicalisation in sentences was done using the exact match between entities and tokens. For training, we use all the available vocabulary. Input and output vocabulary sizes are reported in Table 5.

Table 5 shows the perplexity results. In both modes, RNNLG yielded lower scores than WEBNLG. This is inline with the observations made above concerning the higher data diversity, larger vocabulary and more complex texts of

[11] We compute LS and CTTR using the Lexical Complexity Analyzer developed by Lu (2012).

[12] We used the TensorFlow code available at https://github.com/tensorflow/models/tree/master/tutorials/rnn/translate. Alternatively, we could have used the implementation of Wen et al. (2016) which is optimised for generation. However the code is geared toward dialog acts and modifying it to handle RDF triples is non trivial. Since the comparison aims at examining the relative performance of the same neural network on the two datasets, we used the tensor flow implementation instead.

WEBNLG. Similary, the BLEU score of the generated sentences (Papineni et al., 2002) is lower for WEBNLG suggesting again a dataset that is more complex and therefore more difficult to learn from.

	Delexicalisation Mode	WEBNLG	RNNLG
Vocab size	Fully	520, 2430	140, 1530
	Name only	1130, 2940	570, 1680
Perplexity	Fully	27.41	17.42
	Name only	25.39	23.93
BLEU	Fully	0.19	0.26
	Name only	0.10	0.27

Table 5: Vocabulary sizes of input, output (number of tokens). Perplexity and BLEU scores.

5 Conclusion

We presented a framework for building NLG data-to-text training corpora from existing knowledge bases.

One feature of our framework is that datasets created using this framework can be used for training and testing KB verbalisers an in particular, verbalisers for RDF knowledge bases. Following the development of the semantic web, many large scale datasets are encoded in the RDF language (e.g., MusicBrainz, FOAF, LinkedGeoData) and official institutions[13] increasingly publish their data in this format. In this context, our framework is useful both for creating training data from RDF KB verbalisers and to increase the number of datasets available for training and testing NLG.

Another important feature of our framework is that it permits creating semantically and linguistically diverse datasets which should support the learning of lexically and syntactically, wide coverage micro-planners. We applied our framework to DBpedia data and showed that although twice smaller than the largest corpora currently available for training data-to-text microplanners, the resulting dataset is more semantically and linguistically diverse. Despite the disparity in size, the number of attributes is comparable in the two datasets. The ratio between input and input patterns is five times lower in our dataset thereby making learning harder but also diminishing the risk of overfitting and providing for wider linguistic coverage. Conversely, the ratio of text per input is twice higher thereby providing better support for learning paraphrases.

[13]See http://museum-api.pbworks.com for examples.

We have recently released a first version of the WebNLG dataset in the context of a shared task on micro-planning[14]. This new dataset consists of 21,855 data/text pairs with a total of 8,372 distinct data input. The input describes entities belonging to 9 distinct DBpedia categories namely, Astronaut, University, Monument, Building, ComicsCharacter, Food, Airport, SportsTeam and WrittenWork. The WebNLG data is licensed under the following license: CC Attribution-Noncommercial-Share Alike 4.0 International and can be downloaded at http://talc1.loria.fr/webnlg/stories/challenge.html.

Recently, several sequence-to-sequence models have been proposed for generation. Our experiments suggest that these are not optimal when it comes to generate linguistically complex texts from rich data. More generally, they indicate that the data-to-text corpora built by our framework are challenging for such models. We hope that the WEBNLG dataset which we have made available for the WEBNLG shared task will drive the deep learning community to take up this new challenge and work on the development of neural generators that can handle the generation of KB verbalisers and of linguistically rich texts.

Acknowledgments

The research presented in this paper was partially supported by the French National Research Agency within the framework of the WebNLG Project (ANR-14-CE24-0033). The third author is supported by the H2020 project SUMMA (under grant agreement 688139).

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (AMR) 1.0 specification. In *Proceedings of EMNLP*.

Eva Banik, Claire Gardent, and Eric Kow. 2013. The KBGen challenge. In *Proceedings of ENLG*.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The

[14]The test data for the WEBNLG challenge will be released on August 18th, 2017 and preliminary results will be presented and discussed at INLG 2017, https://eventos.citius.usc.es/inlg2017/index.

first surface realisation shared task: Overview and evaluation results. In *Proceedings of ENLG*.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of ICML*.

Gerasimos Lampouras and Andreas Vlachos. 2016. Imitation learning for language generation from unaligned data. In *Proceedings of COLING*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *Proceedings of EMNLP*.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning Semantic Correspondences with Less Supervision. In *Proceedings of ACL-IJCNLP*.

Xiaofei Lu. 2012. The relationship of lexical richness to the quality of ESL learners' oral narratives. *The Modern Language Journal* 96(2):190–208.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL:System Demonstrations*.

Pablo N Mendes, Max Jakob, and Christian Bizer. 2012. DBpedia: A Multilingual Cross-domain Knowledge Base. In *Proceedings of LREC*.

Jekaterina Novikova and Verena Rieser. 2016. The aNALoGuE challenge: Non aligned language generation. In *Proceedings of INLG*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Laura Perez-Beltrachini, Rania Mohamed Sayed, and Claire Gardent. 2016. Building RDF content for Data-to-Text generation. In *Proceedings of COLING*.

Adwait Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proceedings of NAACL*.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proceedings of ICSLP*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of NIPS*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of NAACL-HLT*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*.

Gated Self-Matching Networks for Reading Comprehension and Question Answering

Wenhui Wang[†♮§*] **Nan Yang**[‡§] **Furu Wei**[‡] **Baobao Chang**[†♮] **Ming Zhou**[‡]
[†]Key Laboratory of Computational Linguistics, Peking University, MOE, China
[‡]Microsoft Research, Beijing, China
[♮]Collaborative Innovation Center for Language Ability, Xuzhou, 221009, China
{wangwenhui,chbb}@pku.edu.cn
{nanya,fuwei,mingzhou}@microsoft.com

Abstract

In this paper, we present the gated self-matching networks for reading comprehension style question answering, which aims to answer questions from a given passage. We first match the question and passage with gated attention-based recurrent networks to obtain the question-aware passage representation. Then we propose a self-matching attention mechanism to refine the representation by matching the passage against itself, which effectively encodes information from the whole passage. We finally employ the pointer networks to locate the positions of answers from the passages. We conduct extensive experiments on the SQuAD dataset. The single model achieves 71.3% on the evaluation metrics of exact match on the hidden test set, while the ensemble model further boosts the results to 75.9%. At the time of submission of the paper, our model holds the first place on the SQuAD leaderboard for both single and ensemble model.

1 Introduction

In this paper, we focus on reading comprehension style question answering which aims to answer questions given a passage or document. We specifically focus on the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), a large-scale dataset for reading comprehension and question answering which is manually created through crowdsourcing. SQuAD constrains answers to the space of all possible spans within the reference passage, which is different from cloze-style reading comprehension datasets (Hermann et al., 2015; Hill et al., 2016) in which answers are single words or entities. Moreover, SQuAD requires different forms of logical reasoning to infer the answer (Rajpurkar et al., 2016).

Rapid progress has been made since the release of the SQuAD dataset. Wang and Jiang (2016b) build question-aware passage representation with match-LSTM (Wang and Jiang, 2016a), and predict answer boundaries in the passage with pointer networks (Vinyals et al., 2015). Seo et al. (2016) introduce bi-directional attention flow networks to model question-passage pairs at multiple levels of granularity. Xiong et al. (2016) propose dynamic co-attention networks which attend the question and passage simultaneously and iteratively refine answer predictions. Lee et al. (2016) and Yu et al. (2016) predict answers by ranking continuous text spans within passages.

Inspired by Wang and Jiang (2016b), we introduce a gated self-matching network, illustrated in Figure 1, an end-to-end neural network model for reading comprehension and question answering. Our model consists of four parts: 1) the recurrent network encoder to build representation for questions and passages separately, 2) the gated matching layer to match the question and passage, 3) the self-matching layer to aggregate information from the whole passage, and 4) the pointer-network based answer boundary prediction layer. The key contributions of this work are three-fold.

First, we propose a gated attention-based recurrent network, which adds an additional gate to the attention-based recurrent networks (Bahdanau et al., 2014; Rocktäschel et al., 2015; Wang and Jiang, 2016a), to account for the fact that words in the passage are of different importance to answer a particular question for reading comprehension and question answering. In Wang and Jiang (2016a), words in a passage with their corresponding attention-weighted question context are en-

[*]Contribution during internship at Microsoft Research.
[§]Equal contribution.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 189–198
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1018

coded together to produce question-aware passage representation. By introducing a gating mechanism, our gated attention-based recurrent network assigns different levels of importance to passage parts depending on their relevance to the question, masking out irrelevant passage parts and emphasizing the important ones.

Second, we introduce a self-matching mechanism, which can effectively aggregate evidence from the whole passage to infer the answer. Through a gated matching layer, the resulting question-aware passage representation effectively encodes question information for each passage word. However, recurrent networks can only memorize limited passage context in practice despite its theoretical capability. One answer candidate is often unaware of the clues in other parts of the passage. To address this problem, we propose a self-matching layer to dynamically refine passage representation with information from the whole passage. Based on question-aware passage representation, we employ gated attention-based recurrent networks on passage against passage itself, aggregating evidence relevant to the current passage word from every word in the passage. A gated attention-based recurrent network layer and self-matching layer dynamically enrich each passage representation with information aggregated from both question and passage, enabling subsequent network to better predict answers.

Lastly, the proposed method yields state-of-the-art results against strong baselines. Our single model achieves 71.3% exact match accuracy on the hidden SQuAD test set, while the ensemble model further boosts the result to 75.9%. At the time[1] of submission of this paper, our model holds the first place on the SQuAD leader board.

2 Task Description

For reading comprehension style question answering, a passage **P** and question **Q** are given, our task is to predict an answer **A** to question **Q** based on information found in **P**. The SQuAD dataset further constrains answer **A** to be a continuous subspan of passage **P**. Answer **A** often includes non-entities and can be much longer phrases. This setup challenges us to understand and reason about both the question and passage in order to infer the answer. Table 1 shows a simple example from the SQuAD dataset.

[1] On Feb. 6, 2017

Passage: Tesla later approached Morgan to ask for more funds to build a more powerful transmitter. **When asked where all the money had gone, Tesla responded by saying that he was affected by the Panic of 1901**, which he (Morgan) had caused. Morgan was shocked by the reminder of his part in the stock market crash and by Tesla's breach of contract by asking for more funds. Tesla wrote another plea to Morgan, but it was also fruitless. Morgan still owed Tesla money on the original agreement, and Tesla had been facing foreclosure even before construction of the tower began.
Question: On what did Tesla blame for the loss of the initial money?
Answer: Panic of 1901

Table 1: An example from the SQuAD dataset.

3 Gated Self-Matching Networks

Figure 1 gives an overview of the gated self-matching networks. First, the question and passage are processed by a bi-directional recurrent network (Mikolov et al., 2010) separately. We then match the question and passage with gated attention-based recurrent networks, obtaining question-aware representation for the passage. On top of that, we apply self-matching attention to aggregate evidence from the whole passage and refine the passage representation, which is then fed into the output layer to predict the boundary of the answer span.

3.1 Question and Passage Encoder

Consider a question Q = $\{w_t^Q\}_{t=1}^m$ and a passage P = $\{w_t^P\}_{t=1}^n$. We first convert the words to their respective word-level embeddings ($\{e_t^Q\}_{t=1}^m$ and $\{e_t^P\}_{t=1}^n$) and character-level embeddings ($\{c_t^Q\}_{t=1}^m$ and $\{c_t^P\}_{t=1}^n$). The character-level embeddings are generated by taking the final hidden states of a bi-directional recurrent neural network (RNN) applied to embeddings of characters in the token. Such character-level embeddings have been shown to be helpful to deal with out-of-vocab (OOV) tokens. We then use a bi-directional RNN to produce new representation $u_1^Q, \ldots, u_m^Q$ and $u_1^P, \ldots, u_n^P$ of all words in the question and passage respectively:

$$u_t^Q = \text{BiRNN}_Q(u_{t-1}^Q, [e_t^Q, c_t^Q]) \quad (1)$$

$$u_t^P = \text{BiRNN}_P(u_{t-1}^P, [e_t^P, c_t^P]) \quad (2)$$

We choose to use Gated Recurrent Unit (GRU) (Cho et al., 2014) in our experiment since it performs similarly to LSTM (Hochreiter and Schmidhuber, 1997) but is computationally cheaper.

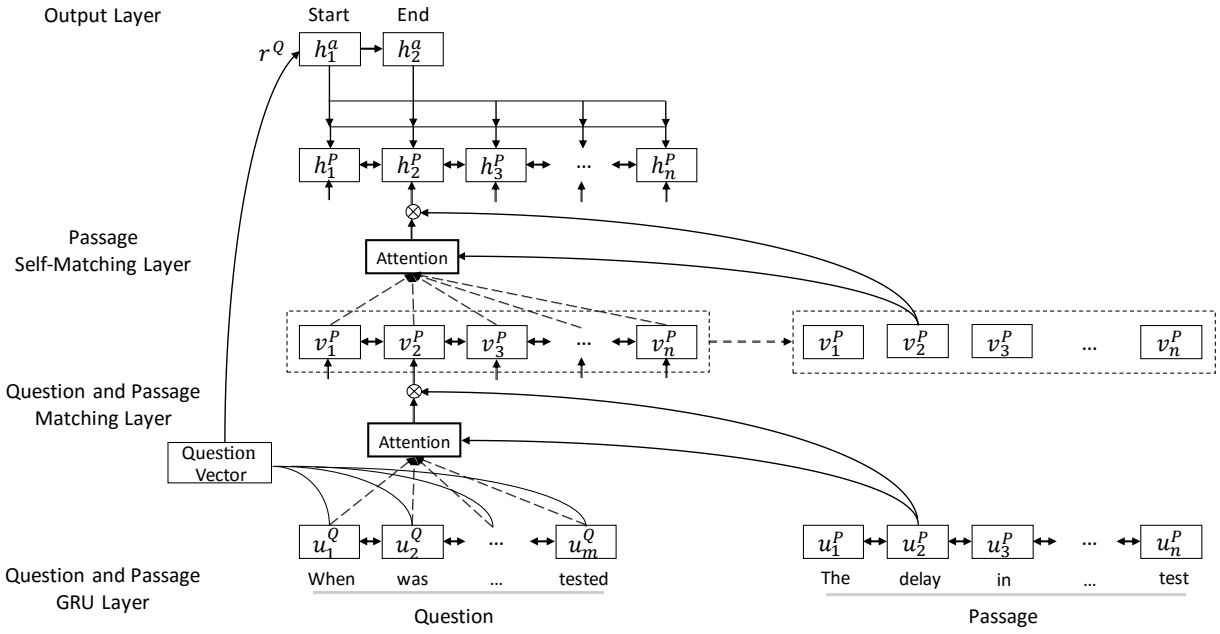


Figure 1: Gated Self-Matching Networks structure overview.

3.2 Gated Attention-based Recurrent Networks

We propose a gated attention-based recurrent network to incorporate question information into passage representation. It is a variant of attention-based recurrent networks, with an additional gate to determine the importance of information in the passage regarding a question. Given question and passage representation $\{u_t^Q\}_{t=1}^m$ and $\{u_t^P\}_{t=1}^n$, Rocktäschel et al. (2015) propose generating sentence-pair representation $\{v_t^P\}_{t=1}^n$ via soft-alignment of words in the question and passage as follows:

$$v_t^P = \text{RNN}(v_{t-1}^P, c_t) \tag{3}$$

where $c_t = att(u^Q, [u_t^P, v_{t-1}^P])$ is an attention-pooling vector of the whole question (u^Q):

$$\begin{aligned} s_j^t &= \text{v}^{\text{T}}\tanh(W_u^Q u_j^Q + W_u^P u_t^P + W_v^P v_{t-1}^P) \\ a_i^t &= \exp(s_i^t)/\Sigma_{j=1}^m \exp(s_j^t) \\ c_t &= \Sigma_{i=1}^m a_i^t u_i^Q \end{aligned} \tag{4}$$

Each passage representation v_t^P dynamically incorporates aggregated matching information from the whole question.

Wang and Jiang (2016a) introduce match-LSTM, which takes u_t^P as an additional input into the recurrent network:

$$v_t^P = \text{RNN}(v_{t-1}^P, [u_t^P, c_t]) \tag{5}$$

To determine the importance of passage parts and attend to the ones relevant to the question, we add another gate to the input ($[u_t^P, c_t]$) of RNN:

$$\begin{aligned} g_t &= \text{sigmoid}(W_g[u_t^P, c_t]) \\ [u_t^P, c_t]^* &= g_t \odot [u_t^P, c_t] \end{aligned} \tag{6}$$

Different from the gates in LSTM or GRU, the additional gate is based on the current passage word and its attention-pooling vector of the question, which focuses on the relation between the question and current passage word. The gate effectively model the phenomenon that only parts of the passage are relevant to the question in reading comprehension and question answering. $[u_t^P, c_t]^*$ is utilized in subsequent calculations instead of $[u_t^P, c_t]$. We call this gated attention-based recurrent networks. It can be applied to variants of RNN, such as GRU and LSTM. We also conduct experiments to show the effectiveness of the additional gate on both GRU and LSTM.

3.3 Self-Matching Attention

Through gated attention-based recurrent networks, question-aware passage representation $\{v_t^P\}_{t=1}^n$ is generated to pinpoint important parts in the passage. One problem with such representation is that it has very limited knowledge of context. One answer candidate is often oblivious to important

cues in the passage outside its surrounding window. Moreover, there exists some sort of lexical or syntactic divergence between the question and passage in the majority of SQuAD dataset (Rajpurkar et al., 2016). Passage context is necessary to infer the answer. To address this problem, we propose directly matching the question-aware passage representation against itself. It dynamically collects evidence from the whole passage for words in passage and encodes the evidence relevant to the current passage word and its matching question information into the passage representation h_t^P:

$$h_t^P = \text{BiRNN}(h_{t-1}^P, [v_t^P, c_t]) \tag{7}$$

where $c_t = att(v^P, v_t^P)$ is an attention-pooling vector of the whole passage (v^P):

$$\begin{aligned} s_j^t &= \text{v}^{\text{T}} \tanh(W_v^P v_j^P + W_v^{\tilde{P}} v_t^P) \\ a_i^t &= \exp(s_i^t) / \Sigma_{j=1}^n \exp(s_j^t) \\ c_t &= \Sigma_{i=1}^n a_i^t v_i^P \end{aligned} \tag{8}$$

An additional gate as in gated attention-based recurrent networks is applied to $[v_t^P, c_t]$ to adaptively control the input of RNN.

Self-matching extracts evidence from the whole passage according to the current passage word and question information.

3.4 Output Layer

We follow Wang and Jiang (2016b) and use pointer networks (Vinyals et al., 2015) to predict the start and end position of the answer. In addition, we use an attention-pooling over the question representation to generate the initial hidden vector for the pointer network. Given the passage representation $\{h_t^P\}_{t=1}^n$, the attention mechanism is utilized as a pointer to select the start position (p^1) and end position (p^2) from the passage, which can be formulated as follows:

$$\begin{aligned} s_j^t &= \text{v}^{\text{T}} \tanh(W_h^P h_j^P + W_{\text{h}}^a h_{t-1}^a) \\ a_i^t &= \exp(s_i^t) / \Sigma_{j=1}^n \exp(s_j^t) \\ p^t &= \arg\max(a_1^t, \ldots, a_n^t) \end{aligned} \tag{9}$$

Here h_{t-1}^a represents the last hidden state of the answer recurrent network (pointer network). The input of the answer recurrent network is the attention-pooling vector based on current predicted probability a^t:

$$\begin{aligned} c_t &= \Sigma_{i=1}^n a_i^t h_i^P \\ h_t^a &= \text{RNN}(h_{t-1}^a, c_t) \end{aligned} \tag{10}$$

When predicting the start position, h_{t-1}^a represents the initial hidden state of the answer recurrent network. We utilize the question vector r^Q as the initial state of the answer recurrent network. $r^Q = att(u^Q, V_r^Q)$ is an attention-pooling vector of the question based on the parameter V_r^Q:

$$\begin{aligned} s_j &= \text{v}^{\text{T}} \tanh(W_u^Q u_j^Q + W_{\text{v}}^Q V_r^Q) \\ a_i &= \exp(s_i) / \Sigma_{j=1}^m \exp(s_j) \\ r^Q &= \Sigma_{i=1}^m a_i u_i^Q \end{aligned} \tag{11}$$

To train the network, we minimize the sum of the negative log probabilities of the ground truth start and end position by the predicted distributions.

4 Experiment

4.1 Implementation Details

We specially focus on the SQuAD dataset to train and evaluate our model, which has garnered a huge attention over the past few months. SQuAD is composed of 100,000+ questions posed by crowd workers on 536 Wikipedia articles. The dataset is randomly partitioned into a training set (80%), a development set (10%), and a test set (10%). The answer to every question is a segment of the corresponding passage.

We use the tokenizer from Stanford CoreNLP (Manning et al., 2014) to preprocess each passage and question. The Gated Recurrent Unit (Cho et al., 2014) variant of LSTM is used throughout our model. For word embedding, we use pre-trained case-sensitive GloVe embeddings[2] (Pennington et al., 2014) for both questions and passages, and it is fixed during training; We use zero vectors to represent all out-of-vocab words. We utilize 1 layer of bi-directional GRU to compute character-level embeddings and 3 layers of bi-directional GRU to encode questions and passages, the gated attention-based recurrent network for question and passage matching is also encoded bidirectionally in our experiment. The hidden vector length is set to 75 for all layers. The hidden size used to compute attention scores is also 75. We also apply dropout (Srivastava et al., 2014) between layers with a dropout rate of 0.2. The model is optimized with AdaDelta (Zeiler, 2012) with an initial learning rate of 1. The ρ and ϵ used in AdaDelta are 0.95 and $1e^{-6}$ respectively.

[2]Downloaded from http://nlp.stanford.edu/data/glove.840B.300d.zip.

	Dev Set	Test Set
Single model	**EM / F1**	**EM / F1**
LR Baseline (Rajpurkar et al., 2016)	40.0 / 51.0	40.4 / 51.0
Dynamic Chunk Reader (Yu et al., 2016)	62.5 / 71.2	62.5 / 71.0
Match-LSTM with Ans-Ptr (Wang and Jiang, 2016b)	64.1 / 73.9	64.7 / 73.7
Dynamic Coattention Networks (Xiong et al., 2016)	65.4 / 75.6	66.2 / 75.9
RaSoR (Lee et al., 2016)	66.4 / 74.9	- / -
BiDAF (Seo et al., 2016)	68.0 / 77.3	68.0 / 77.3
jNet (Zhang et al., 2017)	- / -	68.7 / 77.4
Multi-Perspective Matching (Wang et al., 2016)	- / -	68.9 / 77.8
FastQA (Weissenborn et al., 2017)	- / -	68.4 / 77.1
FastQAExt (Weissenborn et al., 2017)	- / -	70.8 / 78.9
R-NET	**71.1 / 79.5**	**71.3 / 79.7**
Ensemble model		
Fine-Grained Gating (Yang et al., 2016)	62.4 / 73.4	62.5 / 73.3
Match-LSTM with Ans-Ptr (Wang and Jiang, 2016b)	67.6 / 76.8	67.9 / 77.0
RaSoR (Lee et al., 2016)	68.2 / 76.7	- / -
Dynamic Coattention Networks (Xiong et al., 2016)	70.3 / 79.4	71.6 / 80.4
BiDAF (Seo et al., 2016)	73.3 / 81.1	73.3 / 81.1
Multi-Perspective Matching (Wang et al., 2016)	- / -	73.8 / 81.3
R-NET	**75.6 / 82.8**	**75.9 / 82.9**
Human Performance (Rajpurkar et al., 2016)	80.3 / 90.5	77.0 / 86.8

Table 2: The performance of our gated self-matching networks (R-NET) and competing approaches[4].

Single Model	EM / F1
Gated Self-Matching (GRU)	**71.1 / 79.5**
-Character embedding	69.6 / 78.6
-Gating	67.9 / 77.1
-Self-Matching	67.6 / 76.7
-Gating, -Self-Matching	65.4 / 74.7

Table 3: Ablation tests of single model on the SQuAD dev set. All the components significantly (t-test, $p < 0.05$) improve the model.

Single Model	EM / F1
Base model (GRU)	64.5 / 74.1
+Gating	**66.2 / 75.8**
Base model (LSTM)	64.2 / 73.9
+Gating	**66.0 / 75.6**

Table 4: Effectiveness of gated attention-based recurrent networks for both GRU and LSTM.

4.2 Main Results

Two metrics are utilized to evaluate model performance: Exact Match (EM) and F1 score. EM measures the percentage of the prediction that matches one of the ground truth answers exactly. F1 measures the overlap between the prediction and ground truth answers which takes the maximum F1 over all of the ground truth answers. The scores on dev set are evaluated by the official script[3]. Since the test set is hidden, we are required to submit the model to Stanford NLP group to obtain the test scores.

Table 2 shows exact match and F1 scores on the dev and test set of our model and competing approaches[4]. The ensemble model consists of 20 training runs with the identical architecture and hyper-parameters. At test time, we choose the answer with the highest sum of confidence scores amongst the 20 runs for each question. As we can see, our method clearly outperforms the baseline and several strong state-of-the-art systems for both single model and ensembles.

4.3 Ablation Study

We do ablation tests on the dev set to analyze the contribution of components of gated self-matching networks. As illustrated in Table 3, the gated

[3]Downloaded from http://stanford-qa.com

[4]Extracted from SQuAD leaderboard http://stanford-qa.com on Feb. 6, 2017.

Figure 2: Part of the attention matrices for self-matching. Each row is the attention weights of the whole passage for the current passage word. The darker the color is the higher the weight is. Some key evidence relevant to the question-passage tuple is more encoded into answer candidates.

attention-based recurrent network (GARNN) and self-matching attention mechanism positively contribute to the final results of gated self-matching networks. Removing self-matching results in 3.5 point EM drop, which reveals that information in the passage plays an important role. Character-level embeddings contribute towards the model's performance since it can better handle out-of-vocab or rare words. To show the effectiveness of GARNN for variant RNNs, we conduct experiments on the base model (Wang and Jiang, 2016b) of different variant RNNs. The base model match the question and passage via a variant of attention-based recurrent network (Wang and Jiang, 2016a), and employ pointer networks to predict the answer. Character-level embeddings are not utilized. As shown in Table 4, the gate introduced in question and passage matching layer is helpful for both GRU and LSTM on the SQuAD dataset.

5 Discussion

5.1 Encoding Evidence from Passage

To show the ability of the model for encoding evidence from the passage, we draw the alignment of the passage against itself in self-matching. The attention weights are shown in Figure 2, in which the darker the color is the higher the weight is. We can see that key evidence aggregated from the whole passage is more encoded into the answer candidates. For example, the answer "Egg of Columbus" pays more attention to the key information "Tesla", "device" and the lexical variation word "known" that are relevant to the question-passage tuple. The answer "world classic of epoch-making oratory" mainly focuses on the evidence "Michael Mullet", "speech" and lexical variation word "considers". For other words, the attention weights are more evenly distributed between evidence and some irrelevant parts. Self-matching do adaptively aggregate evidence for words in passage.

5.2 Result Analysis

To further analyse the model's performance, we analyse the F1 score for different question types (Figure 3(a)), different answer lengths (Figure 3(b)), different passage lengths (Figure 3(c)) and different question lengths (Figure 3(d)) of our

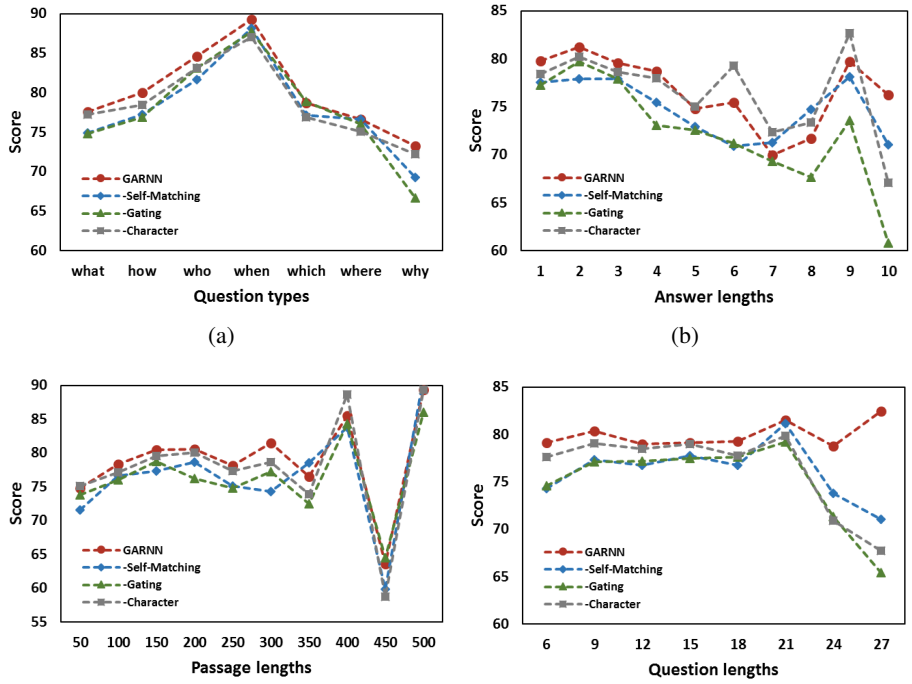


Figure 3: Model performance on different question types (a), different answer lengths (b), different passage lengths (c), different question lengths (d). The point on the x-axis of figure (c) and (d) represent the datas whose passages length or questions length are between the value of current point and last point.

model and its ablation models. As we can see, both four models show the same trend. The questions are split into different groups based on a set of question words we have defined, including "what", "how", "who", "when", "which", "where", and "why". As we can see, our model is better at "when" and "who" questions, but poorly on "why" questions. This is mainly because the answers to why questions can be very diverse, and they are not restricted to any certain type of phrases. From the Graph 3(b), the performance of our model obviously drops with the increase of answer length. Longer answers are harder to predict. From Graph 3(c) and 3(d), we discover that the performance remains stable with the increase in length, the obvious fluctuation in longer passages and questions is mainly because the proportion is too small. Our model is largely agnostic to long passages and focuses on important part of the passage.

6 Related Work

Reading Comprehension and Question Answering Dataset Benchmark datasets play an important role in recent progress in reading comprehension and question answering research. Existing datasets can be classified into two categories according to whether they are manually labeled. Those that are labeled by humans are always in high quality (Richardson et al., 2013; Berant et al., 2014; Yang et al., 2015), but are too small for training modern data-intensive models. Those that are automatically generated from natural occurring data can be very large (Hill et al., 2016; Hermann et al., 2015), which allow the training of more expressive models. However, they are in cloze style, in which the goal is to predict the missing word (often a named entity) in a passage. Moreover, Chen et al. (2016) have shown that the CNN / Daily News dataset (Hermann et al., 2015) requires less reasoning than previously thought, and conclude that performance is almost saturated.

Different from above datasets, the SQuAD provides a large and high-quality dataset. The answers in SQuAD often include non-entities and can be much longer phrase, which is more challenging than cloze-style datasets. Moreover, Rajpurkar et al. (2016) show that the dataset retains a diverse set of answers and requires different forms of logical reasoning, including multi-sentence reasoning. MS MARCO (Nguyen et al., 2016) is also a large-scale dataset. The questions in the dataset

are real anonymized queries issued through Bing or Cortana and the passages are related web pages. For each question in the dataset, several related passages are provided. However, the answers are human generated, which is different from SQuAD where answers must be a span of the passage.

End-to-end Neural Networks for Reading Comprehension Along with cloze-style datasets, several powerful deep learning models (Hermann et al., 2015; Hill et al., 2016; Chen et al., 2016; Kadlec et al., 2016; Sordoni et al., 2016; Cui et al., 2016; Trischler et al., 2016; Dhingra et al., 2016; Shen et al., 2016) have been introduced to solve this problem. Hermann et al. (2015) first introduce attention mechanism into reading comprehension. Hill et al. (2016) propose a window-based memory network for CBT dataset. Kadlec et al. (2016) introduce pointer networks with one attention step to predict the blanking out entities. Sordoni et al. (2016) propose an iterative alternating attention mechanism to better model the links between question and passage. Trischler et al. (2016) solve cloze-style question answering task by combining an attentive model with a reranking model. Dhingra et al. (2016) propose iteratively selecting important parts of the passage by a multiplying gating function with the question representation. Cui et al. (2016) propose a two-way attention mechanism to encode the passage and question mutually. Shen et al. (2016) propose iteratively inferring the answer with a dynamic number of reasoning steps and is trained with reinforcement learning.

Neural network-based models demonstrate the effectiveness on the SQuAD dataset. Wang and Jiang (2016b) combine match-LSTM and pointer networks to produce the boundary of the answer. Xiong et al. (2016) and Seo et al. (2016) employ variant coattention mechanism to match the question and passage mutually. Xiong et al. (2016) propose a dynamic pointer network to iteratively infer the answer. Yu et al. (2016) and Lee et al. (2016) solve SQuAD by ranking continuous text spans within passage. Yang et al. (2016) present a fine-grained gating mechanism to dynamically combine word-level and character-level representation and model the interaction between questions and passages. Wang et al. (2016) propose matching the context of passage with the question from multiple perspectives.

Different from the above models, we introduce self-matching attention in our model. It dynamically refines the passage representation by looking over the whole passage and aggregating evidence relevant to the current passage word and question, allowing our model make full use of passage information. Weightedly attending to word context has been proposed in several works. Ling et al. (2015) propose considering window-based contextual words differently depending on the word and its relative position. Cheng et al. (2016) propose a novel LSTM network to encode words in a sentence which considers the relation between the current token being processed and its past tokens in the memory. Parikh et al. (2016) apply this method to encode words in a sentence according to word form and its distance. Since passage information relevant to question is more helpful to infer the answer in reading comprehension, we apply self-matching based on question-aware representation and gated attention-based recurrent networks. It helps our model mainly focus on question-relevant evidence in the passage and dynamically look over the whole passage to aggregate evidence.

Another key component of our model is the attention-based recurrent network, which has demonstrated success in a wide range of tasks. Bahdanau et al. (2014) first propose attention-based recurrent networks to infer word-level alignment when generating the target word. Hermann et al. (2015) introduce word-level attention into reading comprehension to model the interaction between questions and passages. Rocktäschel et al. (2015) and Wang and Jiang (2016a) propose determining entailment via word-by-word matching. The gated attention-based recurrent network is a variant of attention-based recurrent network with an additional gate to model the fact that passage parts are of different importance to the particular question for reading comprehension and question answering.

7 Conclusion

In this paper, we present gated self-matching networks for reading comprehension and question answering. We introduce the gated attention-based recurrent networks and self-matching attention mechanism to obtain representation for the question and passage, and then use the pointer-networks to locate answer boundaries. Our model achieves state-of-the-art results on the SQuAD

dataset, outperforming several strong competing systems. As for future work, we are applying the gated self-matching networks to other reading comprehension and question answering datasets, such as the MS MARCO dataset (Nguyen et al., 2016).

Acknowledgement

We thank all the anonymous reviewers for their helpful comments. We thank Pranav Rajpurkar for testing our model on the hidden test dataset. This work is partially supported by National Key Basic Research Program of China under Grant No.2014CB340504 and National Natural Science Foundation of China under Grant No.61273318. The corresponding author of this paper is Baobao Chang.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* .

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1724–1734.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *CoRR* .

Bhuwan Dhingra, Hanxiao Liu, William W. Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *CoRR* .

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*. pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the International Conference on Learning Representations*.

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

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Wang Ling, Yulia Tsvetkov, Silvio Amir, Ramon Fermandez, Chris Dyer, Alan W. Black, Isabel Trancoso, and Chu-Cheng Lin. 2015. Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR* abs/1611.09268.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 193–203.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR* .

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 colocated with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016.*.

Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *CoRR* abs/1606.02245.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* .

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. 2016. Natural language comprehension with the epireader. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 2692–2700.

Shuohang Wang and Jing Jiang. 2016a. Learning natural language inference with LSTM. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*.

Shuohang Wang and Jing Jiang. 2016b. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* .

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Fastqa: A simple and efficient neural architecture for question answering. *arXiv preprint arXiv:1703.04816* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*. Citeseer, pages 2013–2018.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. 2016. Words or characters? fine-grained gating for reading comprehension. *CoRR* abs/1611.01724.

Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. 2016. End-to-end reading comprehension with dynamic answer chunk ranking. *arXiv preprint arXiv:1610.09996* .

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701.

Junbei Zhang, Xiaodan Zhu, Qian Chen, Lirong Dai, and Hui Jiang. 2017. Exploring question understanding and adaptation in neural-network-based question answering. *arXiv preprint arXiv:1703.04617* .

Generating Natural Answers by Incorporating Copying and Retrieving Mechanisms in Sequence-to-Sequence Learning

Shizhu He[1], **Cao Liu**[1,2], **Kang Liu**[1] and **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
{shizhu.he, cao.liu, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Generating answer with natural language sentence is very important in real-world question answering systems, which needs to obtain a right answer as well as a coherent natural response. In this paper, we propose an end-to-end question answering system called COREQA in sequence-to-sequence learning, which incorporates copying and retrieving mechanisms to generate natural answers within an encoder-decoder framework. Specifically, in COREQA, the semantic units (words, phrases and entities) in a natural answer are dynamically predicted from the vocabulary, copied from the given question and/or retrieved from the corresponding knowledge base jointly. Our empirical study on both synthetic and real-world datasets demonstrates the efficiency of COREQA, which is able to generate correct, coherent and natural answers for knowledge inquired questions.

1 Introduction

Question answering (QA) systems devote to providing exact answers, often in the form of phrases and entities for natural language questions (Woods, 1977; Ferrucci et al., 2010; Lopez et al., 2011; Yih et al., 2015), which mainly focus on analyzing questions, retrieving related facts from text snippets or knowledge bases (KBs), and finally predicting the answering semantic units-**SU** (words, phrases and entities) through ranking (Yao and Van Durme, 2014) and reasoning (Kwok et al., 2001).

However, in real-world environments, most people prefer the correct answer replied with a more natural way. For example, most existing commercial products such as Siri[1] will reply a natural answer "*Jet Li is 1.64m in height.*" for the question "*How tall is Jet Li?*", rather than only answering one entity "*1.64m*". Basic on this observation, we define the "natural answer" as the natural response in our daily communication for replying factual questions, which is usually expressed in a complete/partial natural language sentence rather than a single entity/phrase. In this case, the system needs to not only parse question, retrieve relevant facts from KB but also generate a proper reply. To this end, most previous approaches employed message-response patterns. Figure 1 schematically illustrates the major steps and features in this process. The system first needs to recognize the topic entity "*Jet Li*" in the question and then extract multiple related facts *<Jet Li, gender, Male>*, *<Jet Li, birthplace, Beijing>* and *<Jet Li, nationality, Singapore>* from KB. Based on the chosen facts and the commonly used message-response patterns "*where was %entity from?*" - "*%entity was born in %birthplace, %pronoun is %nationality citizen.*"[2], the system could finally generate the natural answer (McTear et al., 2016).

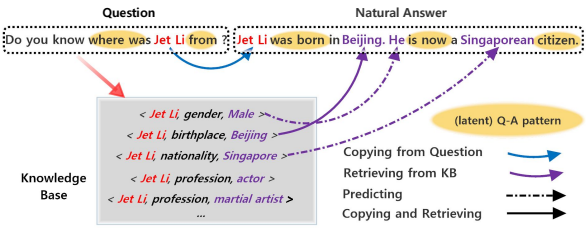


Figure 1: Incorporating copying and retrieving mechanisms in generating a natural answer.

In order to generate natural answers, typical

[1] http://www.apple.com/ios/siri/

[2] In this pattern, *%entity* indicates the placeholder of the topic entity, *%property* indicates the property value of the topic entity.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 199–208
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1019

products need lots of Natural Language Processing (NLP) tools and pattern engineering (McTear et al., 2016), which not only suffers from high costs of manual annotations for training data and patterns, but also have low coverage that cannot flexibly deal with variable linguistic phenomena in different domains. Therefore, this paper devotes to develop an end-to-end paradigm that generates natural answers without any NLP tools (e.g. POS tagging, parsing, etc.) and pattern engineering. This paradigm tries to consider question answering in an end-to-end framework. In this way, the complicated QA process, including analyzing question, retrieving relevant facts from KB, and generating correct, coherent, natural answers, could be resolved jointly.

Nevertheless, generating natural answers in an end-to-end manner is not an easy task. The key challenge is that the words in a natural answer may be generated by different ways, including: *1)* the common words usually are predicted using a (conditional) language model (e.g. "*born*" in Figure 1); *2)* the major entities/phrases are selected from the source question (e.g. "*Jet Li*"); *3)* the answering entities/phrases are retrieved from the corresponding KB (e.g. "*Beijing*"). In addition, some words or phrases even need to be inferred from related knowledge (e.g. "*He*" should be inferred from the value of "*gender*"). And we even need to deal with some morphological variants (e.g. "*Singapore*" in KB but "*Singaporean*" in answer). Although existing end-to-end models for KB-based question answering, such as GenQA (Yin et al., 2016), were able to retrieve facts from KBs with neural models. Unfortunately, they cannot copy SUs from the question in generating answers. Moreover, they could not deal with complex questions which need to utilize multiple facts. In addition, existing approaches for conversational (Dialogue) systems are able to generate natural utterances (Serban et al., 2016; Li et al., 2016) in sequence-to-sequence learning (Seq2Seq). But they cannot interact with KB and answer information-inquired questions. For example, CopyNet (Gu et al., 2016) is able to copy words from the original source in generating the target through incorporating copying mechanism in conventional Seq2Seq learning, but they cannot retrieve SUs from external memory (e.g. KBs, Texts, etc.).

Therefore, facing the above challenges, this paper proposes a neural generative model called COREQA with Seq2Seq learning, which is able to reply an answer in a natural way for a given question. Specifically, we incorporate **CO**pying and **RE**trieving mechanisms within Seq2Seq learning. COREQA is able to analyze the question, retrieve relevant facts and generate a sequence of SUs using a hybrid method with a completely end-to-end learning framework. We conduct experiments on both synthetic data sets and real-world datasets, and the experimental results demonstrate the efficiency of COREQA compared with existing end-to-end QA/Dialogue methods.

In brief, our main contributions are as follows:

- We propose a new and practical question answering task which devotes to generating natural answers for information inquired questions. It can be regarded as a fusion task of QA and Dialogue.
- We propose a neural network based model, named as COREQA, by incorporating copying and retrieving mechanism in Seq2Seq learning. In our knowledge, it is the first end-to-end model that could answer complex questions in a natural way.
- We implement experiments on both synthetic and real-world datasets. The experimental results demonstrate that the proposed model could be more effective for generating correct, coherent and natural answers for knowledge inquired questions compared with existing approaches.

2 Background: Neural Models for Sequence-to-Sequence Learning

2.1 RNN Encoder-Decoder

Recurrent Neural Network (RNN) based Encoder-Decoder is the backbone of Seq2Seq learning (Cho et al., 2014). In the Encoder-Decoder framework, an encoding RNN first transform a source sequential object $X = [x_1, ..., x_{L_X}]$ into an encoded representation $\mathbf{c}$. For example, we can utilize the basic model: $\mathbf{h}_t = f(x_t, \mathbf{h}_{t-1}); \mathbf{c} = \phi(\mathbf{h}_1, ..., \mathbf{h}_{L_X})$, where $\{\mathbf{h}_t\}$ are the RNN hidden states, $\mathbf{c}$ is the context vector which could be assumed as an abstract representation of X. In practice, gated RNN variants such as LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Chung et al., 2014) are commonly used for learning long-term dependencies. And the another encoding

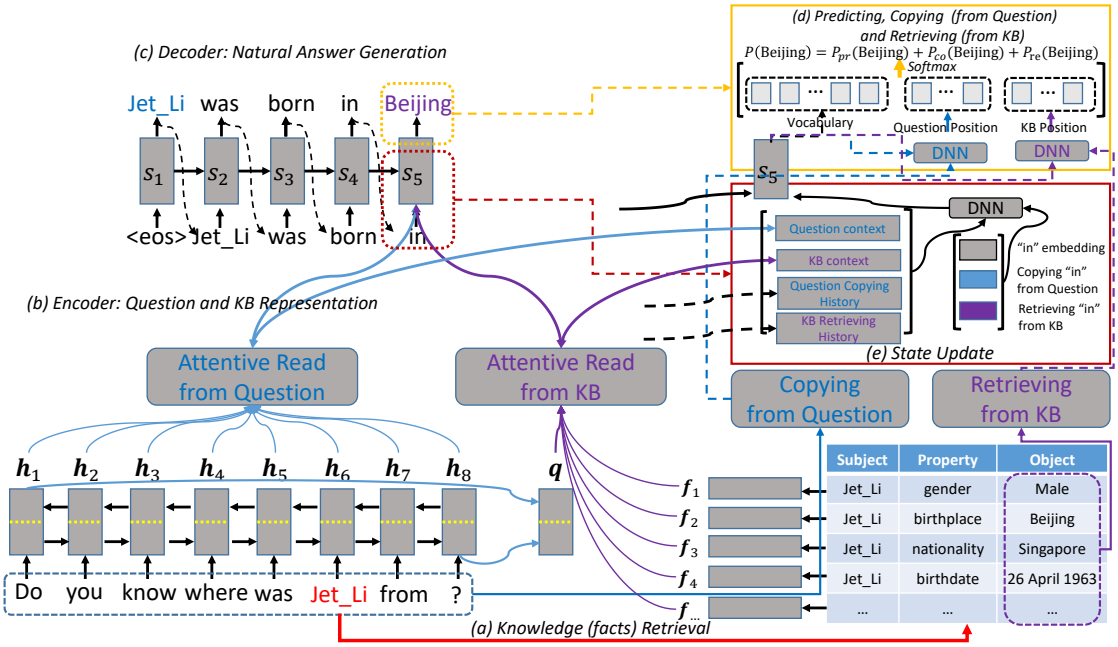


Figure 2: The overall diagram of COREQA.

tricks is Bi-directional RNN, which connect two hidden states of positive time direction and negative time direction. Once the source sequence is encoded, another decoding RNN model is to generate a target sequence $Y = [y_1, ..., y_{L_Y}]$, through the following prediction model: $\mathbf{s}_t = f(y_{t-1}, \mathbf{s}_{t-1}, \mathbf{c})$; $p(y_t|y_{<t}, X) = g(y_{t-1}, \mathbf{s}_t, \mathbf{c})$, where $\mathbf{s}_t$ is the RNN hidden state at time t, the predicted target word y_t at time t is typically performed by a $softmax$ classifier over a settled vocabulary (e.g. 30,000 words) through function g.

2.2 The Attention Mechanism

The prediction model of classical decoders for each target word y_i share the same context vector $\mathbf{c}$. However, a fixed vector is not enough to obtain a better result on generating a long targets.The attention mechanism in the decoding can dynamically choose context $\mathbf{c}_t$ at each time step (Bahdanau et al., 2014), for example, representing $\mathbf{c}_t$ as the weighted sum of the source states $\{\mathbf{h}_t\}$,

$$\mathbf{c}_t = \sum_{i=1}^{L_X} \alpha_{ti}\mathbf{h}_i; \quad \alpha_{ti} = \frac{e^{\rho(\mathbf{s}_{t-1}, \mathbf{h}_i)}}{\sum_{i'} e^{\rho(\mathbf{s}_{t-1}, \mathbf{h}'_i)}} \quad (1)$$

where the function ρ use to compute the attentive strength with each source state, which usually adopts a neural network such as multi-layer perceptron (MLP).

2.3 The Copying Mechanism

Seq2Seq learning heavily rely on the "meaning" for each word in source and target sequences, however, some words in sequences are "no-meaning" symbols and it is improper to encode them in encoding and decoding processes. For example, generating the response *"Of course, read"* for replying the message *"Can you read the word 'read'?"* should not consider the meaning of the second *"read"*. By incorporating the copying mechanism, the decoder could directly copy the sub-sequences of source into the target (Vinyals et al., 2015). The basic approach is to jointly predict the indexes of the target word in the fixed vocabulary and/or matched positions in the source sequences (Gu et al., 2016; Gulcehre et al., 2016).

3 COREQA

To generate natural answers for information inquired questions, we should first recognize key topics in the question, then extract related facts from KB, and finally fusion those instance-level knowledge with some global-level "smooth" and "glue" words to generate a coherent reply. In this section, we present COREQA, a differentiable Seq2Seq model to generate natural answers, which is able to analyze the question, retrieve relevant facts and predict SUs in an end-to-end fashion, and the predicted SUs may be predicted from the vo-

cabulary, copied from the given question, and/or retrieved from the corresponding KB.

3.1 Model Overview

As illustrated in Figure 2, COREQA is an encoder-decoder framework plugged with a KB engineer. A knowledge retrieval module is firstly employed to retrieve related facts from KB by question analysis (see Section 3.2). And then the input question and the retrieved facts are transformed into the corresponding representations by **Encoders** (see Section 3.3). Finally, the encoded representations are feed to **Decoder** for generating the target natural answer (see Section 3.4).

3.2 Knowledge (facts) Retrieval

We mainly focus on answering the information inquired questions (factual questions, and each question usually contains one or more topic entities). This paper utilizes the gold topic entities for simplifying our design. Given the topic entities, we retrieve the related facts from the corresponding KB. KB consists of many relational data, which usually are sets of inter-linked subject-property-object (*SPO*) triple statements. Usually, question contains the information used to match the *subject* and *property* parts in a fact triple, and answer incorporates the *object* part information.

3.3 Encoder

The encoder transforms all discrete input symbols (including words, entities, properties and properties' values) and their structures into numerical representations which are able to feed into neural models (Weston et al., 2014).

3.3.1 Question Encoding

Following (Gu et al., 2016), a bi-directional RNN (Schuster and Paliwal, 1997) is used to transform the question sequence into a sequence of concatenated hidden states with two independent RNNs. The forward and backward RNN respectively obtain $\{\overrightarrow{\mathbf{h}}_1, ..., \overrightarrow{\mathbf{h}}_{L_X}\}$ and $\{\overleftarrow{\mathbf{h}}_{L_X}, ..., \overleftarrow{\mathbf{h}}_1\}$. The concatenated representation is considered to be the short-term memory of question ($\mathbf{M}_Q = \{\mathbf{h}_t\}, \mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_{L_X-t+1}]$). $\mathbf{q} = [\overrightarrow{\mathbf{h}}_{L_X}, \overleftarrow{\mathbf{h}}_1]$ is used to represent the entire question, which could be used to compute the similarity between the question and the retrieved facts.

3.3.2 Knowledge Base Encoding

We use s, p and o denote the subject, property and object (value) of one fact f, and $\mathbf{e}_s$, $\mathbf{e}_p$ and $\mathbf{e}_o$ to denote its corresponding embeddings. The fact representation $\mathbf{f}$ is then defined as the concatenation of $\mathbf{e}_s$, $\mathbf{e}_p$ and $\mathbf{e}_o$. The list of all related facts' representations, $\{\mathbf{f}\} = \{\mathbf{f}_1, ..., \mathbf{f}_{L_F}\}$ (refer to $\mathbf{M}_{KB}$, L_F denotes the maximum of candidate facts), is considered to be a short-term memory of KB while answering questions about the topic entities.

In addition, given the distributed representation of question and candidate facts, we define the matching scores function between question and facts as $S(q, f_j) = DNN_1(\mathbf{q}, \mathbf{f}_j) = tanh(\mathbf{W}_2 \cdot tanh(\mathbf{W}_1 \cdot [\mathbf{q}, \mathbf{f}_j] + \mathbf{b}_1) + \mathbf{b}_2)$, , where DNN_1 is the matching function defined by a two-layer perceptron, $[\cdot, \cdot]$ denotes vector concatenation, and $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are the learning parameters. In fact, we will make a slight change of the matching function because it will also depend on the state of decoding process at different times. The modified function is $S(q, s_t, f_j) = DNN_1(\mathbf{q}, \mathbf{s}_t, \mathbf{f}_j)$ where s_t is the hidden state of decoder at time t.

3.4 Decoder

The decoder uses an RNN to generate a natural answer based on the short-term memory of question and retrieved facts which represented as $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$, respectively. The decoding process of COREQA have the following differences compared with the conventional decoder:

Answer words prediction: COREQA predicts SUs based on a mixed probabilistic model of three modes, namely the ***predict-mode***, the ***copy-mode*** and the ***retrieve-mode***, where the first mode predicts words with the vocabulary, and the two latter modes pick SUs from the questions and matched facts, respectively;

State update: the predicted word at step $t-1$ is used to update $\mathbf{s}_t$, but COREQA uses not only its word embedding but also its corresponding positional attention informations in $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$;

Reading short-Memory $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$: $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$ are fed into COREQA with two ways, the first one is the "meaning" with embeddings and the second one is the positions of different words (properties' values).

3.4.1 Answer Words Prediction

The generated words (entities) may come from vocabulary, source question and matched KB. Accordingly, our model use three correlative output

layer: *shortlist* prediction layer, *question location* copying layer and *candidate-facts location* retrieving layer, respectively. And we use the $softmax$ classifier of the above three cascaded output layers to pick SUs. We assume a vocabulary $\mathcal{V} = \{v_1, ..., v_N\} \cup \{\text{UNK}\}$, where UNK indicates any out-of-vocabulary (OOV) words. Therefore, we have adopted another two set of SUs $\mathcal{X}_Q$ and $\mathcal{X}_{KB}$ which cover words/entities in the source question and the partial KB. That is, we have adopted the instance-specific vocabulary $\mathcal{V} \cup \mathcal{X}_Q \cup \mathcal{X}_{KB}$ for each question. It's important to note that these three vocabularies $\mathcal{V}$, $\mathcal{X}_Q$ and $\mathcal{X}_{KB}$ may overlap.

At each time step t in the decoding process, given the RNN state $\mathbf{s}_t$ together with $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$, the probabilistic function for generating any target SU y_t is a "mixture" model as follow

$$\begin{aligned} &p(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{M}_Q, \mathbf{M}_{KB}) = \\ &p_{pr}(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t) \cdot p_m(pr|\mathbf{s}_t, y_{t-1}) + \\ &p_{co}(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{M}_Q) \cdot p_m(co|\mathbf{s}_t, y_{t-1}) + \\ &p_{re}(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{M}_{KB}) \cdot p_m(re|\mathbf{s}_t, y_{t-1}) \end{aligned} \tag{2}$$

where pr, co and re stand for the predict-mode, the copy-mode and the retrieve-mode, respectively, $p_m(\cdot|\cdot)$ indicates the probability model for choosing different modes (we use a $softmax$ classifier with two-layer MLP). The probability of the three modes are given by

$$\begin{aligned} p_{pr}(y_t|\cdot) &= \frac{1}{Z} e^{\psi_{pr}(y_t)} \\ p_{co}(y_t|\cdot) &= \frac{1}{Z} \sum\nolimits_{j:Q_j=y_t} e^{\psi_{co}(y_t)} \\ p_{re}(y_t|\cdot) &= \frac{1}{Z} \sum\nolimits_{j:KB_j=y_t} e^{\psi_{re}(y_t)} \end{aligned} \tag{3}$$

where $\psi_{pr}(\cdot)$, $\psi_{co}(\cdot)$ and $\psi_{re}(\cdot)$ are score functions for choosing SUs in predict-mode (from $\mathcal{V}$), copy-mode (from $\mathcal{X}_Q$) and retrieve-mode (from $\mathcal{X}_{KB}$), respectively. And Z is the normalization term shared by the three modes, $Z = e^{\psi_{pr}(v)} + \sum_{j:Q_j=v} e^{\psi_{co}(v)} + \sum_{j:KB_j=v} e^{\psi_{re}(v)}$. And the three modes could compete with each other through a $softmax$ function in generating target SUs with the shared normalization term (as shown in Figure 2. Specifically, the scoring functions of each mode are defined as follows:

Predict-mode: Some generated words need reasoning (e.g. "He" in Figure 1) and morphological transformation (e.g. "Singaporean" in Figure 1). Therefore, we modify the function as $\psi_{pr}(y_t = v_i) = \mathbf{v}_i^T \mathbf{W}_{pr}[\mathbf{s}_t, \mathbf{c}_{q_t}, \mathbf{c}_{kb_t}]$, where $\mathbf{v}_i \in \mathcal{R}^{d_o}$ is the word vector at the output layer (not the input word embedding), $\mathbf{W}_{pr} \in \mathcal{R}^{(d_h+d_i+d_f)\times d_o}$ (d_i, d_h and d_f indicate the size of input word vector, RNN decoder hidden state and fact representation respectively), and $\mathbf{c}_{q_t}$ and $\mathbf{c}_{kb_t}$ are the temporary memory of reading $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$ at time t (see Section 3.4.3).

***Co*py-mode**: The score for "copying" the word x_j from question Q is calculated as $\psi_{co}(y_t = x_j) = DNN_2(\mathbf{h}_j, \mathbf{s}_t, \mathbf{hist}_Q)$, where DNN_2 is a neural network function with a two-layer MLP and $\mathbf{hist}_Q \in \mathcal{R}^{L_X}$ is an accumulated vector which record the attentive history for each word in question (similar with the coverage vector in (Tu et al., 2016)).

***Re*trieve-mode**: The score for "retrieving" the entity word v_j from retrieval facts ("Object" part) is calculated as $\psi_{re}(y_t = v_j) = DNN_3(\mathbf{f}_j, \mathbf{s}_t, \mathbf{hist}_{KB})$, where DNN_3 is also a neural network function and $\mathbf{hist}_{KB} \in \mathcal{R}^{L_F}$ is an accumulated vector which record the attentive history for each fact in candidate facts.

3.4.2 State Update

In the generic decoding process, each RNN hidden state $\mathbf{s}_t$ is updated with the previous state $\mathbf{s}_{t-1}$, the word embedding of previous predicted symbol y_{t-1}, and an optional context vector $\mathbf{c}_t$ (with attention mechanism). However, y_{t-1} may not come from vocabulary $\mathcal{V}$ and not owns a word vector. Therefore, we modify the state update process in COREQA. More specifically, y_{t-1} will be represented as concatenated vector of $[\mathbf{e}(y_{t-1}), \mathbf{r}_{q_{t-1}}, \mathbf{r}_{kb_{t-1}}]$, where $\mathbf{e}(y_{t-1})$ is the word embedding associated with y_{t-1}, $\mathbf{r}_{q_{t-1}}$ and $\mathbf{r}_{kb_{t-1}}$ are the weighted sum of hidden states in $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$ corresponding to y_{t-1} respectively.

$$\begin{aligned} \mathbf{r}_{q_t} &= \sum\nolimits_{j=1}^{L_X} \rho_{tj} \mathbf{h}_j, \mathbf{r}_{kb_t} = \sum\nolimits_{j=1}^{L_F} \delta_{tj} \mathbf{f}_j \\ \rho_{tj} &= \begin{cases} \frac{1}{K_1} p_{co}(x_j|\cdot), & x_j = y_t \\ \mathbf{0} & otherwise \end{cases} \\ \delta_{tj} &= \begin{cases} \frac{1}{K_2} p_{re}(f_j|\cdot), & object(f_j) = y_t \\ \mathbf{0} & otherwise \end{cases} \end{aligned} \tag{4}$$

where $object(f)$ indicate the "object" part of fact f (see Figure 2), and K_1 and K_2 are the normalization terms which equal $\sum_{j':x'_j=y_t} p_{co}(x'_j|\cdot)$ and $\sum_{j':object(f'_j)=y_t} p_{re}(f'_j|\cdot)$, respectively, and it

could consider the multiple positions matching y_t in source question and KB.

3.4.3 Reading short-Memory $\mathbf{M}_Q$ and $\mathbf{M}_{KB}$

COREQA employ the attention mechanism at decoding process. At each decoder time t, we selective read the context vector $\mathbf{c}_{q_t}$ and $\mathbf{c}_{kb_t}$ from the short-term memory of question $\mathbf{M}_Q$ and retrieval facts $\mathbf{M}_{KB}$ (alike to Formula 1). In addition, the accumulated attentive vectors $\mathbf{hist}_Q$ and $\mathbf{hist}_{KB}$ are able to record the positional information of SUs in the source question and retrieved facts.

3.5 Training

Although some target SUs in answer are copied and retrieved from the source question and the external KB respectively, COREQA is fully differential and can be optimized in an end-to-end manner using back-propagation. Given the batches of the source questions $\{X\}_M$ and target answers $\{Y\}_M$ both expressed with natural language (symbolic sequences), the objective function is to minimize the negative log-likelihood:

$$\mathcal{L} = -\frac{1}{N}\sum_{k=1}^{M}\sum_{t=1}^{L_Y} log[p(y_t^{(k)}|y_{<t}^{(k)}, X^{(k)}] \quad (5)$$

where the superscript (k) indicates the index of one question-answer (Q-A) pair. The network is no need for any additional labels for training models, because the three modes sharing the same $softmax$ classifier for predicting target words, they can learn to coordinate with each other by maximizing the likelihood of observed Q-A pairs.

4 Experiments

In this section, we present our main experimental results in two datasets. The first one is a small synthetic dataset in a restricted domain (only involving four properties of persons) (Section 4.1). The second one is a big dataset in open domain, where the Q-A pairs are extracted from community QA website and grounded against a KB with an Integer Linear Programming (ILP) method (Section 4.2). COREQA and all baseline models are trained on a NVIDIA TITAN X GPU using TensorFlow[3] tools, where we used the Adam (Kingma and Ba, 2014) learning rule to update gradients in all experimental configures. The sources codes and data will be released at the personal homepage of the first author[4].

4.1 Natural QA in Restricted Domain

Task: The QA systems need to answer questions involving 4 concrete properties of birthdate (including `year`, `month` and `day`) and `gender`). Through merely involving 4 properties, there are plenty of QA patterns which focus on different aspects of birthdate, for example, "*What year were you born?*" touches on "`year`", but "*When is your birthday?*" touches on "`month` and `day`".

Dataset: Firstly, 108 different Q-A patterns have been constructed by two annotators, one in charge of raising question patterns and another one is responsible for generating corresponding suitable answer patterns, e.g. `When is` $\%e$ `birthday?` $\rightarrow$ `She was born in` $\%m$ $\%d$`th.` where the variables $\%e$, $\%y$, $\%m$, $\%d$ and $\%g$ (deciding *she* or *he*) indicates the person's name, birth year, birth month, birth day and gender, respectively. Then we randomly generate a KB which contains 80,000 person entities, and each entity including four facts. Given KB facts, we can finally obtain specific Q-A pairs. And the sampling KB, patterns, and the generated Q-A pairs are shown in Table 1. In order to maintain the diversity, we randomly select 6 patterns for each person. Finally, we totally obtain 239,934 sequences pairs (half patterns may be unmatched because of "gender" property).

Q-A Patterns	**Examples** (e.g. KB facts (e2,year,1987);(e2,month,6); (e2,day,20);(e2,gender,male))
When is $\%e$ birthday? He was born in $\%m$ $\%d$th.	When is e2 birthday? He was born in June 20th.
What year were $\%e$ born? $\%e$ is born in $\%y$ year.	What year were e2 born? e2 is born in 1987 year.

Table 1: Sample KB facts, patterns and their generated Q-A pairs.

Experimental Setting: The total 239,934 Q-A pairs are split into training (90%) and testing set (10%). The baseline includes 1) generic RNN Encoder-Decoder (marked as *RNN*), 2) Seq2Seq with attention (marked as *RNN+atten*), 3) *CopyNet*, and 4) *GenQA*. For a fair comparison, we use bi-directional LSTM for encoder and another LSTM for decoder for all Seq2Seq models, with hidden layer size = 600 and word embedding dimen-

[3]https://www.tensorflow.org/

[4]http://www.nlpr.ia.ac.cn/cip/shizhuhe/publications.html

sion = 200. We set L_F as 5.

Metrics: We adopt (**automatic evaluation (AE)** to test the effects of different models. **AE** considers the precisions of the entire predicted answers and four specific properties, and the answer is complete correct only when all predicted properties' values is right. To measure the performance of the proposed method, we select following metrics, including P_g[5], P_y, P_m and P_d which denote the precisions for 'gender', 'year', 'month' and 'day' properties, respectively. And P_A, R_A and $F1_A$ indicate the precision, recall and F1 in the complete way.

Experimental Results: The **AE** experimental results are shown in Table 2. It is very clear from Table 2 that COREQA significantly outperforms all other compared methods. The reason of the GenQA's poor performance is that all synthetic questions need multiple facts, and GenQA will "safely" choose the most frequent property ("gender") for all questions. We also found the performances on "year" and "day" have a little worse than other properties such as "gender", it may because there have more ways to answer questions about "year" and "day".

Models	P_g	P_y	P_m	P_d	P_A	R_A	$F1_A$
RNN	72.2	0	1.1	0.2	0	27.5	0
RNN+atten	55.8	1.1	11.3	9.5	1.7	34	3.2
CopyNet	75.2	8.7	28.3	5.8	3.7	32.5	6.7
GenQA	73.4	0	0	0	0	27.1	0
COREQA	**100**	**84.8**	**93.4**	**81**	**87.4**	**94**	**90.6**

Table 2: The AE results (%) on synthetic test data.

Discussion: Because of the feature of directly "hard" copy and retrieve SUs from question and KB, COREQA could answer questions about unseen entities.To evaluate the effects of answering questions about unseen entities, we re-construct 2,000 new person entities and their corresponding facts about four known properties, and obtain 6,081 Q-A pairs through matching the sampling patterns mentioned above. The experimental results are shown in Table 3, it can be seen that the performance did not fall too much.

Entities	P_g	P_y	P_m	P_d	P_A	R_A	$F1_A$
Seen	100	84.8	93.4	81	87.4	94	90.6
Unseen	75.1	84.5	93.5	81.2	63.8	85.1	73.1

Table 3: The AE (%) for seen and unseen entities.

[5]The "gender" is right when the entity name (e.g. 'e2') or the personal pronoun (e.g. 'She') in answer is correct.

4.2 Natural QA in Open Domain

Task: To test the performance of the proposed approach in open domains, we modify the task of GenQA (Yin et al., 2016) for supporting multi-facts (a typical example is shown in Figure 1). That is, a natural QA system should generate a sequence of SUs as the natural answer for a given natural language question through interacting with a KB.

Dataset: GenQA have released a corpus[6], which contains a crawling KB and a set of ground Q-A pairs. However, the original Q-A pairs only matched with just one single fact. In fact, we found that a lot of questions need more than one fact (about 20% based on sampling inspection). Therefore, we crawl more Q-A pairs from Chinese community QA website (Baidu Zhidao[7]). Combined with the originally published corpus, we create a lager and better-quality data for natural question answering. Specifically, an Integral Linear Programming (ILP) based method is employed to automatically construct "grounding" Q-A pairs with the facts in KB (inspired by the work of adopting ILP to parse questions (Yahya et al., 2012)). In ILP, the main constraints and considered factors are listed below: 1) the "`subject`" entity and "`object`" entity of a triple have to match with question words/phrases (marked as *subject mention*) and answer words/phrases (marked as *object mention*) respectively; 2) any two *subject mentions* or *object mentions* should not overlap; 3) a *mention* can match at most one `entity`; 4) the edit distance between the Q-A pair and the matched candidate fact (use a space to joint three parts) is smaller, they are more relevant. Finally, we totally obtain 619,199 instances (an instance contains a question, an answer, and multiple facts), and the number of instances that can match one and multiple facts in KB are 499,809 and 119,390, respectively. Through the evaluation of 200 sampling instances, we estimate that approximate 81% matched facts are helpful for the generating answers. However, strictly speaking, only 44% instances are truly correct grounding. In fact, grounding the Q-A pairs from community QA website is a very challenge problem, we will leave it in the future work.

Experimental Setting: The dataset is split into training (90%) and testing set (10%). The sen-

[6]https://github.com/jxfeb/Generative_QA

[7]https://zhidao.baidu.com/

tences in Chinese are segmented into word sequences with Jieba[8] tool. And we use the words with the frequency larger than 3, which covering 98.4% of the word in the corpus. For a fair comparison, we use bi-directional LSTM for the encoder and another LSTM for decoder for all Seq2Seq models, with hidden layer size = 1024 and word embedding dimension = 300. We select CopyNet (more advanced Seq2Seq model) and GenQA for comparison. We set L_F as 10.

Metrics: Besides adopting the **AE** as a metric (same as GenQA (Yin et al., 2016)), we additionally use **manual evaluation (ME)** as another metric. **ME** considers three aspects about the quality of the generated answer (refer to (Asghar et al., 2016)): 1) **correctness**; 2) syntactical **fluency**; 3) **coherence** with the question. We employ two annotators to rate such three aspects of CopyNet, GenQA and COREQA. Specifically, we sample 100 questions, and conduct $C_3^2 = 3$ pair-wise comparisons for each question and count the winning times of each model (comparisons may both win or both lose).

Experimental Results: The **AE** and **ME** results are shown in Table 4 and Table 5, respectively. Meanwhile, we separately present the results according to the number of the facts which a question needs in KB, including just one single fact (marked as **Single**), multiple facts (marked as **Multi**) and all (marked as **Mixed**). In fact, we train two separate models for **Single** and **Multi** questions for the unbalanced data . From Table 4 and Table 5, we can clearly observe that COREQA significantly outperforms all other baseline models. And COREQA could generate a better natural answer in three aspects: correctness, fluency and coherence. CopyNet cannot interact with KB which is important to generate correct answers. For example, for "Who is the director of The_Little_Chinese_Seamstress?", if without the fact (The_Little_Chinese_Seamstress, director, Dai_Siji), QA systems cannot generate a correct answer.

Models	Single	Multi	Mixed
CopyNet	9.7	0.8	8.7
GenQA	47.2	28.9	45.1
COREQA	**58.4**	**42.7**	**56.6**

Table 4: The AE accuracies (%) on real world test data.

[8]https://github.com/fxsjy/jieba

Models	Correctness	Fluency	Coherence
CopyNet	0	13.3	3.3
GenQA	26.7	33.3	20
COREQA	**46.7**	**50**	**60**

Table 5: The ME results (%) on sampled mixed test data.

Case Study and Error Analysis: Table 6 gives some examples of generated by COREQA and the gold answers to the questions in test set. It is very clearly seen that the parts of generating SUs are predicted from the vocabulary, and other SUs are copied from the given question (marked as **bold**) and retrieved from the KB (marked as underline). And we analyze sampled examples and believe that there are several major causes of errors: **1)** did not match the right facts (ID 6); **2)** the generated answers contain some repetition of meaningless words (ID 7); **3)** the generated answers are not coherence natural language sentences (ID 8).

5 Related Work

Seq2Seq learning is to maximize the likelihood of predicting the target sequence Y conditioned on the observed source sequence X (Sutskever et al., 2014), which has been applied successfully to a large number of NLP tasks such as Machine Translation (Wu et al., 2016) and Dialogue (Vinyals and Le, 2015). Our work is partially inspired by the recent work of QA and Dialogue which have adopted Seq2Seq learning. CopyNet (Gu et al., 2016) and Pointer Networks (Vinyals et al., 2015; Gulcehre et al., 2016) which could incorporate copying mechanism in conventional Seq2Seq learning. Different from our application which deals with knowledge inquired questions and generates natural answers, CopyNet (Gu et al., 2016) and Pointer Networks (Gulcehre et al., 2016) can only copy words from the original input sequence. In contrast, COREQA is able to retrieve SUs from external memory. And GenQA (Yin et al., 2016) can only deal with the simple questions which could be answered by one fact, and it also did not incorporate the copying mechanism in Seq2Seq learning.

Moreover, our work is also inspired by Neural Abstract Machine (Graves et al., 2016; Yin et al., 2015; Liang et al., 2016) which could retrieve facts from KBs with neural models. Unlike natural answer, Neural Abstract Machine (Mou et al., 2016) is concentrating on obtaining concrete answer en-

ID	Question	Gold Answer	Generated Natural Answer
1	拜仁的哈格里夫斯是哪国人？ Which country did *Hargreaves* of Bayern comes from?	英国人！加拿大出生 British! born in Canada	加拿大人，英国人 Canadians, British
2	沙祖康是哪个族的？ Which ethnic groups is *Sha Zukang* from?	汉族现在是联合国副秘书长 Han, now he is the DSG of the UN	联合国副秘书长，是汉族 DSG of the UN, Han
3	罗伯特·舒曼是谁？ Who is *Robert Schumann*?	是一个作家、音乐家 a writer, musician	**罗伯特·舒曼**个作家 哲学家 **Robert Schuhmann** is a writer and philosopher.
4	马斯切拉诺是谁？ Who is *Mascherano*?	他是一名足球运动员 He is a football player	一个优秀的足球 运动员 An excellent football player
5	巴尔扎克与小裁缝的导演是谁？ Who is the director of *The Little Chinese Seamstress*?	戴思杰 Dai Sijie	**导演**是戴思杰 **Director** Dai Sijie
6	电影铁三角是谁拍的？ Who shot the movie *The Iron Triangle*?	徐克、林岭东、杜琪峰 Tsui Hark, Johnny To, Ringo Lam	冯小刚 Feng Xiaogang
7	谁知道席慕蓉这个人的一些详细资料？ Who knows some details of *Xi Murong*?	女诗人 poetess	是是是 yes, yes, yes
8	泡泡堂是哪个公司开发的？ Which company developed the game *Crazy Arcade*?	应该是上海盛大 should be the Shanda Group	玩上海盛大 playing Shanda Group

Table 6: Examples of the generated natural answers by COREQA.

tities with neural network based reasoning.

6 Conclusion and Future Work

In this paper, we propose an end-to-end system to generate natural answers through incorporating copying and retrieving mechanisms in sequence-to-sequence learning. Specifically, the sequences of SUs in the generated answer may be predicted from the vocabulary, copied from the given question and retrieved from the corresponding KB. And the future work includes: a) lots of questions cannot be answered directly by facts in a KB (e.g. "*Who is Jet Li's father-in-law?*"), we plan to learn QA system with latent knowledge (e.g. KB embedding (Bordes et al., 2013)); b) we plan to adopt memory networks (Sukhbaatar et al., 2015) to encode the temporary KB for each question.

Acknowledgments

The authors are grateful to anonymous reviewers for their constructive comments. The work was supported by the Natural Science Foundation of China (No.61533018) and the National High Technology Development 863 Program of China (No.2015AA015405).

References

Nabiha Asghar, Pascal Poupart, Jiang Xin, and Hang Li. 2016. Online sequence-to-sequence reinforcement learning for open-domain conversational agents. *arXiv preprint arXiv:1612.03929* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. pages 2787–2795.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

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

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538(7626):471–476.

Jiatao Gu, Zhengdong Lu, Hang Li, and O.K. Victor Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1631–1640.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 140–149.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Cody Kwok, Oren Etzioni, and Daniel S Weld. 2001. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)* 19(3):242–262.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1192–1202.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020* .

Vanessa Lopez, Victoria Uren, Marta Sabou, and Enrico Motta. 2011. Is question answering fit for the semantic web?: a survey. *Semantic Web* 2(2):125–155.

Michael McTear, Zoraida Callejas, and David Griol. 2016. *The Conversational Interface: Talking to Smart Devices*. Springer Publishing Company, Incorporated, 1st edition.

Lili Mou, Zhengdong Lu, Hang Li, and Zhi Jin. 2016. Coupling distributed and symbolic execution for natural language queries. *arXiv preprint arXiv:1612.02741* .

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

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. *arXiv preprint arXiv:1601.04811* .

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916.

William A Woods. 1977. Lunar rocks in natural english: Explorations in natural language question answering. In *Linguistic structures processing*. pages 521–569.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 379–390.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 956–966.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1321–1331.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables. *arXiv preprint arXiv:1512.00965* .

Coarse-to-Fine Question Answering for Long Documents

Eunsol Choi[†]
University of Washington
eunsol@cs.washington.edu

Daniel Hewlett, Jakob Uszkoreit
Google
{dhewlett,usz}@google.com

Illia Polosukhin[†]
XIX.ai
i@xix.ai

Alexandre Lacoste[†]
Element AI
allac@elementai.com

Jonathan Berant[†]
Tel Aviv University
joberant@cs.tau.ac.il

Abstract

We present a framework for question answering that can efficiently scale to longer documents while maintaining or even improving performance of state-of-the-art models. While most successful approaches for reading comprehension rely on recurrent neural networks (RNNs), running them over long documents is prohibitively slow because it is difficult to parallelize over sequences. Inspired by how people first skim the document, identify relevant parts, and carefully read these parts to produce an answer, we combine a coarse, fast model for selecting relevant sentences and a more expensive RNN for producing the answer from those sentences. We treat sentence selection as a latent variable trained jointly from the answer only using reinforcement learning. Experiments demonstrate the state of the art performance on a challenging subset of the WIKIREADING dataset (Hewlett et al., 2016) and on a new dataset, while speeding up the model by 3.5x-6.7x.

1 Introduction

Reading a document and answering questions about its content are among the hallmarks of natural language understanding. Recently, interest in question answering (QA) from unstructured documents has increased along with the availability of large scale datasets for reading comprehension (Hermann et al., 2015; Hill et al., 2015; Rajpurkar et al., 2016; Onishi et al., 2016; Nguyen et al., 2016; Trischler et al., 2016a).

Current state-of-the-art approaches for QA over documents are based on recurrent neural networks (RNNs) that encode the document and the question to determine the answer (Hermann et al., 2015; Chen et al., 2016; Kumar et al., 2016; Kadlec et al., 2016; Xiong et al., 2016). While such models have access to all the relevant information, they are slow because the model needs to be run sequentially over possibly thousands of tokens, and the computation is not parallelizable. In fact, such models usually truncate the documents and consider only a limited number of tokens (Miller et al., 2016; Hewlett et al., 2016). Inspired by studies on how people answer questions by first skimming the document, identifying relevant parts, and carefully reading these parts to produce an answer (Masson, 1983), we propose a coarse-to-fine model for question answering.

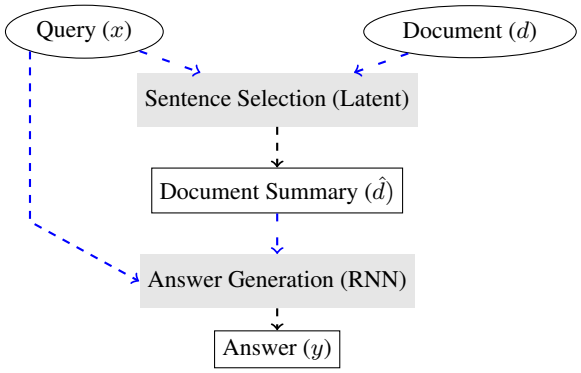


Figure 1: Hierarchical question answering: the model first selects relevant sentences that produce a document summary ($\hat{d}$) for the given query (x), and then generates an answer (y) based on the summary ($\hat{d}$) and the query x.

Our model takes a hierarchical approach (see Figure 1), where first a fast model is used to select a few sentences from the document that are relevant for answering the question (Yu et al., 2014; Yang et al., 2016a). Then, a slow RNN is employed to produce the final answer from the selected sentences. The RNN is run over a fixed number of tokens, regardless of the length of the document. Empirically, our model encodes the

[†]Work done while the authors were at Google.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 209–220
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1020

d: s_1: The 2011 Joplin tornado was a catastrophic EF5-rated multiple-vortex tornado that struck Joplin, Missouri . . .
s_4: It was the third tornado to strike Joplin since May 1971.
s_5: Overall, the tornado killed **158 people** . . ., injured some 1,150 others, and caused damages . . .

x: how many people died in joplin mo tornado

y: 158 people

Figure 2: A training example containing a document d, a question x and an answer y in the WIKISUGGEST dataset. In this example, the sentence s_5 is necessary to answer the question.

text up to 6.7 times faster than the base model, which reads the first few paragraphs, while having access to four times more tokens.

A defining characteristic of our setup is that an answer does not necessarily appear verbatim in the input (the genre of a movie can be determined even if not mentioned explicitly). Furthermore, the answer often appears multiple times in the document in spurious contexts (the year '2012' can appear many times while only once in relation to the question). Thus, we treat sentence selection as a latent variable that is trained jointly with the answer generation model from the answer only using reinforcement learning. Treating sentence selection as a latent variable has been explored in classification (Yessenalina et al., 2010; Lei et al., 2016), however, to our knowledge, has not been applied for question answering.

We find that jointly training sentence selection and answer generation is especially helpful when locating the sentence containing the answer is hard. We evaluate our model on the WIKIREADING dataset (Hewlett et al., 2016), focusing on examples where the document is long and sentence selection is challenging, and on a new dataset called WIKISUGGEST that contains more natural questions gathered from a search engine.

To conclude, we present a modular framework and learning procedure for QA over long text. It captures a limited form of document structure such as sentence boundaries and deals with long documents or potentially multiple documents. Experiments show improved performance compared to the state of the art on the subset of WIKIREADING, comparable performance on other datasets, and a 3.5x-6.7x speed up in document encoding, while allowing access to much longer documents.

	% answer string exists	avg # of ans. match	% match first sent
WIKIREADING	47.1	1.22	75.1
WR-LONG	50.4	2.18	31.3
WIKISUGGEST	100	13.95	33.6

Table 1: Statistics on string matches of the answer y^* in the document. The third column only considers examples with answer match. Often the answer string is missing or appears many times while it is relevant to query only once.

2 Problem Setting

Given a training set of question-document-answer triples $\{x^{(i)}, d^{(i)}, y^{(i)}\}_{i=1}^{N}$, our goal is to learn a model that produces an answer y for a question-document pair (x, d). A document d is a list of sentences $s_1, s_2, \ldots, s_{|d|}$, and we assume that the answer can be produced from a small latent subset of the sentences. Figure 2 illustrates a training example in which sentence s_5 is in this subset.

3 Data

We evaluate on WIKIREADING, WIKIREADING LONG, and a new dataset, WIKISUGGEST.

WIKIREADING (Hewlett et al., 2016) is a QA dataset automatically generated from Wikipedia and Wikidata: given a Wikipedia page about an entity and a Wikidata property, such as PROFESSION, or GENDER, the goal is to infer the target value based on the document. Unlike other recently released large-scale datasets (Rajpurkar et al., 2016; Trischler et al., 2016a), WIKIREADING does not annotate answer spans, making sentence selection more challenging.

Due to the structure and short length of most Wikipedia documents (median number of sentences: 9), the answer can usually be inferred from the first few sentences. Thus, the data is not ideal for testing a sentence selection model compared to a model that uses the first few sentences. Table 1 quantifies this intuition: We consider sentences containing the answer y^* as a proxy for sentences that should be selected, and report how often y^* appears in the document. Additionally, we report how frequently this proxy oracle sentence is the first sentence. We observe that in WIKIREADING, the answer appears verbatim in 47.1% of the examples, and in 75% of them the match is in the first sentence. Thus, the importance of modeling sentence selection is limited.

To remedy that, we filter WIKIREADING and ensure a more even distribution of answers throughout the document. We prune short docu-

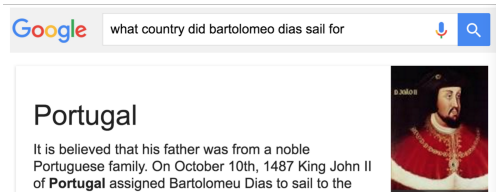
	# of uniq. queries	# of examples	# of words / query	# of tokens / doc.
WIKIREADING	867	18.58M	2.35	489.2
WR-LONG	239	1.97M	2.14	1200.7
WIKISUGGEST	3.47M	3.47M	5.03	5962.2

Table 2: Data statistics.

ments with less than 10 sentences, and only consider Wikidata properties for which Hewlett et al. (2016)'s best model obtains an accuracy of less than 60%. This prunes out properties such as GENDER, GIVEN NAME, and INSTANCE OF.[1] The resulting WIKIREADING LONG dataset contains 1.97M examples, where the answer appears in 50.4% of the examples, and appears in the first sentence only 31% of the time. On average, the documents in WIKIREADING LONG contain 1.2k tokens, more tokens than those of SQuAD (average 122 tokens) or CNN (average 763 tokens) datasets (see Table 2). Table 1 shows that the exact answer string is often missing from the document in WIKIREADING. This is since Wikidata statements include properties such as NATIONALITY, which are not explicitly mentioned, but can still be inferred. A drawback of this dataset is that the queries, Wikidata properties, are not natural language questions and are limited to 858 properties.

To model more realistic language queries, we collect the WIKISUGGEST dataset as follows. We use the Google Suggest API to harvest natural language questions and submit them to Google Search. Whenever Google Search returns a box with a short answer from Wikipedia (Figure 3), we create an example from the question, answer, and the Wikipedia document. If the answer string is missing from the document this often implies a spurious question-answer pair, such as ('what time is half time in rugby', '80 minutes, 40 minutes'). Thus, we pruned question-answer pairs without the exact answer string. We examined fifty examples after filtering and found that 54% were well-formed question-answer pairs where we can ground answers in the document, 20% contained answers without textual evidence in the document (the answer string exists in an irreleveant context), and 26% contain incorrect QA pairs such as the last two examples in Figure 3. The data collection was performed in May 2016.

[1]These three relations alone account for 33% of the data.



WIKISUGGEST Query	Answer
what year did virgina became a state	1788
general manager of smackdown	Theodore Long
minnesota viking colors	purple
coco martin latest movies	maybe this time
longest railway station in asia	Gorakhpur
son from modern family	Claire Dunphy
north dakota main religion	Christian
lands end' brand	Lands' End
wdsu radio station	WCBE

Figure 3: Example queries and answers of WIKISUGGEST.

4 Model

Our model has two parts (Figure 1): a fast sentence selection model (Section 4.1) that defines a distribution $p(s \mid x, d)$ over sentences given the input question (x) and the document (d), and a more costly answer generation model (Section 4.3) that generates an answer y given the question and a document summary, $\hat{d}$ (Section 4.2), that focuses on the relevant parts of the document.

4.1 Sentence Selection Model

Following recent work on sentence selection (Yu et al., 2014; Yang et al., 2016b), we build a feed-forward network to define a distribution over the sentences $s_1, s_2, \ldots, s_{|d|}$. We consider three simple sentence representations: a bag-of-words (BoW) model, a chunking model, and a (parallelizable) convolutional model. These models are efficient at dealing with long documents, but do not fully capture the sequential nature of text.

BoW Model Given a sentence s, we denote by $\text{BoW}(s)$ the bag-of-words representation that averages the embeddings of the tokens in s. To define a distribution over the document sentences, we employ a standard attention model (e.g., (Hermann et al., 2015)), where the BoW representation of the query is concatenated to the BoW representation of each sentence s_l, and then passed through a single layer feed-forward network:

$$h_l = [\text{BoW}(x); \text{BoW}(s_l)]$$
$$v_l = v^\top \text{ReLU}(W h_l),$$
$$p(s = s_l \mid x, d) = \text{softmax}(v_l),$$

where $[;]$ indicates row-wise concatenation, and the matrix W, the vector v, and the word embeddings are learned parameters.

Chunked BoW Model To get more fine-grained granularity, we split sentences into fixed-size smaller chunks (seven tokens per chunk) and score each chunk separately (Miller et al., 2016). This is beneficial if questions are answered with sub-sentential units, by allowing to learn attention over different chunks. We split a sentence s_l into a fixed number of chunks $(c_{l,1}, c_{l,2} \ldots, c_{l,J})$, generate a BoW representation for each chunk, and score it exactly as in the BoW model. We obtain a distribution over chunks, and compute sentence probabilities by marginalizing over chunks from the same sentence. Let $p(c = c_{l,j} \mid x, d)$ be the distribution over chunks from all sentences, then:

$$p(s = s_l \mid x, d) = \sum_{j=1}^{J} p(c = c_{l,j} \mid x, d),$$

with the same parameters as in the BoW model.

Convolutional Neural Network Model While our sentence selection model is designed to be fast, we explore a convolutional neural network (CNN) that can compose the meaning of nearby words. A CNN is still efficient, since all filters can be computed in parallel. Following previous work (Kim, 2014; Kalchbrenner et al., 2014), we concatenate the embeddings of tokens in the query x and the sentence s_l, and run a convolutional layer with F filters and width w over the concatenated embeddings. This results in F features for every span of length w, and we employ max-over-time-pooling (Collobert et al., 2011) to get a final representation $h_l \in \mathbb{R}^F$. We then compute $p(s = s_l \mid x, d)$ by passing h_l through a single layer feed-forward network as in the BoW model.

4.2 Document Summary

After computing attention over sentences, we create a summary that focuses on the document parts related to the question using deterministic soft attention or stochastic hard attention. Hard attention is more flexible, as it can focus on multiple sentences, while soft attention is easier to optimize and retains information from multiple sentences.

Hard Attention We sample a sentence $\hat{s} \sim p(s \mid x, d)$ and fix the document summary $\hat{d} = \hat{s}$ to be that sentence during training. At test time, we choose the most probable sentence. To extend the document summary to contain more information, we can sample without replacement K sentences from the document and define the summary to be the concatenation of the sampled sentences $\hat{d} = [\hat{s}_1; \hat{s}_2; \ldots; \hat{s}_K]$.

Soft Attention In the soft attention model (Bahdanau et al., 2015) we compute a weighted average of the tokens in the sentences according to $p(s \mid x, d)$. More explicitly, let $\hat{d}_m$ be the mth token of the document summary. Then, by fixing the length of every sentence to M tokens,[2] the *blended* tokens are computed as follows:

$$\hat{d}_m = \sum_{l=1}^{|d|} p(s = s_l \mid x, d) \cdot s_{l,m},$$

where $s_{l,m}$ is the mth word in the lth sentence ($m \in \{1, \ldots, M\}$).

As the answer generation models (Section 4.3) take a sequence of vectors as input, we average the tokens at the word level. This gives the hard attention an advantage since it samples a "real" sentence without mixing words from different sentences. Conversely, soft attention is trained more easily, and has the capacity to learn a low-entropy distribution that is similar to hard attention.

4.3 Answer Generation Model

State-of-the-art question answering models use RNN models to encode the document and question and selects the answer. We focus on a hierarchical model with fast sentence selection, and do not subscribe to a particular answer generation architecture.

Here we implemented the state-of-the-art word-level sequence-to-sequence model with placeholders, described by Hewlett et al. (2016). This models can produce answers that does not appear in the sentence verbatim. This model takes the query tokens, and the document (or document summary) tokens as input and encodes them with a Gated Recurrent Unit (GRU; Cho et al. (2014)). Then, the answer is decoded with another GRU model, defining a distribution over answers $p(y \mid x, \hat{d})$. In this work, we modified the original RNN: the word embeddings for the RNN decoder input, output and original word embeddings are shared.

[2] Long sentences are truncated and short ones are padded.

5 Learning

We consider three approaches for learning the model parameters (denoted by θ): (1) We present a pipeline model, where we use distant supervision to train a sentence selection model independently from an answer generation model. (2) The hard attention model is optimized with REINFORCE (Williams, 1992) algorithm. (3) The soft attention model is fully differentiable and is optimized end-to-end with backpropagation.

Distant Supervision While we do not have an explicit supervision for sentence selection, we can define a simple heuristic for labeling sentences. We define the gold sentence to be the first sentence that has a full match of the answer string, or the first sentence in the document if no full match exists. By labeling gold sentences, we can train sentence selection and answer generation independently with standard supervised learning, maximizing the log-likelihood of the gold sentence and answer, given the document and query. Let y^* and s^* be the target answer and sentence , where s^* also serves as the document summary. The objective is to maximize:

$$\begin{aligned} J(\theta) &= \log p_\theta(y^*, s^* \mid x, d) \\ &= \log p_\theta(s^* \mid x, d) + \log p_\theta(y^* \mid s^*, x). \end{aligned}$$

Since at test time we do not have access to the target sentence s^* needed for answer generation, we replace it by the model prediction $\arg\max_{s_l \in d} p_\theta(s = s_l \mid d, x)$.

Reinforcement Learning Because the target sentence is missing, we use reinforcement learning where our action is sentence selection, and our goal is to select sentences that lead to a high reward. We define the reward for selecting a sentence as the log probability of the correct answer given that sentence, that is, $R_\theta(s_l) = \log p_\theta(y = y^* \mid s_l, x)$. Then the learning objective is to maximize the expected reward:

$$\begin{aligned} J(\theta) &= \sum_{s_l \in d} p_\theta(s = s_l \mid x, d) \cdot R_\theta(s_l) \\ &= \sum_{s_l \in d} p_\theta(s = s_l \mid x, d) \cdot \log p_\theta(y = y^* \mid s_l, x). \end{aligned}$$

Following REINFORCE (Williams, 1992), we approximate the gradient of the objective with a sample, $\hat{s} \sim p_\theta(s \mid x, d)$:

$$\begin{aligned} \nabla J(\theta) \approx \nabla \log p_\theta(y \mid \hat{s}, x) \\ + \log p_\theta(y \mid \hat{s}, x) \cdot \nabla \log p_\theta(\hat{s} \mid x, d). \end{aligned}$$

Sampling K sentences is similar and omitted for brevity.

Training with REINFORCE is known to be unstable due to the high variance induced by sampling. To reduce variance, we use curriculum learning, start training with distant supervision and gently transition to reinforcement learning, similar to DAGGER (Ross et al., 2011). Given an example, we define the probability of using the distant supervision objective at each step as r^e, where r is the decay rate and e is the index of the current training epoch.[3]

Soft Attention We train the soft attention model by maximizing the log likelihood of the correct answer y^* given the input question and document $\log p_\theta(y^* \mid d, x)$. Recall that the answer generation model takes as input the query x and document summary $\hat{d}$, and since $\hat{d}$ is an average of sentences weighted by sentence selection, the objective is differentiable and is trained end-to-end.

6 Experiments

Experimental Setup We used 70% of the data for training, 10% for development, and 20% for testing in all datasets. We used the first 35 sentences in each document as input to the hierarchical models, where each sentence has a maximum length of 35 tokens. Similar to Miller et al. (2016), we add the first five words in the document (typically the title) at the end of each sentence sequence for WIKISUGGEST. We add the sentence index as a one hot vector to the sentence representation.

We coarsely tuned and fixed most hyperparameters for all models. The word embedding dimension is set to 256 for both sentence selection and answer generation models. We used the decay rate of 0.8 for curriculum learning. Hidden dimension is fixed at 128, batch size at 128, GRU state cell at 512, and vocabulary size at 100K. For CNN sentence selection model, we used 100 filters and set filter width as five. The initial learning rate and gradient clipping coefficients for each model are tuned on the development set. The ranges for learning rates were 0.00025, 0.0005, 0.001, 0.002, 0.004 and 0.5, 1.0 for gradient clipping coefficient.

[3] We tuned $r \in [0.3, 1]$ on the development set.

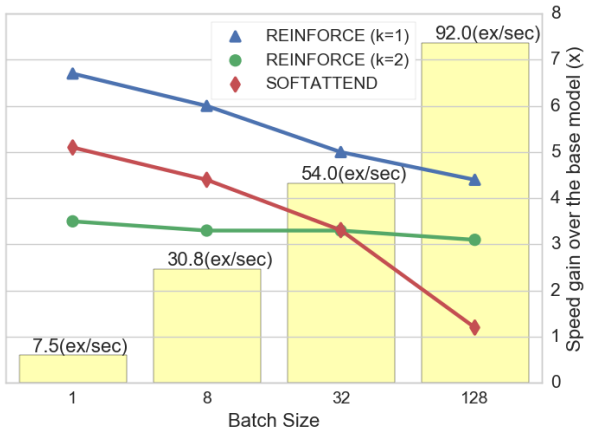


Figure 4: Runtime for document encoding on an Intel Xeon CPU E5-1650 @3.20GHz on WIKIREADING at test time. The boxplot represents the throughput of BASE and each line plot shows the proposed models' speed gain over BASE. Exact numbers are reported in the supplementary material.

We halved the learning rate every 25k steps. We use the Adam (Kingma and Ba, 2015) optimizer and TensorFlow framework (Abadi et al., 2015).

Evaluation Metrics Our main evaluation metric is answer accuracy, the proportion of questions answered correctly. For sentence selection, since we do not know which sentence contains the answer, we report approximate accuracy by matching sentences that contain the answer string (y^*). For the soft attention model, we treat the sentence with the highest probability as the predicted sentence.

Models and Baselines The models PIPELINE, REINFORCE, and SOFTATTEND correspond to the learning objectives in Section 5. We compare these models against the following baselines:

> **FIRST** always selects the first sentence of the document. The answer appears in the first sentence in 33% and 15% of documents in WIKISUGGEST and WIKIREADING LONG.
> **BASE** is the re-implementation of the best model by Hewlett et al. (2016), consuming the first 300 tokens. We experimented with providing additional tokens to match the length of document available to hierarchical models, but this performed poorly.[4]
> **ORACLE** selects the first sentence with the answer string if it exists, or otherwise the first sentence in the document.

[4]Our numbers on WIKIREADING outperform previously reported numbers due to modifications in implementation and better optimization.

Dataset	Learning	Accuracy
WIKIREADING LONG	FIRST	26.7
	BASE	40.1
	ORACLE	43.9
	PIPELINE	36.8
	SOFTATTEND	38.3
	REINFORCE (K=1)	40.1
	REINFORCE (K=2)	**42.2**
WIKI SUGGEST	FIRST	44.0
	BASE	**46.7**
	ORACLE	60.0
	PIPELINE	45.3
	SOFTATTEND	45.4
	REINFORCE (K=1)	45.4
	REINFORCE (K=2)	45.8
WIKIREADING	FIRST	71.0
	HEWLETT ET AL. (2016)	71.8
	BASE	**75.6**
	ORACLE	74.6
	SOFTATTEND	71.6
	PIPELINE	72.4
	REINFORCE (K=1)	73.0
	REINFORCE (K=2)	73.9

Table 3: Answer prediction accuracy on the test set. K is the number of sentences in the document summary.

Answer Accuracy Results Table 3 summarizes answer accuracy on all datasets. We use BOW encoder for sentence selection as it is the fastest. The proposed hierarchical models match or exceed the performance of BASE, while reducing the number of RNN steps significantly, from 300 to 35 (or 70 for K=2), and allowing access to later parts of the document. Figure 4 reports the speed gain of our system. While throughput at training time can be improved by increasing the batch size, at test time real-life QA systems use batch size 1, where REINFORCE obtains a 3.5x-6.7x speedup (for K=2 or K=1). In all settings, REINFORCE was at least three times faster than the BASE model.

All models outperform the FIRST baseline, and utilizing the proxy oracle sentence (ORACLE) improves performance on WIKISUGGEST and WIKIREADNG LONG. In WIKIREADING, where the proxy oracle sentence is often missing and documents are short, BASE outperforms ORACLE.

Jointly learning answer generation and sentence selection, REINFORCE outperforms PIPELINE, which relies on a noisy supervision signal for sentence selection. The improvement is larger in WIKIREADING LONG, where the approximate supervision for sentence selection is missing for 51% of examples compared to 22% of examples in WIKISUGGEST.[5]

On WIKIREADING LONG, REINFORCE outper-

[5]The number is lower than in Table 1 because we cropped sentences and documents, as mentioned above.

Dataset	Learning	Model	Accuracy
WIKI READING LONG	PIPELINE	CNN	70.7
		BOW	69.2
		CHUNKBOW	**74.6**
	REINFORCE	CNN	74.2
		BOW	72.2
		CHUNKBOW	**74.4**
	FIRST		31.3
	SOFTATTEND (BOW)		70.1
WIKI SUGGEST	PIPELINE	CNN	62.3
		BOW	**67.5**
		CHUNKBOW	57.4
	REINFORCE	CNN	64.6
		BOW	**67.3**
		CHUNKBOW	59.3
	FIRST		42.6
	SOFTATTEND (BOW)		49.9

Table 4: Approximate sentence selection accuracy on the development set for all models. We use ORACLE to find a proxy gold sentence and report the proportion of times each model selects the proxy sentence.

forms all other models (excluding ORACLE, which has access to gold labels at test time). In other datasets, BASE performs slightly better than the proposed models, at the cost of speed. In these datasets, the answers are concentrated in the first few sentences. BASE is advantageous in categorical questions (such as GENDER), gathering bits of evidence from the whole document, at the cost of speed. Encouragingly, our system almost reaches the performance of ORACLE in WIKIREADING, showing strong results in a limited token setting.

Sampling an additional sentence into the document summary increased performance in all datasets, illustrating the flexibility of hard attention compared to soft attention. Additional sampling allows recovery from mistakes in WIKIREADING LONG, where sentence selection is challenging.[6] Comparing hard attention to soft attention, we observe that REINFORCE performed better than SOFTATTEND. The attention distribution learned by the soft attention model was often less peaked, generating noisier summaries.

Sentence Selection Results Table 4 reports sentence selection accuracy by showing the proportion of times models selects the proxy gold sentence when it is found by ORACLE. In WIKIREADING LONG, REINFORCE finds the approximate gold sentence in 74.4% of the examples where the the answer is in the document. In WIKISUGGEST performance is at 67.5%, mostly due to noise in the data. PIPELINE performs slightly better as it is directly trained towards our noisy evaluation. However, not all sentences that contain the answer are useful to answer the question (first example in Table 6). REINFORCE learned to choose sentences that are likely to generate a correct answer rather than proxy gold sentences, improving the final answer accuracy. On WIKIREADING LONG, complex models (CNN and CHUNKBOW) outperform the simple BOW, while on WIKISUGGEST BOW performed best.

[6] Sampling more help pipeline methods less.

	WR LONG	WIKI SUGGEST
No evidence in doc.	29	8
Error in answer generation	13	15
Noisy query & answer	0	24
Error in sentence selection	8	3

Table 5: Manual error analysis on 50 errors from the development set for REINFORCE (K=1).

Qualitative Analysis We categorized the primary reasons for the errors in Table 5 and present an example for each error type in Table 6. All examples are from REINFORCE with BOW sentence selection. The most frequent source of error for WIKIREADING LONG was lack of evidence in the document. While the dataset does not contain false answers, the document does not always provide supporting evidence (examples of properties without clues are ELEVATION ABOVE SEA LEVEL and SISTER). Interestingly, the answer string can still appear in the document as in the first example in Table 6: 'Saint Petersburg' appears in the document (4th sentence). Answer generation at times failed to generate the answer even when the correct sentence was selected. This was pronounced especially in long answers. For the automatically collected WIKISUGGEST dataset, noisy question-answer pairs were problematic, as discussed in Section 3. However, the models frequently guessed the spurious answer. We attribute higher proxy performance in sentence selection for WIKISUGGEST to noise. In manual analysis, sentence selection was harder in WIKIREADING LONG, explaining why sampling two sentences improved performance.

In the first correct prediction (Table 6), the model generates the answer, even when it is not in the document. The second example shows when our model spots the relevant sentence without obvious clues. In the last example the model spots a sentence far from the head of the document.

Figure 5 contains a visualization of the atten-

WIKIREADING LONG (WR LONG)	Error Type (Query, Answer) System Output		No evidence in doc. (place_of_death, Saint Petersburg) Crimean Peninsula
	1 4 25	11.7 3.4 **63.6**	Alexandrovich Friedmann (also spelled Friedman or [Fridman] , Russian : . . . Friedmann was baptized . . . and lived much of his life in Saint Petersburg . Friedmann died on September 16 , 1925 , at the age of 37 , from typhoid fever that he contracted while returning from a vacation in Crimean Peninsula .
	Error Type (Query, Answer) System Output		Error in sentence selection (position_played_on_team_speciality, power forward) point guard
	1 3	**37.8** 22.9	James Patrick Johnson (born February 20 , 1987) is an American professional basketball player for the Toronto Raptors of the National Basketball Association (NBA). Johnson was the starting power forward for the Demon Deacons of Wake Forest University
WIKISUGGEST (WS)	Error Type (Query, Answer) System Output		Error in answer generation (david blaine's mother, Patrice Maureen White) Maureen
	1 8	14.1 **22.6**	David Blaine (born David Blaine White; April 4, 1973) is an American magician, illusionist . . . Blaine was born and raised in, Brooklyn , New York the son of Patrice Maureen White . . .
	Error Type (Query, Answer) System Output		Noisy query & answer (what are dried red grapes called, dry red wines) Chardonnay
	1 2	2.8 **90.8**	Burgundy wine (French : Bourgogne or vin de Bourgogne) is wine made in the . . . The most famous wines produced here . . . are dry red wines made from Pinot noir grapes . . .

Correctly Predicted Examples

WR LONG	(Query, Answer)		(position_held, member of the National Assembly of South Africa)
	1	**98.4**	Anchen Margaretha Dreyer (born 27 March 1952) is a South African politician, a Member of Parliament for the opposition Democratic Alliance , and currently . . .
	(Query, Answer)		(headquarters_locations, Solihull)
	1 4	13.8 **82.3**	LaSer UK is a provider of credit and loyalty programmes , operating in the UK and Republic . . . The company 's operations are in Solihull and Belfast where it employs 800 people .
WS	(Query, Answer)		(avril lavigne husband, Chad Kroeger)
	1 23	17.6 **68.4**	Avril Ramona Lavigne ([vrɬl] [lvin] / ; French pronunciation : ¡200b¿ ([avil] [lavi]) ;. . . Lavigne married Nickelback frontman , Chad Kroeger , in 2013 . Avril Ramona Lavigne was . . .

Table 6: Example outputs from REINFORCE (K=1) with BOW sentence selection model. First column: sentence index (l). Second column: attention distribution $p_\theta(s_l|d,x)$. Last column: text s_l.

tion distribution over sentences, $p(s_l \mid d, x)$, for different learning procedures. The increased frequency of the answer string in WIKISUGGEST vs. WIKIREADING LONG is evident in the leftmost plot. SOFTATTEND and CHUNKBOW clearly distribute attention more evenly across the sentences compared to BOW and CNN.

7 Related Work

There has been substantial interest in datasets for reading comprehension. MCTest (Richardson et al., 2013) is a smaller-scale datasets focusing on common sense reasoning; bAbi (Weston et al., 2015) is a synthetic dataset that captures various aspects of reasoning; and SQuAD (Rajpurkar et al., 2016; Wang et al., 2016; Xiong et al., 2016) and NewsQA (Trischler et al., 2016a) are QA datasets where the answer is a span in the document. Compared to Wikireading, some datasets covers shorter passages (average 122 words for SQuAD). Cloze-style question answering datasets (Hermann et al., 2015; Onishi et al., 2016; Hill et al., 2015) assess machine comprehension but do not form questions. The recently released MS MARCO dataset (Nguyen et al., 2016) consists of query logs, web documents and crowd-sourced answers.

Answer sentence selection is studied with the TREC QA (Voorhees and Tice, 2000), WikiQA (Yang et al., 2016b) and SelQA (Jurczyk et al., 2016) datasets. Recently, neural networks models (Wang and Nyberg, 2015; Severyn and

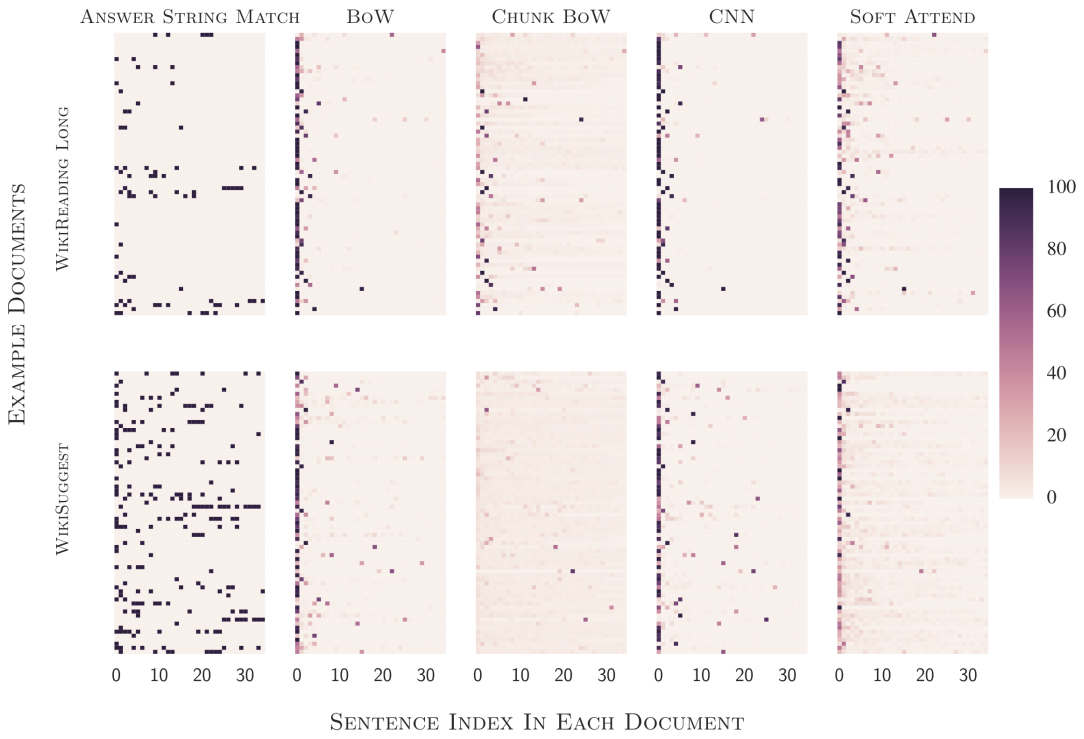


Figure 5: For a random subset of documents in the development set, we visualized the learned attention over the sentences ($p(s_l|d, x)$).

Moschitti, 2015; dos Santos et al., 2016) achieved improvements on TREC datsaet. Sultan et al. (2016) optimized the answer sentence extraction and the answer extraction jointly, but with gold labels for both parts. Trischler et al. (2016b) proposed a model that shares the intuition of observing inputs at multiple granularities (sentence, word), but deals with multiple choice questions. Our model considers answer sentence selection as latent and generates answer strings instead of selecting text spans, and we found that WIKIREADING dataset suits our purposes best with some pruning, which still provided 1.97 million examples compared to 2K questions for TREC dataset.

Hierarchical models which treats sentence selection as a latent variable have been applied text categorization (Yang et al., 2016b), extractive summarization (Cheng and Lapata, 2016), machine translation (Ba et al., 2014) and sentiment analysis (Yessenalina et al., 2010; Lei et al., 2016). To the best of our knowledge, we are the first to use the hierarchical nature of a document for QA.

Finally, our work is related to the reinforcement learning literature. Hard and soft attention were examined in the context of caption generation (Xu et al., 2015). Curriculum learning was investigated in Sachan and Xing (2016), but they focused on the ordering of training examples while we combine supervision signals. Reinforcement learning recently gained popularity in tasks such as coreference resolution (Clark and Manning, 2016), information extraction (Narasimhan et al., 2016), semantic parsing (Andreas et al., 2016) and textual games (Narasimhan et al., 2015; He et al., 2016).

8 Conclusion

We presented a coarse-to-fine framework for QA over long documents that quickly focuses on the relevant portions of a document. In future work we would like to deepen the use of structural clues and answer questions over multiple documents, using paragraph structure, titles, sections and more. Incorporating coreference resolution would be another important direction for future work. We argue that this is necessary for developing systems that can efficiently answer the information needs of users over large quantities of text.

Acknowledgement

We appreciate feedbacks from Google colleagues. We also thank Yejin Choi, Kenton Lee, Mike Lewis, Mark Yatskar and Luke Zettlemoyer for comments on the earlier draft of the paper. The last author is partially supported by Israel Science Foundation, grant 942/16.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. http://tensorflow.org/.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* .

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2014. Multiple object recognition with visual attention. *The International Conference on Learning Representations* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of the International Conference on Learning Representations* .

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Association for Computational Linguistics*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *Proceedings of the Annual Meeting of the Association for Computational Linguistics* .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Proceedings of the Conference of the Empirical Methods in Natural Language Processing* .

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the Conference of the Empirical Methods in Natural Language Processing*.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)* 12:2493–2537.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR* abs/1602.03609.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with an unbounded action space. *Proceedings of the Conference of the Association for Computational Linguistics* .

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. http://arxiv.org/abs/1506.03340.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A novel large-scale language understanding task over wikipedia. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *The International Conference on Learning Representations* .

Tomasz Jurczyk, Michael Zhai, and Jinho D. Choi. 2016. SelQA: A New Benchmark for Selection-based Question Answering. In *Proceedings of the 28th International Conference on Tools with Artificial Intelligence*. San Jose, CA, ICTAI'16. https://arxiv.org/abs/1606.08513.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 908–918. http://www.aclweb.org/anthology/P16-1086.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proceedings of the Annual Meeting of the Association for Computational Linguistics* .

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proceedings of the Conference of the Empirical Methods in Natural Language Processing* .

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations* .

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the International Conference on Machine Learning*.

Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2016. Rationalizing neural predictions. *Proceedings of the Conference of the Empirical Methods in Natural Language Processing* .

Michael EJ Masson. 1983. Conceptual processing of text during skimming and rapid sequential reading. *Memory & Cognition* 11(3):262–274.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *Proceedings of the Conference of the Empirical Methods in Natural Language Processing* .

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *Proceedings of the Conference of the Empirical Methods in Natural Language Processing* .

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. *Proceedings of the Conference of the Empirical Methods in Natural Language Processing* .

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Workshop in Advances in Neural Information Processing Systems*.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. *Proceedings of Empirical Methods in Natural Language Processing* .

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference of the Empirical Methods in Natural Language Processing*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the Conference of the Empirical Methods in Natural Language Processing*.

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*.

Mrinmaya Sachan and Eric P Xing. 2016. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 373–382.

Md. Arafat Sultan, Vittorio Castelli, and Radu Florian. 2016. A joint model for answer sentence ranking and answer extraction. *Transactions of the Association for Computational Linguistics* 4:113–125.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016a. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830* .

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. 2016b. A parallel-hierarchical model for machine comprehension on sparse data. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* .

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 200–207.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* .

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *Proceedings of the International Conference on Machine Learning* .

Yi Yang, Wen-tau Yih, and Christopher Meek. 2016a. Wikiqa: A challenge dataset for open-domain question answering. *Proceedings of the Conference of the Empirical Methods in Natural Language Processing* .

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016b. Hierarchical attention networks for document classification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1046–1056.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. In *NIPS Deep Learning Workshop*. http://arxiv.org/abs/1412.1632.

An End-to-End Model for Question Answering over Knowledge Base with Cross-Attention Combining Global Knowledge

Yanchao Hao[1,2], **Yuanzhe Zhang**[1,3], **Kang Liu**[1], **Shizhu He**[1], **Zhanyi Liu**[3], **Hua Wu**[3] and **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Baidu Inc., Beijing, 100085, China
{yanchao.hao, yzzhang, kliu, shizhu.he, jzhao}@nlpr.ia.ac.cn
{liuzhanyi, wu_hua}@baidu.com

Abstract

With the rapid growth of knowledge bases (KBs) on the web, how to take full advantage of them becomes increasingly important. Question answering over knowledge base (KB-QA) is one of the promising approaches to access the substantial knowledge. Meanwhile, as the neural network-based (NN-based) methods develop, NN-based KB-QA has already achieved impressive results. However, previous work did not put more emphasis on question representation, and the question is converted into a fixed vector regardless of its candidate answers. This simple representation strategy is not easy to express the proper information in the question. Hence, we present an end-to-end neural network model to represent the questions and their corresponding scores dynamically according to the various candidate answer aspects via cross-attention mechanism. In addition, we leverage the global knowledge inside the underlying KB, aiming at integrating the rich KB information into the representation of the answers. As a result, it could alleviates the out-of-vocabulary (OOV) problem, which helps the cross-attention model to represent the question more precisely. The experimental results on WebQuestions demonstrate the effectiveness of the proposed approach.

1 Introduction

As the amount of the knowledge bases (KBs) grows, people are paying more attention to seeking effective methods for accessing these precious intellectual resources. There are several tailor-made languages designed for querying KBs, such as SPARQL (Prudhommeaux and Seaborne, 2008). However, to handle such query languages, users are required to not only be familiar with the particular language grammars, but also be aware of the architectures of the KBs. By contrast, knowledge base-based question answering (KB-QA) (Unger et al., 2014), which takes natural language as query language, is a more user-friendly solution, and has become a research focus in recent years.

Given natural language questions, the goal of KB-QA is to automatically return answers from the KB. There are two mainstream research directions for this task: semantic parsing-based (SP-based) (Zettlemoyer and Collins, 2009, 2012; Kwiatkowski et al., 2013; Cai and Yates, 2013; Berant et al., 2013; Yih et al., 2015, 2016; Reddy et al., 2016) and information retrieval-based (IR-based) (Yao and Van Durme, 2014; Bordes et al., 2014a,b, 2015; Dong et al., 2015; Xu et al., 2016a,b) methods. SP-based methods usually focus on constructing a semantic parser that could convert natural language questions into structured expressions like logical forms. IR-based methods usually search answers from the KB based on the information conveyed in questions, where ranking techniques are often adopted to make correct selections from candidate answers.

Recently, with the progress of deep learning, neural network-based (NN-based) methods have been introduced to the KB-QA task (Bordes et al., 2014b). Different from previous methods, NN-based methods represent both of the questions and the answers as semantic vectors. Then the complex process of KB-QA could be converted into a similarity matching process between an input question and its candidate answers in a semantic space. The candidates with the highest similarity score will be selected as the final answers. Because they are more adaptive, NN-based methods have attracted more and more attention, and this

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 221–231
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1021

paper also focuses on using end-to-end neural networks to answer questions over knowledge base.

In NN-based methods, the crucial step is to compute the similarity score between a question and a candidate answer, where the key is to learn their representations. Previous methods put more emphasis on learning representation of the answer end. For example, Bordes et al. (2014a) consider the importance of the subgraph of the candidate answer. Dong et al. (2015) make use of the context and the type of the answer. However, the representation of the question end is oligotrophic. Existing approaches often represent a question into a single vector using simple bag-of-words (BOW) model (Bordes et al., 2014a,b), whereas the relatedness to the answer end is neglected. We argue that a question should be represented differently according to the different focuses of various answer aspects[1].

Take the question "Who is the president of France?" and one of its candidate answers "Francois Hollande" as an example. When dealing with the answer entity `Francois Holland`, "president" and "France" in the question is more focused, and the question representation should bias towards the two words. While facing the answer type `/business/board_member`, "Who" should be the most prominent word. Meanwhile, some questions may value answer type more than other answer aspects. While in some other questions, answer relation may be the most important information we should consider, which is dynamic and flexible corresponding to different questions and answers. Obviously, this is an attention mechanism, which reveals the mutual influences between the representation of questions and the corresponding answer aspects.

We believe that such kind of representation is more expressive. Dong et al. (2015) represents questions using three CNNs with different parameters when dealing with different answer aspects including answer path, answer context and answer type. The method is very enlightening and achieves the best performance on WebQeustions at that time among the end-to-end approaches. However, we argue that simply selecting three independent CNNs is mechanical and inflexible. Thus, we go one step further, and propose a cross-attention based neural network to perform KB-QA. The cross-attention model, which stands for the mutual attention between the question and the answer aspects, contains two parts: the answer-towards-question attention part and the question-towards-answer attention part. The former help learn flexible and adequate question representation, and the latter help adjust the question-answer weight, getting the final score. We illustrate in section 3.2 for more details. In this way, we formulate the cross-attention mechanism to model the question answering procedure. Note that our proposed model is an entire end-to-end approach which only depends on training data. Some integrated systems which use extra patterns and resources are not directly comparable to ours. Our target is to explore a better solution following the end-to-end KB-QA technical path.

Moreover, we notice that the representations of the KB resources (entities and relations) are also limited in previous work. specifically, they are often learned barely on the QA training data, which results in two limitations. 1) The global information of the KB is deficient. For example, if question-answer pair (q, a) appears in the training data, and the global KB information implies us that a' is similar to a[2], denoted by (a $\sim a'$), then (q, a') is more probable to be right. However, current QA training mechanism cannot guarantee (a $\sim a'$) could be learned. 2) The problem of out-of-vocabulary (OOV) stands out. Due to the limited coverage of the training data, the OOV problem is common while testing, and many answer entities in testing candidate set have never been seen before. The attention of these resources become the same because they shared the same OOV embedding, and this will do harm to the proposed attention model. To tackle these two problems, we additionally incorporates KB itself as training data for training embeddings besides original question-answer pairs. In this way, the global structure of the whole knowledge could be captured, and the OOV problem could be alleviated naturally.

In summary, the contributions are as follows.
1) We present a novel cross-attention based NN model tailored to KB-QA task, which considers the mutual influence between the representation of questions and the corresponding answer aspects.
2) We leverage the global KB information, aiming at represent the answers more precisely. It also al-

[1] An answer aspect could be the answer entity itself, the answer type, the answer context, etc.

[2] The complete KB is able to offer this kind of information, e.g., a and a' share massive context.

leviates the OOV problem, which is very helpful to the cross-attention model.
3) The experimental results on the open dataset WebQuestions demonstrate the effectiveness of the proposed approach.

2 Overview

The goal of KB-QA task could be formulated as follows. Given a natural language question q, the system returns an entity set A as answers. The architecture of our proposed KB-QA system is shown in Figure 1, which illustrates the basic flow of our approach. First, we identify the topic entity of the question, and generate candidate answers from Freebase. Then, a cross-attention based neural network is employed to represent the question under the influence of the candidate answer aspects. Finally, the similarity score between the question and each corresponding candidate answer is calculated, and the candidates with highest score will be selected as the final answers[3].

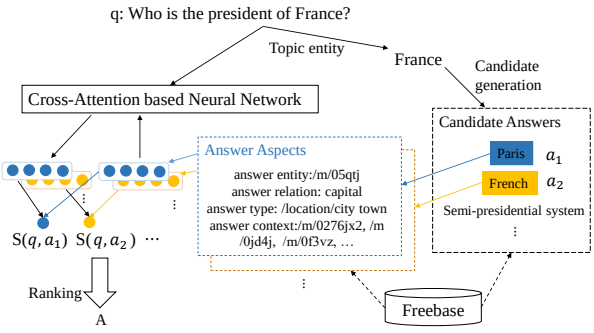


Figure 1: The overview of the proposed KB-QA system.

We utilize Freebase (Bollacker et al., 2008) as our knowledge base. It has more than 3 billion facts, and is used as the supporting KB for many QA tasks. In Freebase, the facts are represented by subject-predicate-object triples (s, p, o). For clarity, we call each basic element a resource, which could be either an entity or a relation. For example, `(/m/0f8l9c, location.country.capital, /m/05qtj)`[4] describes the fact that the capital of France is Paris, where `/m/0f8l9c` and `/m/05qtj` are entities denoting `France` and `Paris` respectively, and `location.country.capital` is a relation.

3 Our Approach

3.1 Candidate Generation

All the entities in Freebase should be candidate answers ideally, but in practice, this is time consuming and not really necessary. For each question q, we use Freebase API (Bollacker et al., 2008) to identify a topic entity, which could be simply understood as the main entity of the question. For example, France is the topic entity of question "Who is the president of France?". Freebase API method is able to resolve as many as 86% questions if we use the top1 result (Yao and Van Durme, 2014). After getting the topic entity, we collect all the entities directly connected to it and the ones connected with 2-hop[5]. These entities constitute a candidate set C_q .

3.2 The Neural Cross-Attention Model

We present a cross-attention based neural network, which represents the question dynamically according to different answer aspects, also considering their connections. Concretely, each aspect of the answer focuses on different words of the question and thus decides how the question is represented. Then the question pays different attention to each answer aspect to decide their weights. Figure 2 is the architecture of our model. We will illustrate how the system works as follows.

3.2.1 Question Representation

First of all, we have to obtain the representation of each word in the question. These representations retain all the information of the question, and could serve the following steps. Suppose question q is expressed as $q = (x_1, x_2, ..., x_n)$, where x_i denotes the ith word. As shown in Figure 2, we first look up a word embedding matrix $E_w \in \mathbb{R}^{d \times v_w}$ to get the word embeddings, which is randomly initialized, and updated during the training process. Here, d means the dimension of the embeddings and v_w denotes the vocabulary size of natural language words.

Then, the embeddings are fed into a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks. LSTM has been proven to

[3]We also adopt a margin strategy to obtain multiple answers for a question and this will be explained in the next section.

[4]Note that the Freebase prefixes are omitted for neatness.

[5]For example, (/m/0f8l9c, governing_officials, government.position_held.office_holder, /m/02qg4z) is a 2-top connection.

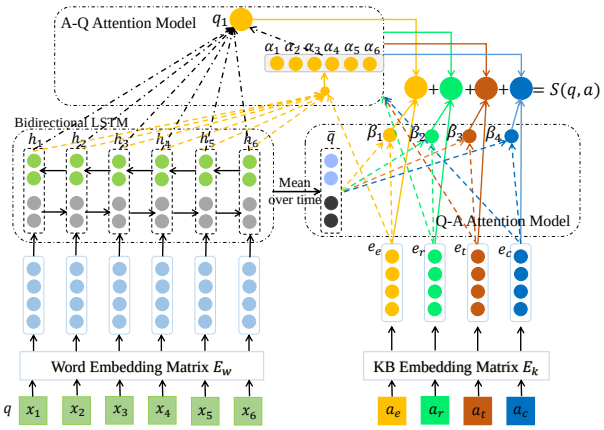


Figure 2: The architecture of the proposed cross-attention based neural network. Note that only one aspect(in orange color) is depicted for clarity. The other three aspects follow the same way.

be effective in many natural language processing (NLP) tasks such as machine translation (Sutskever et al., 2014) and dependency parsing (Dyer et al., 2015), and it is adept in harnessing long sentences. Note that if we use unidirectional LSTM, the outcome of a specific word contains only the information of the words before it, whereas the words after it are not taken into account. To avoid this, we employ bidirectional LSTM as Bahdanau (2015) does, which consists of both forward and backward networks. The forward LSTM handles the question from left to right, and the backward LSTM processes in the reverse order. Thus, we could acquire two hidden state sequences, one from the forward one $(\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_n})$ and the other from the backward one $(\overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_n})$. We concatenate the forward hidden state and the backward hidden state of each word, resulting in $h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$. The hidden unit of forward and backward LSTM is $\frac{d}{2}$, so the concatenated vector is of dimension d. In this way, we obtain the representation of each word in the question.

3.2.2 Answer aspect representation

We directly use the embedding for each answer aspect through the KB embedding matrix $E_k \in \mathbb{R}^{d \times v_k}$. Here, v_k means the vocabulary size of the KB resources. The embedding matrix is randomly initialized and learned during training, and could be further enhanced with the help of global information as described in Section 3.3. Concretely, we employ four kinds of answer aspects: answer entity a_e, answer relation a_r, answer type a_t and answer context a_c[6]. Their embeddings are denoted as e_e, e_r, e_t and e_c, respectively. It is worth noting that the answer context consists of multiple KB resources, and we denote it as $(c_1, c_2, ..., c_m)$. We first acquire their KB embeddings $(e_{c_1}, e_{c_2}, ..., e_{c_m})$ through E_k, then calculate an average embedding by $e_c = \frac{1}{m}\sum_{i=1}^{m} e_{c_i}$.

3.2.3 Cross-Attention model

The most crucial part of the proposed approach is the cross-attention mechanism. The cross-attention mechanism is composed of two parts: the answer-towards-question attention part and the question-towards-answer attention part.

The proposed cross-attention model could also be intuitively interpreted as a re-reading mechanism (Hermann et al., 2015). Our aim is to select correct answers from a candidate set. When we judge a candidate answer, suppose we first look at its type, and we will reread the question to find out which part of the question should be more focused (handling attention). Then we go to next aspect and reread the question again, until the all the aspects are utilized. After we read all the answer aspects and get all the scores, the final similarity score between question and answer should be a weighted sum of all the scores. We believe that this mechanism is beneficial for the system to better understand the question with the help of the answer aspects, and it may lead to a performance promotion.

• Answer-towards-question(A-Q) attention

Based on our assumption, each answer aspect should focus on different words of the same question. The extent of attention can be measured by the relatedness between each word representation h_j and an answer aspect embedding e_i. We propose the following formulas to calculate the weights.

$$\alpha_{ij} = \frac{\exp(\omega_{ij})}{\sum_{k=1}^{n} \exp(\omega_{ik})} \quad (1)$$

$$\omega_{ij} = f(W^T[h_j; e_i] + b) \quad (2)$$

Here, α_{ij} denotes the weight of attention from answer aspect e_i to the jth word in the question, where $e_i \in \{e_e, e_r, e_t, e_c\}$. $f(\cdot)$ is a non-linear activation function, such as hyperbolic tangent transformation here. Let n be the length of the question. $W \in \mathbb{R}^{2d \times d}$ is an intermediate matrix and b

[6]Answer context is the 1-hop entities and predicates which connect to the answer entity along the answer path.

is the offset. Both of them are randomly initialized and updated during training. Subsequently, according to the specific answer aspect e_i, the attention weights are employed to calculate a weighted sum of the hidden representations, resulting in a semantic vector that represent the question.

$$q_i = \sum_{j=1}^{n} \alpha_{ij} h_j \tag{3}$$

The similarity score of the question q and this particular candidate answer aspect e_i ($e_i \in \{e_e, e_r, e_t, e_c\}$) could be defined as follows.

$$S(q, e_i) = h(q_i, e_i) \tag{4}$$

The scoring function $h(\cdot)$ is computed as the inner product between the sentence representation q_i, which has already carried the attention from the answer aspect part, and the corresponding answer aspect e_i, and is parametrized into the network and updated during the training process.

• **Question-towards-answer(Q-A) attention**

Intuitively, different question should value the four answer aspect differently. Since we have already calculated the scores of (q, e_i), we define the final similarity score of the question q and each candidate answer a as follows.

$$S(q, a) = \sum_{e_i \in \{e_e, e_r, e_t, e_c\}} \beta_{e_i} S(q, e_i) \tag{5}$$

$$\beta_{e_i} = \frac{\exp(\omega_{e_i})}{\sum_{e_k \in \{e_e, e_r, e_t, e_c\}} \exp(\omega_{e_k})} \tag{6}$$

$$\omega_{e_i} = f\left(W^T [\overline{q}; e_i] + b\right) \tag{7}$$

$$\overline{q} = \frac{1}{n} \sum\nolimits_{j}^{n} h_j \tag{8}$$

Here β_{e_i} denotes the attention of question towards answer aspects, indicating which answer aspect should be more focused in one (q, a) pair. $W \in \mathbb{R}^{2d \times d}$ is also a intermediate matrix as in the answer-towards-question attention part, and b is an offset value.[7] $\overline{q}$ is calculated by averagely pooling all the bi-directional LSTM hidden state sequences, resulting a vector which represents the question to determine which answer aspect should be more focused.

[7]Note that the W and b in the two attention part is different and independent.

3.2.4 Training

We first construct the training data. Since we have (q, a) pairs as supervision data, candidate set C_q can be divided into two subsets, namely, correct answer set P_q and wrong answer set N_q. For each correct answer $a \in P_q$, we randomly select k wrong answers $a' \in N_q$ as negative examples. For some topic entities, there may be not enough wrong answers to acquire k wrong answers. Under this circumstance, we extend N_q from other randomly selected candidate set C'_q. With the generated training data, we are able to make use of pairwise training. The training loss is given as follows, which is a hinge loss.

$$L_{q,a,a'} = [\gamma + S(q, a') - S(q, a)]_+ \tag{9}$$

where γ is a positive real number that ensures a margin between positive and negative examples. And $[z]_+$ means $max(0, z)$. The intuition of this training strategy is to guarantee the score of positive question-answer pairs to be higher than negative ones with a margin. The objective function is as follows.

$$\min \sum_{q} \frac{1}{|P_q|} \sum_{a \in P_q} \sum_{a' \in N_q} L_{q,a,a'} \tag{10}$$

We adopt stochastic gradient descent (SGD) to minimize the learning process, shuffled mini-batches are utilized.

3.2.5 Inference

In testing stage, given the candidate answer set C_q, we have to calculate $S(q, a)$ for each $a \in C_q$, and find out the maximum value S_{max}.

$$S_{\max} = \arg\max_{a \in C_q} \{S(q, a)\} \tag{11}$$

It is worth noting that many questions have more than one answer, so it is improper to set the candidate answer which have the maximum value as the final answer. Instead, we take advantage of the margin γ. If the score of a candidate answer is within the margin compared with S_{max}, we put it in the final answer set.

$$A = \{\hat{a} | S_{\max} - S(q, \hat{a}) < \gamma\} \tag{12}$$

3.3 Combining Global Knowledge

In this section, we elaborate how the global information of a KB could be leveraged. As stated before, we try to take into account the complete knowledge information of the KB. To this

end, we adopt TransE model (Bordes et al., 2013) and integrate its outcome into our training process. In TransE, relations are considered as translations in the embedding space. For consistency, we denote each fact as (s, p, o). TransE utilizes pairwise training strategy as well. Randomly sampled corrupted facts (s', p, o') are the negative examples. The distance measure $d(s + p, o)$ is defined as $\|s + p - o\|_2^2$. And the training loss is given as follows.

$$L_k = \sum_{(s,p,o)\in S} \sum_{(s',p,o')\in S'} [\gamma_k + d(s + p, o) - d(s' + p, o')]_+ \tag{13}$$

Where S is the set of KB facts and S' is the corrupted facts. In our QA task, we filter out the completely unrelated facts to save time. Specifically, we first collect all the topic entities of all the questions as initial set. Then, we expand the set by adding directly connected and 2-hop entities. Finally, all the facts containing these entities form the positive set, and the negative facts are randomly corrupted. This is a compromising solution due to the large scale of Freebase. To employ the global information in our training process, we adopt a multi-task training strategy. Specifically, we perform KB-QA training and TransE training in turn. The proposed training process ensures that the global KB information acts as additional supervision, and the interconnections among the resources are fully considered. In addition, as more KB resources are involved, the OOV problem is relieved. Since all the OOV resources have exactly the same attention towards a question, it will weaken the effectiveness of the attention model. So the alleviation of OOV is able to bring additional benefits to the attention model.

4 Experiments

To evaluate the proposed method, we conduct experiments on WebQuestions (Berant et al., 2013) dataset that includes 3,778 question-answer pairs for training and 2,032 for testing. The questions are collected from Google Suggest API, and the answers are labeled manually by Amazon MTurk. All the answers are from Freebase. We use three-quarter of the training data as training set, and the left as validate set. We use F_1 score as evaluation matric, and the average result is computed by the script provided by Berant et al. (2013).

Note that our proposed approach is an entire end-to-end method, which totally depends on training data. It is worth noting that Yih et al. (2015; 2016) achieve much higher F_1 scores than other methods. Their staged system is able to address more questions with constraints and aggregations. However, their approach applies numbers of manually designed rules and features, which come from the observations on the training set questions. These particular manual efforts reduce the adaptability of their approach. Moreover, there are some integrated systems such as Xu et al. (2016a; 2016b) achieve higher F_1 scores which leverage Wikipedia free text as external knowledge, so their systems are not directly comparable to ours.

4.1 Settings

For KB-QA training, we use mini-batch stochastic gradient descent to minimize the pairwise training loss. The minibatch size is set to 100. The learning rate is set to 0.01. Both the word embedding matrix E_w and KB embedding matrix E_v are normalized after each epoch. The embedding size d = 512, then the hidden unit size is 256. Margin γ is set to 0.6. Negative example number k = 2000. We set the embedding dimension to 512 in TransE training process, and the minibatch size is also 100. γ_k is set to 1. All these hyperparameters of the proposed network is determined according to the performance on the validate set.

4.2 Results

The effectiveness of the proposed approach

To demonstrate the effectiveness of the proposed approach, we compare our method with state-of-the-art end-to-end NN-based methods.

Methods	Avg F_1
Bordes et al., 2014b	29.7
Bordes et al., 2014a	39.2
Yang et al., 2014	41.3
Dong et al., 2015	40.8
Bordes et al., 2015	42.2
our approach	**42.9**

Table 1: The evaluation results on WebQuestions.

Table 1 shows the results on WebQuestions dataset. Bordes et al. (2014b) apply BOW method to obtain a single vector for both questions and answers. Bordes et al. (2014a) further improve their work by proposing the concept of subgraph embeddings. Besides the answer path, the sub-

graph contains all the entities and relations connected to the answer entity. The final vector is also obtained by bag-of-words strategy. Yang et al. (2014) follow the SP-based manner, but uses embeddings to map entities and relations into KB resources, then the question can be converted into logical forms. They jointly consider the two mapping processes. Dong et al. (2015) use three columns of Convolutional Neural Networks (CNNs) to represent questions corresponding to three aspects of the answers, namely the answer context, the answer path and the answer type. Bordes et al. (2015) put KB-QA into the memory networks framework (Sukhbaatar et al., 2015), and achieves the state-of-the-art performance of end-to-end methods. Our approach employs bidirectional LSTM, cross-attention model and global KB information.

From the results, we observe that our approach achieves the best performance of all the end-to-end methods on WebQuestions. Bordes et al. (2014b; 2014a; 2015) all utilize BOW model to represent the questions, while ours takes advantage of the attention of answer aspects to dynamically represent the questions. Also note that Bordes et al. (2015) uses additional training data such as Reverb (Fader et al., 2011) and their original dataset SimpleQuestions. Dong et al. (2015) employs three fixed CNNs to represent questions, while ours is able to express the focus of each unique answer aspect to the words in the question. Besides, our approach employs the global KB information. So, we believe that the results faithfully show that the proposed approach is more effective than the other competitive methods.

Model Analysis

In this part, we further discuss the impacts of the components of our model. Table 2 indicates the effectiveness of different parts in the model.

Methods	Avg F_1
LSTM	38.2
Bi_LSTM	39.1
Bi_LSTM+A-Q-ATT	41.6
Bi_LSTM+C-ATT	41.8
Bi_LSTM+GKI	40.4
Bi_LSTM+A-Q-ATT+GKI	42.6
Bi_LSTM+C-ATT+GKI	42.9

Table 2: The ablation results of our models.

LSTM employs unidirectional LSTM, and uses the last hidden state as the question representation. *Bi_LSTM* adopts a bidirectional LSTM. *A-Q-ATT* denotes the answer-towards-question attention part, and C-ATT stands for our cross-attention. *GKI* means global knowledge information. *Bi_LSTMS+C-ATT+GKI* is our full proposed approach. From the results, we could observe the following.

1) *Bi_LSTM+C-ATT* dramatically improves the F_1 score by 2.7 points compared with *Bi_LSTM*, 0.2 points higher than *Bi_LSTM+A-Q-ATT*. Similarly, *Bi_LSTM+C-ATT+GKI* significantly outperforms *Bi_LSTM+GKI* by 2.5 points, improving *Bi_LSTM+A-Q-ATT+GKI* by 0.3 points. The results prove that the proposed cross-attention model is effective.

2) *Bi_LSTM+GKI* performs better than *Bi_LSTM*, and achieves an improvement of 1.3 points. Similarly, *Bi_LSTM+C-ATT+GKI* improves *Bi_LSTM+C-ATT* by 1.1 points, which indicates that the proposed training strategy successfully leverages the global information of the underlying KB.

3) *Bi_LSTM+C-ATT+GKI* achieves the best performance as we expected, and improves the original *Bi_LSTM* dramatically by 3.8 points. This directly shows the power of the attention model and the global KB information.

To illustrate the effectiveness of the attention mechanism clearly, we present the attention weights of a question in the form of heat map as shown in Figure 3.

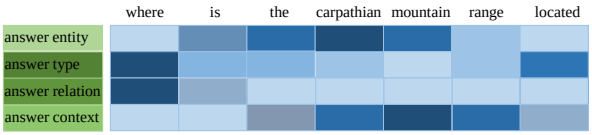


Figure 3: The visualized attention heat map. Answer entity: `/m/06npd(Slovakia)`, answer relation: `partially_containedby`, answer type: `/location/country`, answer context: `(/m/04dq9kf, /m/01mp, ...)`

From this example we observe that our methods is able to capture the attention properly. It is instructive to figure out the attention part of the question when dealing with different answer aspects. The heat map will help us understand which parts are most useful for selecting correct answers. For instance, from Figure 3, we can see that `location.country` is paying great attention

to "Where", indicating that "Where" is much more important than the other parts in the question when dealing with this type. In other words, the other parts are not that crucial since "Where" is strongly implying that the question is asking about a location. As for Q-A attention part, we see that answer type and answer relation are more important than other answer aspects in this example.

4.3 Error Analysis

We randomly sample 100 imperfectly answered questions and categorize the errors into two main classes as follows.

Wrong attention

In some occasions (18 in 100 questions, 18%), we find the generated attention weights unreasonable. For instance, for question "What are the songs that Justin Bieber wrote?", answer type `/music/composition` pays the most attention on "What" rather than "songs". We think this is due to the bias of the training data, and we believe these errors could be solved by introducing more instructive training data.

Complex questions and label errors

Another challenging problem is the complex questions (35%). For example, "When was the last time Knicks won the championship?" is actually to ask the last championship, but the predicted answers give all the championships. This is due to that the model cannot learn what "last" mean in the training process. In addition, the label mistakes also influence the evaluation (3%), such as, "What college did John Nash teach at?", where the labeled answer is `Princeton University`, but `Massachusetts Institute of Technology` should also be an answer, and the proposed method is able to answer it correctly. Other errors include topic entity generation error and the multiple answers error (giving more answers than expected). We guess these errors are caused by the simple implementations of the related steps in our method, and we will not explain them in detail.

5 Related Work

The past years have seen a growing amount of research on KB-QA, shaping an interaction paradigm that allows end users to profit from the expressive power of Semantic Web data while at the same time hiding their complexity behind an intuitive and easy-to-use interface. At the same time the growing amount of data has led to a heterogeneous data landscape where QA systems struggle to keep up with the volume, variety and veracity of the underlying knowledge.

5.1 Neural Network-based KB-QA

In recent years, deep neural networks have been applied to many NLP tasks, showing promising results. Bordes et al. (2014b) was the first to introduce NN-based method to solve KB-QA problem. The questions and KB triples were represented by vectors in a low dimensional space. Thus the cosine similarity could be used to find the most possible answer. BOW method was employed to obtain a single vector for both the questions and the answers. Pairwise training was utilized, and the negative examples were randomly selected from the KB facts. Bordes et al. (2014a) further improved their work by proposing the concept of subgraph embeddings. The key idea was to involve as much information as possible in the answer end. Besides the answer triple, the subgraph contained all the entities and relations connected to the answer entity. The final vector was also obtained by bag-of-words strategy.

Yih et al. (2014) focused on single-relation questions. The KB-QA task was divided into two steps. Firstly, they found the topic entity of the question. Then, the rest of the question was represented by CNNs and used to match relations. Yang et al. (2014) tackled entity and relation mapping as joint procedures. Actually, these two methods followed the SP-based manner, but they took advantage of neural networks to obtain intermediate mapping results.

The most similar work to ours is Dong et al. (2015). They considered the different aspects of answers, using three columns of CNNs to represent questions respectively. The difference is that our approach uses cross-attention mechanism for each unique answer aspect, so the question representation is not fixed to only three types. Moreover, we utilize the global KB information.

Xu et al. (2016a; 2016b) proposed integrated systems to address KB-QA problems incorporating Wikipedia free text, in which they used multi-channel CNNs to extract relations.

5.2 Attention-based Model

The attention mechanism has been widely used in different areas. Bahdanau et al. (2015) first applied attention model in NLP. They improved

the encoder-decoder Neural Machine Translation (NMT) framework by jointly learning align and translation. They argued that representing source sentence by a fixed vector is unreasonable, and proposed a soft-align method, which could be understood as attention mechanism. Rush et al. (2015) implemented sentence-level summarization task. They utilized local attention-based model that generated each word of the summary conditioned on the input sentence. Wang et al. (2016) proposed an inner attention mechanism that the attention was imposed directly to the input. And their experiment on answer selection showed the advantage of inner attention compared with traditional attention methods.

Yin et al. (2016) tackled simple question answering by an attentive convolutional neural network. They stacked an attentive max-pooling above convolution layer to model the relationship between predicates and question patterns. Our approach differs from previous work in that we use attentions to help represent questions dynamically, not generating current word from vocabulary as before.

6 Conclusion

In this paper, we focus on KB-QA task. Firstly, we consider the impacts of the different answer aspects when representing the question, and propose a novel cross-attention model for KB-QA. Specifically, we employ the focus of the answer aspects to each question word and the attention weights of the question towards the answer aspects. This kind of dynamic representation is more precise and flexible. Secondly, we leverage the global KB information, which could take full advantage of the complete KB, and also alleviate the OOV problem for the attention model. The extensive experiments demonstrate that the proposed approach could achieve better performance compared with state-of-the-art end-to-end methods.

Acknowledgments

This work was supported by the Natural Science Foundation of China (No.61533018) and the National Program of China (973 program No. 2014CB340505). And this research work was also supported by Google through focused research awards program. We would like to thank the anonymous reviewers for their useful comments and suggestions.

References

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

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1533–1544. http://aclweb.org/anthology/D13-1160.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, pages 1247–1250.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 615–620. https://doi.org/10.3115/v1/D14-1067.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075* .

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. pages 2787–2795.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 165–180.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 423–433. http://aclweb.org/anthology/P13-1042.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 260–269. https://doi.org/10.3115/v1/P15-1026.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and A. Noah Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 334–343. https://doi.org/10.3115/v1/P15-1033.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1535–1545. http://aclweb.org/anthology/D11-1142.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

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

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1545–1556. http://aclweb.org/anthology/D13-1161.

Eric Prudhommeaux and Andy Seaborne. 2008. Sparql query language for rdf. w3c recommendation, january 2008.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association of Computational Linguistics* 4:127–141. http://aclweb.org/anthology/Q16-1010.

M. Alexander Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 379–389. https://doi.org/10.18653/v1/D15-1044.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Christina Unger, André Freitas, and Philipp Cimiano. 2014. An introduction to question answering over linked data. In *Reasoning Web International Summer School*. Springer, pages 100–140.

Bingning Wang, Kang Liu, and Jun Zhao. 2016. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1288–1297. https://doi.org/10.18653/v1/P16-1122.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016b. Hybrid question answering over knowledge base and free text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 2397–2407. http://aclweb.org/anthology/C16-1226.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016a. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 2326–2336. https://doi.org/10.18653/v1/P16-1220.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 645–650. https://doi.org/10.3115/v1/D14-1071.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 956–966. https://doi.org/10.3115/v1/P14-1090.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1321–1331. https://doi.org/10.3115/v1/P15-1128.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 643–648. https://doi.org/10.3115/v1/P14-2105.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of

semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 201–206. https://doi.org/10.18653/v1/P16-2033.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 1746–1756. http://aclweb.org/anthology/C16-1164.

Luke Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, pages 976–984. http://aclweb.org/anthology/P09-1110.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420* .

Translating Neuralese

Jacob Andreas **Anca Dragan** **Dan Klein**
Computer Science Division
University of California, Berkeley
{jda,anca,klein}@cs.berkeley.edu

Abstract

Several approaches have recently been proposed for learning decentralized deep multiagent policies that coordinate via a differentiable communication channel. While these policies are effective for many tasks, interpretation of their induced communication strategies has remained a challenge. Here we propose to interpret agents' messages by translating them. Unlike in typical machine translation problems, we have no parallel data to learn from. Instead we develop a translation model based on the insight that agent messages and natural language strings mean the same thing *if they induce the same belief about the world in a listener*. We present theoretical guarantees and empirical evidence that our approach preserves both the semantics and pragmatics of messages by ensuring that players communicating through a translation layer do not suffer a substantial loss in reward relative to players with a common language.[1]

1 Introduction

Several recent papers have described approaches for learning *deep communicating policies* (DCPs): decentralized representations of behavior that enable multiple agents to communicate via a differentiable channel that can be formulated as a recurrent neural network. DCPs have been shown to solve a variety of coordination problems, including reference games (Lazaridou et al., 2016b), logic puzzles (Foerster et al., 2016), and simple control (Sukhbaatar et al., 2016). Appealingly, the agents' communication protocol can be learned via direct backpropagation through the communication channel, avoiding many of the challenging inference problems associated with learning in classical decentralized decision processes (Roth et al., 2005).

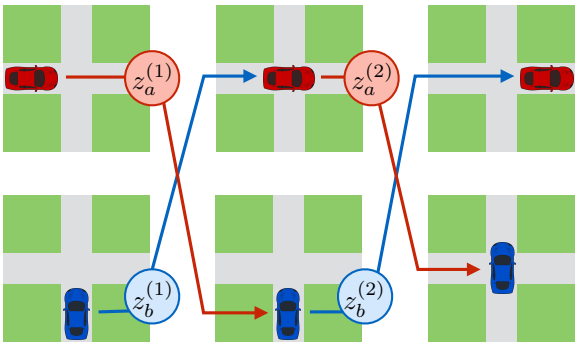


Figure 1: Example interaction between a pair of agents in a deep communicating policy. Both cars are attempting to cross the intersection, but cannot see each other. By exchanging message vectors $z^{(t)}$, the agents are able to coordinate and avoid a collision. This paper presents an approach for *understanding* the contents of these message vectors by translating them into natural language.

But analysis of the strategies induced by DCPs has remained a challenge. As an example, Figure 1 depicts a driving game in which two cars, which are unable to see each other, must both cross an intersection without colliding. In order to ensure success, it is clear that the cars must communicate with each other. But a number of successful communication strategies are possible—for example, they might report their exact (x, y) coordinates at every timestep, or they might simply announce whenever they are entering and leaving the intersection. If these messages were communicated in natural language, it would be straightforward to determine which strategy was being employed. However, DCP agents instead communicate with an automatically induced protocol of unstructured, real-valued recurrent state vectors—an artificial language we might call "neuralese," which superficially bears little resemblance to natural language, and thus frustrates attempts at direct interpretation.

[1] We have released code and data at http://github.com/jacobandreas/neuralese.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 232–242
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1022

We propose to understand neuralese messages by *translating* them. In this work, we present a simple technique for inducing a dictionary that maps between neuralese message vectors and short natural language strings, given only examples of DCP agents interacting with other agents, and humans interacting with other humans. Natural language already provides a rich set of tools for describing beliefs, observations, and plans—our thesis is that these tools provide a useful complement to the visualization and ablation techniques used in previous work on understanding complex models (Strobelt et al., 2016; Ribeiro et al., 2016).

While structurally quite similar to the task of machine translation between pairs of human languages, interpretation of neuralese poses a number of novel challenges. First, there is no natural source of parallel data: there are no bilingual "speakers" of both neuralese and natural language. Second, there may not be a direct correspondence between the strategy employed by humans and DCP agents: even if it were constrained to communicate using natural language, an automated agent might choose to produce a different message from humans in a given state. We tackle both of these challenges by appealing to the grounding of messages in gameplay. Our approach is based on one of the core insights in natural language semantics: messages (whether in neuralese or natural language) have similar meanings *when they induce similar beliefs about the state of the world*.

Based on this intuition, we introduce a translation criterion that matches neuralese messages with natural language strings by minimizing statistical distance in a common representation space of distributions over speaker states. We explore several related questions:

- What makes a good translation, and under what conditions is translation possible at all? (Section 4)
- How can we build a model to translate between neuralese and natural language? (Section 5)
- What kinds of theoretical guarantees can we provide about the behavior of agents communicating via this translation model? (Section 6)

Our translation model and analysis are general, and in fact apply equally to human–computer and human–human translation problems grounded in gameplay. In this paper, we focus our experiments specifically on the problem of interpreting communication in deep policies, and apply our approach to the driving game in Figure 1 and two reference games of the kind shown in Figure 2. We find that this approach outperforms a more conventional machine translation criterion both when attempting to interoperate with neuralese speakers and when predicting their state.

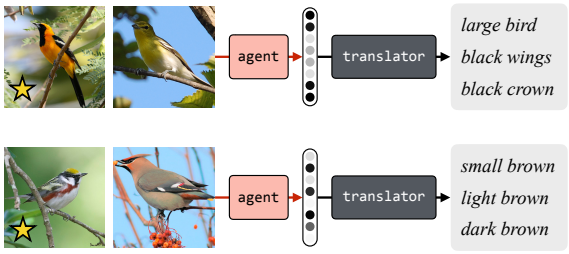


Figure 2: Overview of our approach—best-scoring translations generated for a reference game involving images of birds. The speaking agent's goal is to send a message that uniquely identifies the bird on the left. From these translations it can be seen that the learned model appears to discriminate based on coarse attributes like size and color.

2 Related work

A variety of approaches for learning deep policies with communication were proposed essentially simultaneously in the past year. We have broadly labeled these as "deep communicating policies"; concrete examples include Lazaridou et al. (2016b), Foerster et al. (2016), and Sukhbaatar et al. (2016). The policy representation we employ in this paper is similar to the latter two of these, although the general framework is agnostic to low-level modeling details and could be straightforwardly applied to other architectures. Analysis of communication strategies in all these papers has been largely ad-hoc, obtained by clustering states from which similar messages are emitted and attempting to manually assign semantics to these clusters. The present work aims at developing tools for performing this analysis automatically.

Most closely related to our approach is that of Lazaridou et al. (2016a), who also develop a model for assigning natural language interpretations to learned messages; however, this approach relies on supervised cluster labels and is targeted specifically towards referring expression games. Here we attempt to develop an approach that can handle general multiagent interactions without assuming a prior discrete structure in space of observations.

The literature on learning decentralized multiagent policies in general is considerably larger (Bernstein et al., 2002; Dibangoye et al., 2016). This includes work focused on communication in multiagent settings (Roth et al., 2005) and even communication using natural language messages (Vogel et al., 2013b). All of these approaches employ structured communication schemes with manually engineered messaging protocols; these are, in some sense, automatically interpretable, but at the cost of introducing considerable complexity into both training and inference.

Our evaluation in this paper investigates communication strategies that arise in a number of different games, including reference games and an extended-horizon driving game. Communication strategies for reference games were previously explored by Vogel et al. (2013a), Andreas and Klein (2016) and Kazemzadeh et al. (2014), and reference games specifically featuring end-to-end communication protocols by Yu et al. (2016). On the control side, a long line of work considers nonverbal communication strategies in multiagent policies (Dragan and Srinivasa, 2013).

Another group of related approaches focuses on the development of more general machinery for interpreting deep models in which messages have no explicit semantics. This includes both visualization techniques (Zeiler and Fergus, 2014; Strobelt et al., 2016), and approaches focused on generating explanations in the form of natural language (Hendricks et al., 2016; Vedantam et al., 2017).

3 Problem formulation

Games Consider a cooperative game with two players a and b of the form given in Figure 3. At every step t of this game, player a makes an observation $x_a^{(t)}$ and receives a message $z_b^{(t-1)}$ from b. It then takes an action $u_a^{(t)}$ and sends a message $z_a^{(t)}$ to b. (The process is symmetric for b.) The distributions $p(u_a|x_a, z_b)$ and $p(z_a|x_a)$ together define a policy π which we assume is shared by both players, i.e. $p(u_a|x_a, z_b) = p(u_b|x_b, z_a)$ and $p(z_a|x_a) = p(z_b|x_b)$. As in a standard Markov decision process, the actions $(u_a^{(t)}, u_b^{(t)})$ alter the world state, generating new observations for both players and a reward shared by both.

The distributions $p(z|x)$ and $p(u|x, z)$ may also be viewed as defining a *language*: they specify how a speaker will generate messages based on world states, and how a listener will respond to these messages. Our goal in this work is to learn to translate between pairs of languages generated by different policies. Specifically, we assume that we have access to two policies for the same game: a "robot policy" π_r and a "human policy" π_h. We would like to use the representation of π_h, the behavior of which is transparent to human users, in order to *understand* the behavior of π_r (which is in general an uninterpretable learned model); we will do this by inducing bilingual dictionaries that map message vectors z_r of π_r to natural language strings z_h of π_h and vice-versa.

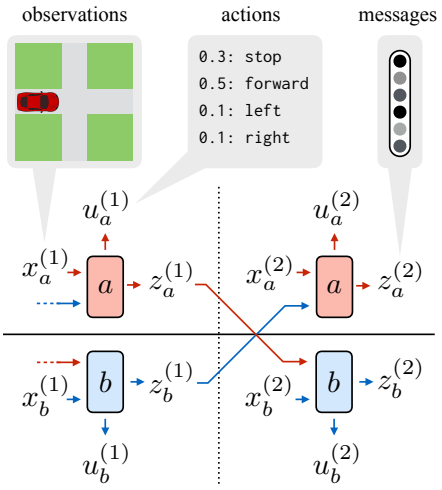


Figure 3: Schematic representation of communication games. At every timestep t, players a and b make an observation $x^{(t)}$ and receive a message $z^{(t-1)}$, then produce an action $u^{(t)}$ and a new message $z^{(t)}$.

Learned agents π_r Our goal is to present tools for interpretation of learned messages that are agnostic to the details of the underlying algorithm for acquiring them. We use a generic DCP model as a basis for the techniques developed in this paper. Here each agent policy is represented as a deep recurrent Q network (Hausknecht and Stone, 2015). This network is built from communicating cells of the kind depicted in Figure 4. At every timestep, this agent receives three pieces of information: an

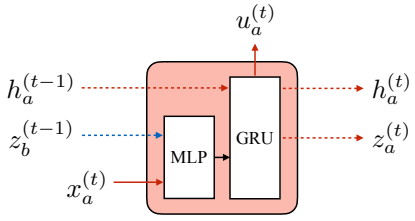


Figure 4: Cell implementing a single step of agent communication (compare with Sukhbaatar et al. (2016) and Foerster et al. (2016)). *MLP* denotes a multilayer perceptron; *GRU* denotes a gated recurrent unit (Cho et al., 2014). Dashed lines represent recurrent connections.

observation of the current state of the world, the agent's memory vector from the previous timestep, and a message from the other player. It then produces three outputs: a predicted Q value for every possible action, a new memory vector for the next timestep, and a message to send to the other agent.

Sukhbaatar et al. (2016) observe that models of this form may be viewed as specifying a single RNN in which weight matrices have a particular block structure. Such models may thus be trained using the standard recurrent Q-learning objective, with communication protocol learned end-to-end.

Human agents π_h The translation model we develop requires a representation of the distribution over messages $p(z_a|x_a)$ employed by human speakers (without assuming that humans and agents produce equivalent messages in equivalent contexts). We model the human message generation process as categorical, and fit a simple multilayer perceptron model to map from observations to words and phrases used during human gameplay.

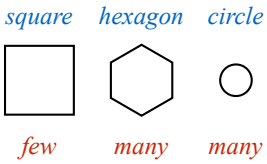
4 What's in a translation?

What does it mean for a message z_h to be a "translation" of a message z_r? In standard machine translation problems, the answer is that z_h is likely to co-occur in parallel data with z_r; that is, $p(z_h|z_r)$ is large. Here we have no parallel data: even if we could observe natural language and neuralese messages produced by agents in the same state, we would have no guarantee that these messages actually served the same function. Our answer must instead appeal to the fact that both natural language and neuralese messages are grounded in a common environment. For a given neuralese message z_r, we will first compute a grounded representation of that message's meaning; to translate, we find a natural-language message whose meaning is most similar. The key question is then what form this grounded meaning representation should take. The existing literature suggests two broad approaches:

Semantic representation The meaning of a message z_a is given by its denotations: that is, by the set of world states of which z_a may be felicitously predicated, given the existing context available to a listener. In probabilistic terms, this says that the meaning of a message z_a is represented by the distribution $p(x_a|z_a, x_b)$ it induces over speaker states. Examples of this approach include Guerin and Pitt (2001) and Pasupat and Liang (2016).

Pragmatic representation The meaning of a message z_a is given by the behavior it induces in a listener. In probabilistic terms, this says that the meaning of a message z_a is represented by the distribution $p(u_b|z_a, x_b)$ it induces over actions given the listener's observation x_b. Examples of this approach include Vogel et al. (2013a) and Gauthier and Mordatch (2016).

These two approaches can give rise to rather different behaviors. Consider the following example:



The top language (in blue) has a unique name for every kind of shape, while the bottom language (in red) only distinguishes between shapes with few sides and shapes with many sides. Now imagine a simple reference game with the following form: player a is covertly assigned one of these three shapes as a reference target, and communicates that reference to b; b must then pull a lever labeled `large` or `small` depending on the size of the target shape. Blue language speakers can achieve perfect success at this game, while red language speakers can succeed at best two out of three times.

How should we translate the blue word *hexagon* into the red language? The semantic approach suggests that we should translate *hexagon* as *many*: while *many* does not uniquely identify the hexagon, it produces a distribution over shapes that is closest to the truth. The pragmatic approach instead suggests that we should translate *hexagon* as *few*, as this is the only message that guarantees that the listener will pull the correct lever `large`. So in order to produce a correct listener action, the translator might have to "lie" and produce a maximally inaccurate listener belief.

If we were exclusively concerned with building a translation layer that allowed humans and DCP agents to interoperate as effectively as possible, it would be natural to adopt a pragmatic representation strategy. But our goals here are broader: we also want to facilitate *understanding*, and specifically to help users of learned systems form true beliefs about the systems' computational processes and representational abstractions. The example above demonstrates that "pragmatically" optimizing directly for task performance can sometimes lead to translations that produce inaccurate beliefs.

We instead build our approach around semantic representations of meaning. By preserving semantics, we allow listeners to reason accurately about the content and interpretation of messages. We might worry that by adopting a semantics-first view, we have given up all guarantees of effective interoperation between humans and agents using a translation layer. Fortunately, this is not so: as we will see in Section 6, it is possible to show that players communicating via a semantic translator perform only boundedly worse (and sometimes better!) than pairs of players with a common language.

5 Translation models

In this section, we build on the intuition that messages should be translated via their semantics to define a concrete translation model—a procedure for constructing a natural language $\leftrightarrow$ neuralese dictionary given agent and human interactions.

We understand the meaning of a message z_a to be represented by the distribution $p(x_a|z_a, x_b)$ it induces over speaker states given listener context. We can formalize this by defining the belief distribution β for a message z and context x_b as:

$$\beta(z_a, x_b) = p(x_a|z_a, x_b) = \frac{p(z_a|x_a)p(x_b|x_a)}{\sum_{x'_a} p(z_a|x'_a)p(x_b|x'_a)}$$

Here we have modeled the listener as performing a single step of Bayesian inference, using the listener state and the message generation model (by assumption shared between players) to compute the posterior over speaker states. While in general neither humans nor DCP agents compute explicit representations of this posterior, past work has found that both humans and suitably-trained neural networks can be modeled as Bayesian reasoners (Frank et al., 2009; Paige and Wood, 2016).

This provides a context-specific representation of belief, but for messages z and z' to have the same semantics, they must induce the same belief over *all* contexts in which they occur. In our probabilistic formulation, this introduces an outer expectation over contexts, providing a final measure q of the quality of a translation from z to z':

$$\begin{aligned} q(z, z') &= \mathbb{E}\big[\mathcal{D}_{\text{KL}}(\beta(z, X_b) \,||\, \beta(z', X_b)) \mid z, z'\big] \\ &= \sum_{x_a, x_b} p(x_a, x_b|z, z')\mathcal{D}_{\text{KL}}(\beta(z, x_b) \,||\, \beta(z', x_b)) \\ &\propto \sum_{x_a, x_b} p(x_a, x_b) \cdot p(z|x_a) \cdot p(z'|x_a) \\ &\qquad \cdot \mathcal{D}_{\text{KL}}(\beta(z, x_b) \,||\, \beta(z', x_b)); \end{aligned} \tag{1}$$

Algorithm 1 Translating messages

given: a phrase inventory L
function TRANSLATE(z)
  return $\arg\min_{z' \in L} \hat{q}(z, z')$

function $\hat{q}(z, z')$
  // sample contexts and distractors
  $x_{ai}, x_{bi} \sim p(X_a, X_b)$ for $i = 1..n$
  $x'_{ai} \sim p(X_a|x_{bi})$
  // compute context weights
  $\tilde{w}_i \leftarrow p(z|x_{ai}) \cdot p(z'|x_{ai})$
  $w_i \leftarrow \tilde{w}_i / \sum_j \tilde{w}_j$
  // compute divergences
  $k_i \leftarrow \sum_{x \in \{x_a, x'_a\}} p(z|x) \log \frac{p(z|x)}{p(z'|x)}$
  return $\sum_i w_i k_i$

recalling that in this setting

$$\begin{aligned} \mathcal{D}_{\text{KL}}(\beta \,||\, \beta') &= \sum_{x_a} p(x_a|z, x_b) \log \frac{p(x_a|z, x_b)}{p(x_a|z', x_b)} \\ &\propto \sum_{x_a} p(x_a|x_b)p(z|x_a) \log \frac{p(z|x_a)}{p(z'|x_a)} \end{aligned} \tag{2}$$

which is zero when the messages z and z' give rise to identical belief distributions and increases as they grow more dissimilar. To translate, we would like to compute $tr(z_r) = \arg\min_{z_h} q(z_r, z_h)$ and $tr(z_h) = \arg\min_{z_r} q(z_h, z_r)$. Intuitively, Equation 1 says that we will measure the quality of a proposed translation $z \mapsto z'$ by asking the following question: in contexts where z is likely to be used, how frequently does z' induce the same belief about speaker states as z?

While this translation criterion directly encodes the semantic notion of meaning described in Section 4, it is doubly intractable: the KL divergence and outer expectation involve a sum over all observations x_a and x_b respectively; these sums are not in general possible to compute efficiently. To avoid this, we approximate Equation 1 by sampling. We draw a collection of samples (x_a, x_b) from the prior over world states, and then generate for each sample a sequence of distractors (x'_a, x_b) from $p(x'_a|x_b)$ (we assume access to both of these distributions from the problem representation). The KL term in Equation 1 is computed over each true sample and its distractors, which are then normalized and averaged to compute the final score.

Sampling accounts for the outer $p(x_a, x_b)$ in Equation 1 and the inner $p(x_a|x_b)$ in Equation 2.

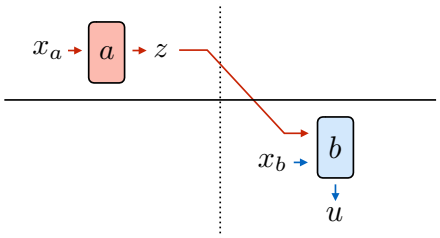


Figure 5: Simplified game representation used for analysis in Section 6. A speaker agent sends a message to a listener agent, which takes a single action and receives a reward.

The only quantities remaining are of the form $p(z|x_a)$. In the case of neuralese, this distribution already is part of the definition of the agent policy π_r and can be reused directly. For natural language, we use transcripts of human interactions to fit a model that maps from world states to a distribution over frequent utterances as discussed in Section 3. Details of these model implementations are provided in Appendix B, and the full translation procedure is given in Algorithm 1.

6 Belief and behavior

The translation criterion in the previous section makes no reference to listener actions at all. The shapes example in Section 4 shows that some model performance might be lost under translation. It is thus reasonable to ask whether this translation model of Section 5 can make any guarantees about the effect of translation on behavior. In this section we explore the relationship between belief-preserving translations and the behaviors they produce, by examining the effect of belief accuracy and strategy mismatch on the reward obtained by cooperating agents.

To facilitate this analysis, we consider a simplified family of communication games with the structure depicted in Figure 5. These games can be viewed as a subset of the family depicted in Figure 3; and consist of two steps: a listener makes an observation x_a and sends a single message z to a speaker, which makes its own observation x_b, takes a single action u, and receives a reward. We emphasize that the results in this section concern the theoretical properties of idealized games, and are presented to provide intuition about high-level properties of our approach. Section 8 investigates empirical behavior of this approach on real-world tasks where these ideal conditions do not hold.

Our first result is that translations that minimize semantic dissimilarity q cause the listener to take near-optimal actions:[2]

[2]Proof is provided in Appendix A.

Proposition 1.
Semantic translations reward rational listeners. Define a *rational listener* as one that chooses the best action in expectation over the speaker's state:

$$U(z, x_b) = \arg\max_u \sum_{x_a} p(x_a|x_b, z) r(x_a, x_b, u)$$

for a reward function $r \in [0, 1]$ that depends only on the two observations and the action.[3] Now let a be a speaker of a language r, b be a listener of the same language r, and b' be a listener of a different language h. Suppose that we wish for a and b' to interact via the translator $tr : z_r \mapsto z_h$ (so that a produces a message z_r, and b' takes an action $U(z_h = tr(z_r), x_{b'})$). If tr respects the semantics of z_r, then the bilingual pair a and b' achieves only boundedly worse reward than the monolingual pair a and b. Specifically, if $q(z_r, z_h) \leq D$, then

$$\mathbb{E}r(X_a, X_b, U(tr(Z))) \geq \mathbb{E}r(X_a, X_b, U(Z)) - \sqrt{2D} \quad (3)$$

So as discussed in Section 4, even by committing to a semantic approach to meaning representation, we have still succeeded in (approximately) capturing the nice properties of the pragmatic approach.

Section 4 examined the consequences of a mismatch between the set of primitives available in two languages. In general we would like some measure of our approach's robustness to the lack of an exact correspondence between two languages. In the case of humans in particular we expect that a variety of different strategies will be employed, many of which will not correspond to the behavior of the learned agent. It is natural to want some assurance that we can identify the DCP's strategy as long as *some* human strategy mirrors it. Our second observation is that it is possible to exactly recover a translation of a DCP strategy from a mixture of humans playing different strategies:

Proposition 2.
Semantic translations find hidden correspondences. Consider a fixed robot policy π_r and a set of human policies $\{\pi_{h1}, \pi_{h2}, \ldots\}$ (recalling from Section 3 that each π is defined by distributions

[3]This notion of rationality is a fairly weak one: it permits many suboptimal communication strategies, and requires only that the listener do as well as possible given a fixed speaker—a first-order optimality criterion likely to be satisfied by any richly-parameterized model trained via gradient descent.

$p(z|x_a)$ and $p(u|z, x_b)$). Suppose further that the messages employed by these human strategies are *disjoint*; that is, if $p_{hi}(z|x_a) > 0$, then $p_{hj}(z|x_a) = 0$ for all $j \neq i$. Now suppose that all $q(z_r, z_h) = 0$ for all messages in the support of some $p_{hi}(z|x_a)$ and > 0 for all $j \neq i$. Then every message z_r is translated into a message produced by π_{hi}, and messages from other strategies are ignored.

This observation follows immediately from the definition of $q(z_r, z_h)$, but demonstrates one of the key distinctions between our approach and a conventional machine translation criterion. Maximizing $p(z_h|z_r)$ will produce the natural language message most often produced in contexts where z_r is observed, regardless of whether that message is useful or informative. By contrast, minimizing $q(z_h, z_r)$ will find the z_h that corresponds most closely to z_r even when z_h is rarely used.

The disjointness condition, while seemingly quite strong, in fact arises naturally in many circumstances—for example, players in the driving game reporting their spatial locations in absolute vs. relative coordinates, or speakers in a color reference game (Figure 6) discriminating based on lightness vs. hue. It is also possible to relax the above condition to require that strategies be only *locally* disjoint (i.e. with the disjointness condition holding for each fixed x_a), in which case overlapping human strategies are allowed, and the recovered robot strategy is a context-weighted mixture of these.

7 Evaluation

7.1 Tasks

In the remainder of the paper, we evaluate the empirical behavior of our approach to translation. Our evaluation considers two kinds of tasks: reference games and navigation games. In a reference game (e.g. Figure 6a), both players observe a pair of candidate referents. A speaker is assigned a target referent; it must communicate this target to a listener, who then performs a choice action corresponding to its belief about the true target. In this paper we consider two variants on the reference game: a simple color-naming task, and a more complex task involving natural images of birds. For examples of human communication strategies for these tasks, we obtain the XKCD color dataset (McMahan and Stone, 2015; Monroe et al., 2016) and the Caltech Birds dataset (Welinder et al., 2010) with accompanying natural language descriptions (Reed et al., 2016). We use standard train / validation / test splits for both of these datasets.

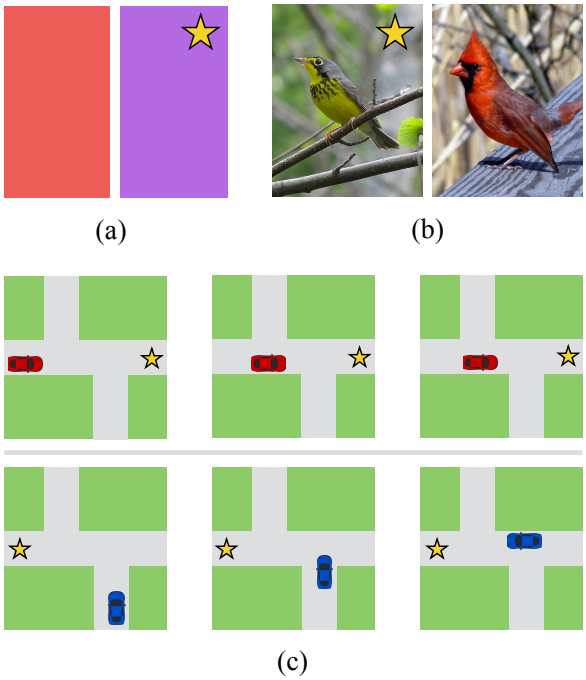


Figure 6: Tasks used to evaluate the translation model. (a–b) Reference games: both players observe a pair of reference candidates (colors or images); Player a is assigned a target (marked with a star), which player b must guess based on a message from a. (c) Driving game: each car attempts to navigate to its goal (marked with a star). The cars cannot see each other, and must communicate to avoid a collision.

The final task we consider is the driving task (Figure 6c) first discussed in the introduction. In this task, two cars, invisible to each other, must each navigate between randomly assigned start and goal positions without colliding. This task takes a number of steps to complete, and potentially involves a much broader range of communication strategies. To obtain human annotations for this task, we recorded both actions and messages generated by pairs of human Amazon Mechanical Turk workers playing the driving game with each other. We collected close to 400 games, with a total of more than 2000 messages exchanged, from which we held out 100 game traces as a test set.

7.2 Metrics

A mechanism for understanding the behavior of a learned model should allow a human user both to correctly infer its beliefs and to successfully interoperate with it; we accordingly report results of both "belief" and "behavior" evaluations.

To support easy reproduction and comparison (and in keeping with standard practice in machine

translation), we focus on developing automatic measures of system performance. We use the available training data to develop simulated models of human decisions; by first showing that these models track well with human judgments, we can be confident that their use in evaluations will correlate with human understanding. We employ the following two metrics:

Belief evaluation This evaluation focuses on the denotational perspective in semantics that motivated the initial development of our model. We have successfully understood the semantics of a message z_r if, after translating $z_r \mapsto z_h$, a human listener can form a correct belief about the state in which z_r was produced. We construct a simple state-guessing game where the listener is presented with a translated message and two state observations, and must guess which state the speaker was in when the message was emitted.

When translating from natural language to neuralese, we use the learned agent model to directly guess the hidden state. For neuralese to natural language we must first construct a "model human listener" to map from strings back to state representations; we do this by using the training data to fit a simple regression model that scores (state, sentence) pairs using a bag-of-words sentence representation. We find that our "model human" matches the judgments of real humans 83% of the time on the colors task, 77% of the time on the birds task, and 77% of the time on the driving task. This gives us confidence that the model human gives a reasonably accurate proxy for human interpretation.

Behavior evaluation This evaluation focuses on the cooperative aspects of interpretability: we measure the extent to which learned models are able to interoperate with each other by way of a translation layer. In the case of reference games, the goal of this semantic evaluation is identical to the goal of the game itself (to identify the hidden state of the speaker), so we perform this additional pragmatic evaluation only for the driving game. We found that the most data-efficient and reliable way to make use of human game traces was to construct a "deaf" model human. The evaluation selects a full game trace from a human player, and replays both the human's actions and messages exactly (disregarding any incoming messages); the evaluation measures the quality of the natural-language-to-neuralese translator, and the extent to which the learned agent model can accommodate a (real) human given translations of the human's messages.

(a)

as listener	as speaker: R	as speaker: H	
R	1.00	0.50	random
		0.70	direct
		0.73	belief (ours)
H*	0.50	0.83	
	0.72		
	0.86		

(b)

as listener	as speaker: R	as speaker: H	
R	0.95	0.50	random
		0.55	direct
		0.60	belief (ours)
H*	0.50	0.77	
	0.57		
	0.75		

Table 1: Evaluation results for reference games. (a) The colors task. (b) The birds task. Whether the model human is in a listener or speaker role, translation based on belief matching outperforms both random and machine translation baselines.

Baselines We compare our approach to two baselines: a *random* baseline that chooses a translation of each input uniformly from messages observed during training, and a *direct* baseline that directly maximizes $p(z'|z)$ (by analogy to a conventional machine translation system). This is accomplished by sampling from a DCP speaker in training states labeled with natural language strings.

8 Results

In all below, "R" indicates a DCP agent, "H" indicates a real human, and "H*" indicates a model human player.

Reference games Results for the two reference games are shown in Table 1. The end-to-end trained model achieves nearly perfect accuracy in both

Figure 7: Best-scoring translations generated for color task.

as listener	as speaker R	as speaker H	
R	0.85	0.50	random
		0.45	direct
		0.61	belief (ours)
H*	0.5	0.77	
	0.45		
	0.57		

Table 2: Belief evaluation results for the driving game. Driving states are challenging to identify based on messages alone (as evidenced by the comparatively low scores obtained by single-language pairs) . Translation based on belief achieves the best overall performance in both directions.

R / R	H / H	R / H	
1.93 / 0.71	— / 0.77	1.35 / 0.64	random
		1.49 / **0.67**	direct
		1.54 / 0.67	belief (ours)

Table 3: Behavior evaluation results for the driving game. Scores are presented in the form "reward / completion rate". While less accurate than either humans or DCPs with a shared language, the models that employ a translation layer obtain higher reward and a greater overall success rate than baselines.

cases, while a model trained to communicate in natural language achieves somewhat lower performance. Regardless of whether the speaker is a DCP and the listener a model human or vice-versa, translation based on the belief-matching criterion in Section 5 achieves the best performance; indeed, when translating neuralese color names to natural language, the listener is able to achieve a slightly higher score than it is natively. This suggests that the automated agent has discovered a more effective strategy than the one demonstrated by humans in the dataset, and that the effectiveness of this strategy is preserved by translation. Example translations from the reference games are depicted in Figure 2 and Figure 7.

Driving game Behavior evaluation of the driving game is shown in Table 3, and belief evaluation is shown in Table 2. Translation of messages in the driving game is considerably more challenging than in the reference games, and scores are uniformly lower; however, a clear benefit from the belief-matching model is still visible. Belief matching leads to higher scores on the belief evaluation in both directions, and allows agents to obtain a higher reward on average (though task completion rates remain roughly the same across all agents). Some example translations of driving game messages are shown in Figure 8.

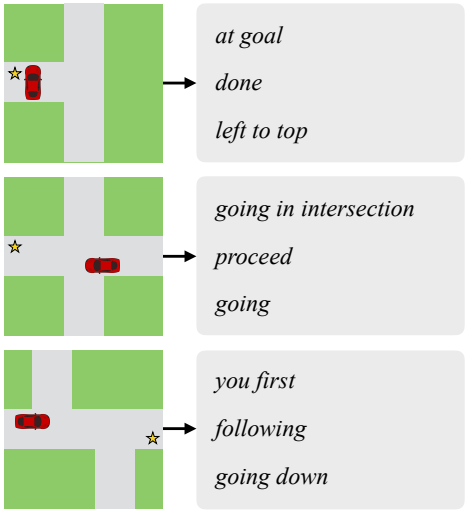


Figure 8: Best-scoring translations generated for driving task generated from the given speaker state.

9 Conclusion

We have investigated the problem of interpreting message vectors from deep networks by translating them. After introducing a translation criterion based on matching listener beliefs about speaker states, we presented both theoretical and empirical evidence that this criterion outperforms a conventional machine translation approach at recovering the content of message vectors and facilitating collaboration between humans and learned agents.

While our evaluation has focused on understanding the behavior of deep communicating policies, the framework proposed in this paper could be much more generally applied. Any encoder–decoder model (Sutskever et al., 2014) can be thought of as a kind of communication game played between the encoder and the decoder, so we can analogously imagine computing and translating "beliefs" induced by the encoding to explain what features of the input are being transmitted. The current work has focused on learning a purely categorical model of the translation process, supported by an unstructured inventory of translation candidates, and future work could explore the *compositional* structure of messages, and attempt to synthesize novel natural language or neuralese messages from scratch. More broadly, the work here shows that the denotational perspective from formal semantics provides a framework for precisely framing the demands of interpretable machine learning (Wilson et al., 2016), and particularly for ensuring that human users without prior exposure to a learned model are able to interoperate with it, predict its behavior, and diagnose its errors.

Acknowledgments

JA is supported by a Facebook Graduate Fellowship and a Berkeley AI / Huawei Fellowship. We are grateful to Lisa Anne Hendricks for assistance with the Caltech Birds dataset.

References

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research* 27(4):819–840.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Jilles Steeve Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. 2016. Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research* 55:443–497.

Anca Dragan and Siddhartha Srinivasa. 2013. Generating legible motion. In *Robotics: Science and Systems*.

Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*. pages 2137–2145.

Michael C Frank, Noah D Goodman, Peter Lai, and Joshua B Tenenbaum. 2009. Informative communication in word production and word learning. In *Proceedings of the 31st annual conference of the cognitive science society*. pages 1228–1233.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. 2016. Compact bilinear pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 317–326.

Jon Gauthier and Igor Mordatch. 2016. A paradigm for situated and goal-driven language learning. *arXiv preprint arXiv:1610.03585* .

Frank Guerin and Jeremy Pitt. 2001. Denotational semantics for agent communication language. In *Proceedings of the fifth international conference on Autonomous agents*. ACM, pages 497–504.

Matthew Hausknecht and Peter Stone. 2015. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527* .

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating visual explanations. In *European Conference on Computer Vision*. Springer, pages 3–19.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. ReferItGame: Referring to objects in photographs of natural scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 787–798.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2016a. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182* .

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2016b. Towards multi-agent communication-based language learning. *arXiv preprint arXiv:1605.07133* .

Brian McMahan and Matthew Stone. 2015. A Bayesian model of grounded color semantics. *Transactions of the Association for Computational Linguistics* 3:103–115.

Will Monroe, Noah D Goodman, and Christopher Potts. 2016. Learning to generate compositional color descriptions. *arXiv preprint arXiv:1606.03821* .

Brooks Paige and Frank Wood. 2016. Inference networks for sequential monte carlo in graphical models. volume 48.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. *arXiv preprint arXiv:1606.06900* .

Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. 2016. Learning deep representations of fine-grained visual descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 49–58.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 1135–1144.

Maayan Roth, Reid Simmons, and Manuela Veloso. 2005. Reasoning about joint beliefs for execution-time communication decisions. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*. ACM, pages 786–793.

Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M Rush. 2016. Visual analysis of hidden state dynamics in recurrent neural networks. *arXiv preprint arXiv:1606.07461* .

Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*. pages 2244–2252.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. pages 3104–3112.

Ramakrishna Vedantam, Samy Bengio, Kevin Murphy, Devi Parikh, and Gal Chechik. 2017. Context-aware captions from context-agnostic supervision. *arXiv preprint arXiv:1701.02870* .

Adam Vogel, Max Bodoia, Christopher Potts, and Daniel Jurafsky. 2013a. Emergence of Gricean maxims from multi-agent decision theory. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. pages 1072–1081.

Adam Vogel, Christopher Potts, and Dan Jurafsky. 2013b. Implicatures and nested beliefs in approximate Decentralized-POMDPs. In *ACL (2)*. pages 74–80.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. 2010. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology.

Andrew Gordon Wilson, Been Kim, and William Herlands. 2016. Proceedings of nips 2016 workshop on interpretable machine learning for complex systems. *arXiv preprint arXiv:1611.09139* .

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. 2016. A joint speaker-listener-reinforcer model for referring expressions. *arXiv preprint arXiv:1612.09542* .

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*. Springer, pages 818–833.

Obtaining referential word meanings from visual and distributional information: Experiments on object naming

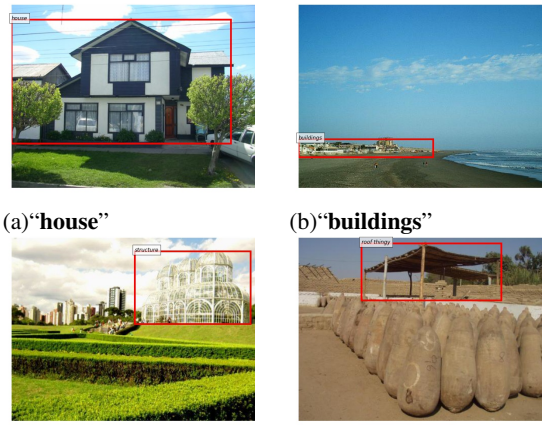
Sina Zarrieß and **David Schlangen**
Dialogue Systems Group // CITEC // Faculty of Linguistics and Literary Studies
Bielefeld University, Germany
{sina.zarriess,david.schlangen}@uni-bielefeld.de

Abstract

We investigate object naming, which is an important sub-task of referring expression generation on real-world images. As opposed to mutually exclusive labels used in object recognition, object names are more flexible, subject to communicative preferences and semantically related to each other. Therefore, we investigate models of referential word meaning that link visual to lexical information which we assume to be given through distributional word embeddings. We present a model that learns individual predictors for object names that link visual and distributional aspects of word meaning during training. We show that this is particularly beneficial for zero-shot learning, as compared to projecting visual objects directly into the distributional space. In a standard object naming task, we find that different ways of combining lexical and visual information achieve very similar performance, though experiments on model combination suggest that they capture complementary aspects of referential meaning.

1 Introduction

Expressions referring to objects in visual scenes typically include a word naming the type of the object: E.g., *house* in Figure 1 (a), or, as a very general type, *thingy* in Figure 1 (d). Determining such a name is a crucial step for referring expression generation (REG) systems, as many other decisions concerning, e.g., the selection of attributes follow from it (Dale and Reiter, 1995; Krahmer and Van Deemter, 2012). For a long time, however, research on REG mostly assumed the availability of symbolic representations of referent and scene, and sidestepped questions about how speakers actually choose these names, due to the lack of models capable of capturing what a word like *house* refers to in the real world.



(a)"**house**" (b)"**buildings**"

(c)"large **structure**" (d)"roof **thingy**"

Figure 1: Examples of object names in the REFERIT corpus referring to instances of buildings

Recent advances in image processing promise to fill this gap, with state-of-the-art computer vision systems being able to classify images into thousands of different categories (e.g. Szegedy et al. (2015)). However, classification is not naming (Ordonez et al., 2016). Standard object classification schemes are inherently "flat", and treat object labels as mutually exclusive (Deng et al., 2014). A state-of-the-art object recognition system would be trained to classify the object in e.g. Figure 1 (a) as either *house* or *building*, ignoring the lexical similarity between these two names. In contrast, humans seem to be more flexible as to the chosen level of generality. Depending on the prototypicality of the object to name, and possibly other visual properties, a general name might be more or less appropriate. For instance, a robin can be named *bird*, but a penguin is better referred

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 243–254
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1023

to as "*penguin*" (Rosch, 1978); along the same lines, the rather unusual building in Figure 1 (c) that is not easy to otherwise categorise was named "*structure*".

Other work at the intersection of image and language processing has investigated models that learn to directly associate visual objects with a continuous representation of word meaning, i.e. through cross-modal transfer into distributional vector spaces (Frome et al., 2013; Norouzi et al., 2013). Here, the idea is to exploit a powerful model of lexical similarity induced from large amounts text for being able to capture inherent lexical relations between object categories. Thus, under the assumption that such semantic spaces represent, in some form at least, taxonomic knowledge, this makes labels on different levels of specificity available for a given object. Moreover, if the mapping is sufficiently general, it should be able to map objects to an appropriate label, even if during training of the mapping this label has not been seen (*zero-shot learning*).

While cross-modal transfer seems to be a conceptually attractive model for learning object names, it is based on an important assumption that, in our view, has not received sufficient attention in previous works: it assumes that a given distributional vector space constitutes an optimal target representation that visual instances of objects can be mapped to. However, distributional representations of word meaning are known to capture a rather fuzzy notion of lexical similarity, e.g. *car* is similar to *van* and to *street*. A cross-modal transfer model is "forced" to learn to map objects into the same area in the semantic space if their names are distributionally similar, but regardless of their actual visual similarity. Indeed, we have found in a recent study that the contribution of distributional information to learning referential word meanings is restricted to certain types of words and does not generalize across the vocabulary (Zarrieß and Schlangen, 2017).

The goal of this work is to learn a model of referential word meaning that makes accurate object naming predictions and goes beyond treating words as independent, mutually exclusive labels in a flat classification scheme. We extend upon work on learning models of referential word use from corpora of images paired with referring expressions (Schlangen et al., 2016; Zarrieß and Schlangen, 2017) that treats words as individual predictors capturing referential appropriateness. We explore different ways of linking these predictors to distributional knowledge, during application and during training. We find that these different models achieve very similar performance in a standard object naming task, though experiments on model combination suggest that they capture complementary aspects of referential meaning. In a zero-shot setup of an object naming task, we find that combining lexical and visual information during training is most beneficial, outperforming variants of cross-modal transfer.

2 Related Work

Grounding and Reference An early example for work in REG that goes beyond Dale and Reiter (1995)'s dominant symbolic paradigm is Deb Roy's work from the early 2000s (Roy et al., 2002; Roy, 2002, 2005). Roy et al. (2002) use computer vision techniques to process a video feed, and to compute colour, positional and spatial features. These features are then associated in a learning process with certain words, resulting in an association of colour features with colour words, spatial features with prepositions, etc., and based on this, these words can be interpreted with reference to the scene currently presented to the video feed. Whereas Roy's work still looked at relatively simple scenes with graphical objects, research on REG has recently started to investigate set-ups based on real-world images (Kazemzadeh et al., 2014; Gkatzia et al., 2015; Zarrieß and Schlangen, 2016; Mao et al., 2015). Importantly, the low-level visual features that can be extracted from these scenes correspond less directly to particular word classes. Moreover, the visual scenes contain many different types of objects, which poses new challenges for REG. For instance, Zarrieß and Schlangen (2016) find that semantic errors related to mismatches between nouns (e.g. the system generates *tree* vs. *man*) are particularly disturbing for users. Whereas Zarrieß and Schlangen (2016) propose a strategy to avoid object names when the systems confidence is low, we focus on improving the generation of object names, using distributional knowledge as an additional source. Similarly, Ordonez et al. (2016) have studied the problem of deriving appropriate object names, or so-called entry-level categories, from the output of an object recognizer. Their approach focusses on linking abstract object categories in ImageNet

to actual words via various translation procedures. We are interested in learning referential appropriateness and extensional word meanings directly from actual human referring expressions (REs) paired with objects in images, using an existing object recognizer for feature extraction.

Multi-modal distributional semantics Distributional semantic models are a well-known method for capturing lexical word meaning in a variety of tasks (Turney and Pantel, 2010; Mikolov et al., 2013; Erk, 2016). Recent work on multimodal distributional vector spaces (Feng and Lapata, 2010; Silberer and Lapata, 2014; Kiela and Bottou, 2014; Lazaridou et al., 2015b; Kottur et al., 2016) has aimed at capturing semantic similarity even more accurately by integrating distributional and perceptual features associated with words (mostly taken from images) into a single representation.

Cross-modal transfer Rather than fusing different modalities into a single, joint space, other work has looked at cross-modal mapping between spaces. Herbelot and Vecchi (2015) present a model that learns to map vectors in a distributional space to vectors in a set-theoretic space, showing that there is a functional relationship between distributional information and conceptual knowledge representing quantifiers and predicates. More related to our work are cross-modal mapping models,that learn to transfer from a representation of an object or image in the visual space to a vector in a distributional space (Socher et al., 2013; Frome et al., 2013; Norouzi et al., 2013; Lazaridou et al., 2014). Here, the motivation is to exploit the rich lexical knowledge encoded in a distributional space for learning visual classifications. In practice, these models are mostly used for zero-shot learning where the test set contains object categories not observed during training. When tested on standard object recognition tasks, transfer, however, comes at a price. Frome et al. (2013) and Norouzi et al. (2013) both find that it slightly degrades performance as compared to a plain object classification using standard accuracy metrics (called flat "hit @k metric" in their paper). Interestingly though, Frome et al. (2013) report better performance using "hierarchical precision", which essentially means that transfer predicts words that are ontologically closer to the gold label and makes "semantically more reasonable errors". To the best of our knowledge, this pattern has not been systematically investigated any further. Another known problem with cross-modal transfer is that it seems to generalize less well than expected, i.e. tends to reproduce word vectors observed during training (Lazaridou et al., 2015a). In this work, we present a model that exploits distributional knowledge for learning referential word meaning as well, but explore and compare different ways of combining visual and lexical aspects of referential word meaning.

3 Task and Data

We define *object naming* as follows: Given an object x in an image, the task is to predict a word w that could be used as the head noun of a realistic referring expression. (Cf. discussion above: "*bird*" when naming a robin, but "*penguin*" when naming a penguin.) To get at this, we develop our approach using a corpus of referring expressions produced by human users under natural, interactive conditions (Kazemzadeh et al., 2014), and train and test on the corresponding head nouns in these REs. This is similar to picture naming setups used in psycholinguistic research (cf. Levelt et al. (1991)) and based on the simplifying assumption that the name used for referring to an object can be determined successfully without looking at other objects in the image.

We now summarise the details of our setup:

Corpus We train and test on the REFERIT corpus (Kazemzadeh et al., 2014), which is based on the SAIAPR image collection (Grubinger et al., 2006) (99.5k image regions;120K REs). We follow (Schlangen et al., 2016) and select words with a minimum frequency of 40 in these two data sets, which gives us a vocabulary of 793 words.

Names For most of our experiments, we only use a subset of this vocabulary, namely the set of object names. As the REs contain nouns that cannot be considered to be object names (*background, bottom*, etc.), we extract a list of names from the semantically annotated held-out set released with the REFERIT. These correspond to 'entry-level' nouns mentioned in Kazemzadeh et al. (2014). This gives us a list of 159 names. This set corresponds to the majority of object names in the corpus: out of the 99.5K available image regions, we use 80K for training and testing. Thus, our experiments are on a smaller scale as compared

to (Ordonez et al., 2016). Nevertheless, the data is challenging, as the corpus contains references to objects that fall outside of the object labeling scheme that available object recognition systems are typically optimized for, cf. Hu et al. (2015)'s discussion on "stuff" entities such "sky" or "grass" in the REFERIT data. For testing, we remove relational REs (containing a relational preposition such as 'left of X'), because here we cannot be sure that the head noun of the target is fully informative; we also remove REs with more than one head noun from our list (i.e. these are mostly relational expressions as well such as 'girl laughing at boy'). We pair each image region from the test set with its corresponding names from the remaining REs.

Image and Word Embeddings Following Schlangen et al. (2016), we derive representations of our visual inputs with a convolutional neural network, 'GoogleNet' (Szegedy et al., 2015), which was trained on the ImageNet corpus (Deng et al., 2009), and extract the final fully-connected layer before the classification layer, to give us a 1024 dimensional representation of the region. We add 7 features that encode information about the region relative to the image, thus representing each object as a vector of 1031 features. As distributional word vectors, we use the `word2vec` representations provided by Baroni et al. (2014) (trained with CBOW, 5-word context window, 10 negative samples, 400 dimensions).

4 Three Models of Interfacing Visual and Distributional Information

4.1 Direct Cross-Modal Mapping

Following Lazaridou et al. (2014), referential meaning can be represented as a translation function that projects visual representations of objects to linguistic representations of words in a distributional vector space. Thus, in contrast to standard object recognition systems or the other models we will use here, cross-modal mapping does not treat words as individual labels or classifiers, but learns to directly predict continuous representations of words in a vector space, such as the space defined by the `word2vec` embeddings that we use in this work. This model will be called TRANSFER below.

During training, we pair each object with the distributional embedding of its name, and use standard Ridge regression for learning the transformation. Lazaridou et al. (2014) and Lazaridou et al. (2015a) test a range of technical tweaks and different algorithms for cross-modal mapping. For ease of comparison with other models, we stick with simple Ridge Regression in this work.

For decoding, we map an object into the distributional space, and retrieve the nearest neighbors of the predicted vector using cosine similarity. In theory, the model should generalize easily to words that it has not observed in a pair with an object during training as it can map an object anywhere in the distributional space.

4.2 Lexical Mapping Through Individual Word Classifiers

Another approach is to keep visual and distributional information separate, by training a separate visual classifier for each word w in the vocabulary. Predictions can then be mapped into distributional space during application time via the vectors of the predicted words. Here, we use Schlangen et al. (2016)'s WAC model, building the training set for each word w as follows: all visual objects in a corpus that have been referred to as w are used as positive instances, the remaining objects as negative instances. Thus, the classifiers learn to predict referential appropriateness for individual words based on the visual features of the objects they refer to, in isolation of other words.

During decoding, we apply all word classifiers from the model's vocabulary to the given object, and take the argmax over the individual word probabilities. The model predicts names directly, without links into a distributional space.

In order to extend the model's vocabulary for zero-shot learning, we follow Norouzi et al. (2013) and associate the top n words with their corresponding distributional vector and compute the convex combination of these vectors. Then, in parallel to cross-modal mapping, we retrieve the nearest neighbors of the combined embedding from the distributional space. Thus, with this model, we use two different modes of decoding: one that projects into distributional space, one that only applies the available word classifiers.

We did some small-scale experiments to find an optimal value for n, similar to Norouzi et al. (2013). In our case, performance started to decrease systematically with $n > 10$, but did not differ significantly for values below 10. In Section 5, we will report results for n set to 5 and 10.

4.3 Word Prediction via Cross-Modal Similarity Mapping

Finally, we implement an approach that combines ideas from cross-modal mapping with the WAC model: we train individual predictors for each word in the vocabulary, but, during training, we exploit lexical similarity relations encoded in a distributional space. Instead of treating a word as a binary classifier, we annotate its training instances with a fine-grained similarity signal according to their object names. When building the training set for such a word predictor w, instead of simply dividing objects into w and $\neg w$ instances, we label each object with a real-valued similarity obtained from cosine similarity between w and v in a distributional vector space, where v is the word that was used to refer to the object. Thus, we task the model with jointly learning similarities and referential appropriateness, by training it with Ridge regression on a continuous output space. Object instances where $v = w$ (i.e., the positive instances in the binary setup) have maximal similarity; the remaining instances have a lower value which is more or less close to maximal similarity. This is the SIM-WAP model, recently proposed in Zarrieß and Schlangen (2017).

Importantly, and going beyond Zarrieß and Schlangen (2017), this model allows for an innovative treatment of words that only exist in a distributional space (without being paired with visual referents in the image corpus): as the predictors are trained on a continuous output space, no genuine positive instances of a word's referent are needed. When training a predictor for such a word w, we use all available objects from our corpus and annotate them with the expected lexical similarity between w and the actual object names v, which for all objects will be below the maximal value that marks genuine positive instances. During decoding, this model does not need to project its predictions into a distributional space, but it simply applies all available predictors to the object, and takes the argmax over the predicted referential appropriateness scores.

5 Experiment 1: Naming Objects

This Section reports on experiments in a standard setup of the object naming task where all object names are paired with visual instances of their referents during training. In a comparable task, i.e. object recognition with known object categories, cross-modal projection or transfer approaches have been reported to perform worse than standard object classification methods (Frome et al., 2013; Norouzi et al., 2013). This seems to suggest that lexical or at least distributional knowledge is detrimental when learning what a word refers to in the real world and that referential meaning should potentially be learned from visual object representation only.

5.1 Model comparison

Setup We use the train/test split of REFERIT data as in (Schlangen et al., 2016). We consider image regions with non-relational referring expressions that contain at least one of the 159 head nouns from the list of entry-level nouns (see section 3). This amounts to 6208 image regions for testing and 73K instances for training.

Results Table 1 shows accuracies in the object naming task for the TRANSFER, WAC and SIM-WAP models according to their accuracies in the top n, including two variants of WAC where its top 5 and top 10 predictions are projected into the distributional space. Overall, the models achieve very similar performance. However, there is an interesting pattern when comparing accuracies @1 and @2 to accuracies in the top 5 predictions. Thus, looking at accuracies for the top (two) predictions, the various models that link referential meaning to word representations in the distributional space all perform slightly worse than the plain WAC model, i.e. individual word classifiers trained on visual features only. This might suggest that certain aspects of referential word meaning are learned less accurately when mapping from visual to distributional space (which replicates results reported in the literature on standard object recognition benchmarks). On the other hand, the SIM-WAP model is on a par with WAC in terms of the @5 accuracy. This effect suggests that distributional knowledge that SIM-WAP has access to during training sometimes distracts the model from predicting the exact name chosen by a human speaker, but that SIM-WAP is still able to rank it among the most probable names. As a simple accuracy-based evaluation is not suited to fully explain this pattern, we carry out a more detailed analysis in Section 5.3.

	hit @k(%)		
	@1	@2	@5
transfer	48.34	60.49	74.89
wac	**49.34**	**61.86**	**75.35**
wac, project top5	48.73	61.10	74.07
wac, project top10	48.68	61.23	74.31
sim-wap	48.13	60.60	**75.40**

Table 1: Accuracies in object naming

	hit @k(%)		
	1	5	10
sim-wap + transfer	49.10	61.78	75.81
sim-wap + wac	51.10	63.45	77.92
transfer + wac	51.13	63.76	77.84
wac + transfer + sim-wap	**52.19**	**64.71**	**78.40**

Table 2: Object naming acc., combined models

5.2 Model combination

In order to get more insight into why the TRANSFER and SIM-WAP models produce slightly worse results than individual visual word classifiers, we now test to what extent the different models are complementary and combine them by aggregating over their naming predictions. If the models are complementary, their combination should lead to more confident and accurate naming decisions.

Setup We combine TRANSFER, SIM-WAP and WAC by aggregating the scores they predict for different object names for a given object. During testing, we apply all models to an image region and consider words ranked among the top 10. We first normalize the referential appropriateness scores in each top-10 list and then compute their sum. This aggregation scheme will give more weight to words that appear in the top 10 list of different models, and less weight to words that only get top-ranked by a single model. We test on the same data as in Section 5.1.

Results Table 2 shows that all model combinations improve over the results of their isolated models in Table 1, suggesting that WAC, TRANSFER and SIM-WAP indeed do capture complementary aspects of referential word meaning. On their own, the distributionally informed models are less tuned to specific word occurrences than the visual word classifiers in the WAC model, but they can add useful information which leads to a clear overall improvement. We take this as a promising finding, supporting our initial hypothesis that knowledge on lexical distributional meaning should and can be exploited when learning how to use words for reference.

	Av. cosine similarity			
	among top k		gold - top k	
	5	10	5	10
transfer	**0.32**	**0.27**	**0.28**	**0.25**
wac	0.18	0.20	0.18	0.16
sim-wap	**0.32**	0.26	**0.28**	**0.25**

Table 3: Cosine similarities between word2vec embeddings of nouns generated in the top k

5.3 Analysis

Figure 2 illustrates objects from our test set where the combination of TRANSFER, SIM-WAP and WAC predicts an accurate name, whereas the models in isolation do not. These examples give some interesting insight into why the models capture different aspects of referential word meaning.

Word Similarities Many of the examples in Figure 2 suggest that the object names ranked among the top 3 by the TRANSFER and SIM-WAP model are semantically similar to each other, whereas WAC generates object names on top that describe very different underlying object categories, such as *seal* / *rock* in Figure 2(a), *animal* / *lamp* in Figure 2(g) or *chair* / *shirt* in Figure 2(c). To quantify this general impression, Table 3 shows cosine similarities among words in the top n generated by our models, using their word2vec embeddings. The average cosine similarity between words in our vocabulary is 0.17. The TRANSFER and SIM-WAP model rank words on top that are clearly more similar to each other than word pairs on average, whereas words ranked top by the WAC model are more dissimilar to each other. Another remarkable finding is that the words generated by TRANSFER and SIM-WAP are not only more similar among the top predictions, but also more similar to the gold name (Table 3 , right columns). This result is noteworthy since the accuracies for the top predictions shown in Table 1 are slightly below WAC. In general, this suggests that there is a trade-off between optimizing a model of referential word meaning to exact naming decisions, or tailoring it to make lexically consistent predictions. This parallels findings by Frome et al. (2013) who found that their transfer-based object recognition made "semantically more reasonable" errors than a standard convolutional network while

not improving accuracies for object recognition, see discussion in Section 2. Additional evaluation metrics, such as success rates in a human evaluation (cf. Zarrieß and Schlangen (2016)), would be an interesting direction for more detailed investigation here.

Word Use But even though the WAC classifiers lack knowledge on lexical similarities, they seem to able to detect relatively specific instances of word use such as *hut* in Figure 2(b), *shirt* in 2(c) or *lamp* in 2(h). Here, the combination with TRANSFER and SIM-WAP is helpful to give more weight to the object name that is taxonomically correct (sometimes pushing up words below the top-3 and hence not shown in Figure 2). In Figure 1(e), SIM-WAP and TRANSFER give more weight to typical names for persons, whereas WAC top-ranks more unusual names, reflecting that the person is difficult to identify visually. Another observation is that the mapping models have difficulties dealing with object names in singular and plural. As these words have very similar representations in the distributional space, they are often predicted as likely variants among the top 10 by SIM-WAP and TRANSFER, whereas the WAC model seems to predict inappropriate plural words less often among the top 3. Such specific phenomena at the intersection of visual and semantic similarity have found very little attention in the literature. We will investigate them further in our Experiments on zero-shot naming in the following Section.

6 Zero-Shot Naming

Zero-shot learning is an attractive prospect for REG from images, as it promises to overcome dependence on pairings of visual instances and natural names being available for all names, if visual/referential data can be generalised from other types of information. Previous work has looked at the feasibility of zero-shot learning as a function of semantic similarity or ontological closeness between unknown and known categories, and confirmed the intuition that the task is harder the less close unknown categories are to known ones (Frome et al., 2013; Norouzi et al., 2013).

Our experiments on object naming in Section 5 suggest that lexical similarities encoded in a distributional space might not always fully carry over to referential meaning. This could constitute an additional challenge for zero-shot learning, as distributional similarities might be misleading when the model has to fully rely on them for learning referential word meanings. Therefore, the following experiments investigate the performance of our models in zero-shot naming as a function of the lexical relation between unknown and known object names, i.e. namely hypernyms and singular/plurals. Both relations are typically captured by distributional models of word meaning in terms of closeness in the vector space, but their visual and referential relation is clearly different.

6.1 Vocabulary Splits and Testsets

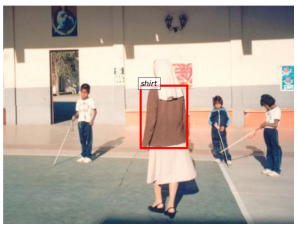
Random As in previous work on zero-shot learning, we consider zero-shot naming for words of varying degrees of similarity. We randomly split our 159 names from Experiment 1 into 10 subsets. We train the models on 90% of the nouns (and all their visual instances in the image corpus) and test on the set of image regions that are named with words which the model did not observe during training. Results reported in Table 4 on the random test set correspond to averaged scores from cross-validation over the 10 splits.

Hypernyms We manually split the model's vocabulary into set of hypernyms (see Appendix A) and the remaining nouns. We train the models on those 84K image regions that where not named with a hypernym, and test on 8895 image regions that were named with a hypernym in the corpus. We checked that for each of these hypernyms, the vocabulary contains at least one or two names that can be considered as hyponyms, i.e. the model sees objects during training that are instances of *vehicle* for example, but never encounters actual uses of that name. This test set is particularly interesting from an REG perspective, as objects named with very general terms by human speakers are often difficult to describe with more common, but more specific terms, as is illustrated by the uses of *structure* and *thingy* in Figure 1.

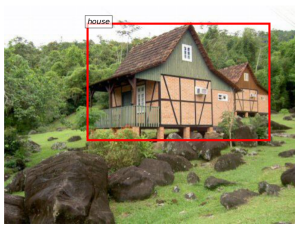
Singulars/Plurals We pick 68 words from our vocabulary that can be grouped into 34 singular-plural noun pairs (see Appendix A). From each pair, we randomly include the singular or plural noun in the set of zero-shot nouns. Thus, we make sure that the model encounters singular and plural names during training, but it never encounters both variants of a name. This results training split of 23K image regions and a test split of 13825 instances.

(a)
wac: seal, rock, water
sim-wap: side, rock,rocks
transfer: rocks, rock, water
combination: rock

(b)
wac: cactus, hut, mountain
sim-wap: side, rock, mountain
transfer: mountain, rocks, rock
combination: hut



(c)
wac: chair, shirt, guy
sim-wap: woman, man, girl
transfer: door, woman, windov
combination: shirt



(d)
wac: roof, house, building
sim-wap: building, house, trees
transfer: building, house, trees
combination: house

(e)
wac: chick, person, guy
sim-wap: man, person, woman
transfer: man, guy, girl
combination: person

(f)
wac: bush, bushes, tree
sim-wap: trees, tree, grass
transfer: trees, tree, bushes
combination: bushes

(g)
wac: animal, lamp, table
sim-wap: man, girl, person
transfer: man, clouds, cloud
combination: person

(h)
wac: post, light, lamp
sim-wap: tree, sky, pole
transfer: tree, sky, trees
combination: lamp

Figure 2: Examples from object naming experiment where model combination is accurate

Zero-shot names	Model	full vocab @1	full vocab @2	full vocab @5	full vocab @10	disjoint vocab @1	disjoint vocab @2
Random	transfer	0.05	2.38	16.57	35.71	41.49	62.34
	wac, project top10	0.00	4.42	21.16	39.17	38.03	58.07
	wac, project top5	0.00	4.39	21.63	40.01	37.46	57.36
	sim-wap	**3.71**	**13.13**	**36.49**	**54.44**	**42.28**	**64.26**
Hypernyms	transfer	0.07	1.25	7.75	29.93	**59.88**	**73.88**
	wac, project top10	0.00	3.01	15.55	36.99	50.51	66.33
	wac, project top5	0.00	2.78	16.75	38.13	47.73	64.38
	sim-wap	**3.16**	**10.33**	**31.14**	**49.62**	57.55	70.15
Singulars/Plurals	transfer	0.01	22.84	44.30	72.85	34.56	51.79
	wac, project top10	0.00	22.21	43.43	68.95	31.46	48.76
	wac, project top5	0.00	22.18	43.93	69.33	31.46	48.88
	sim-wap	**15.39**	**34.73**	**56.62**	**77.32**	**37.24**	**54.02**

Table 4: Accuracies in zero-shot object naming on different vocabulary splits

6.2 Evaluation

Some previous work on zero-shot image labeling assumes additional components that first identify whether an image should be labelled by a known or unknown word (Frome et al., 2013). We follow Lazaridou et al. (2014) and let the model decide whether to refer to an object by a known or unknown name. Related to that, distinct evaluation procedures have been used in the literature on zero-shot learning:

Testing on full vocabulary A realistic way to test zero-shot learning performance is to consider all words from a given vocabulary during testing, though the testset only contains instances of objects that have been named with a 'zero-shot word' (for which no visual instances were seen during training). Accuracies in this setup reflect how well the model is able to generalize, i.e. how often it decides to deviate from the words it was trained on, and (implicitly) predicts that the given object requires a "new" name. In case of the (i) hypernym and (ii) singular/plural test set, this accuracy also reflects to what extent the model is able to detect cases where (i) a more general or vague term is needed, where (ii) an unknown singular/plural counterpart of a known object type occurs.

Testing on disjoint vocabulary Alternatively, the model's vocabulary can be restricted during testing to zero-shot words only, such that names encountered during training and testing are disjoint, see e.g. (Lampert et al., 2009, 2013). This setup factors out the generalization problem, and assesses to what extent a model is able to capture the referential meaning of a word that does not have instances in the training data.

6.3 Results

As compared to Experiment 1 where models achieved similar performance, differences are more pronounced in the zero-shot setup, as shown in Table 4. In particular, we find that the SIM-WAP model which induces individual predictors for words that have not been observed in the training data is clearly more successful than TRANSFER or WAC that project predictions into the distributional space. When tested on the full vocabulary, we find that TRANSFER and WAC very rarely generate names whose referents were excluded from training, which is in line with observations made by Lazaridou et al. (2015a). The SIM-WAP predictors generalize much better, in particular on the singular/plural testset.

An interesting exception is the good performance of the TRANSFER model on the hypernym test set, when evaluated with a disjoint vocabulary. This corroborates evidence from Experiment 1, namely that the transfer model captures taxonomic aspects of object names better than the other models. Projection via individual word classifiers, on the other hand, seems to generalize better than TRANSFER, at least when looking at accuracies @2 ... @10. Thus, combining several vectors predicted by a model of referential word meaning can provide additional information, as compared to mapping an object to a single vector in distributional space. More work is needed to establish how these approaches can be integrated more effectively.

7 Discussion and Conclusion

In this paper, we have investigated models of referential word meaning, using different ways of combining visual information about a word's referent and distributional knowledge about its lexical similarities. Previous cross-modal mapping models essentially force semantically similar objects to be mapped into the same area in the semantic space regardless of their actual visual similarity. We found that cross-modal mapping produces semantically appropriate and mutually highly similar object names in its top-n list, but does not preserve differences in referential word use (e.g. appropriatness of *person* vs. *woman*) especially within the same semantic field. We have shown that it is beneficial for performance in standard and zero-shot object naming to treat words as individual predictors that capture referential appropriateness and are only indirectly linked to a distributional space, either through lexical mapping during application or through cross-modal similarity mapping during training. As we have tested these approaches on a rather small vocabulary, which may limit generality of conclusions, future work will be devoted to scaling up these findings to larger test sets, as e.g. recently collected through conversational agents (Das et al., 2016) that circumvent the need for human-human interaction data. Also from an REG perspective, various extensions of this approach are possible, such as the inclusion of contextual information during object naming and its combination with attribute selection.

Acknowledgments

We acknowledge support by the Cluster of Excellence "Cognitive Interaction Technology" (CITEC; EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG). We thank the anonymous reviewers for their very valuable, very detailed and highly interesting comments.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 238–247. http://www.aclweb.org/anthology/P14-1023.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science* 19(2):233–263.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. 2016. Visual dialog. *CoRR* abs/1611.08669. http://arxiv.org/abs/1611.08669.

Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. 2014. Large-scale object classification using label relation graphs. In *European Conference on Computer Vision*. Springer, pages 48–64.

Jia Deng, W. Dong, Richard Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Katrin Erk. 2016. What do you know about an alligator when you know the company it keeps? *Semantics and Pragmatics* 9(17):1–63. https://doi.org/10.3765/sp.9.17.

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 91–99.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 2121–2129.

Dimitra Gkatzia, Verena Rieser, Phil Bartie, and William Mackaness. 2015. From the virtual to the realworld: Referring to objects in real-world spatial scenes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1936–1942. http://aclweb.org/anthology/D15-1224.

Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. 2006. The IAPR TC-12 benchmark: a new evaluation resource for visual information systems. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2006)*. Genoa, Italy, pages 13–23.

Aurélie Herbelot and Eva Maria Vecchi. 2015. Building a shared world: mapping distributional to model-theoretic semantic spaces. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 22–32. http://aclweb.org/anthology/D15-1003.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2015. Natural language object retrieval. *CoRR* abs/1511.04164. http://arxiv.org/abs/1511.04164.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. Doha, Qatar, pages 787–798.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 36–45. http://www.aclweb.org/anthology/D14-1005.

Satwik Kottur, Ramakrishna Vedantam, José MF Moura, and Devi Parikh. 2016. Visual word2vec (vis-w2v): Learning visually grounded word embeddings using abstract scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 4985–4994.

Emiel Krahmer and Kees Van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics* 38(1):173–218.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Computer Vision and Pattern Recognition*. IEEE, pages 951–958.

Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. 2013. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(3):453–465.

Angeliki Lazaridou, Elia Bruni, and Marco Baroni. 2014. Is this a wampimuk? Cross-modal mapping between distributional semantics and the visual world. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 1403–1414.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015a. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 270–280. http://www.aclweb.org/anthology/P15-1027.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015b. Combining language and vision with a multimodal skip-gram model. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 153–163. http://www.aclweb.org/anthology/N15-1016.

Willem JM Levelt, Herbert Schriefers, Dirk Vorberg, Antje S Meyer, Thomas Pechmann, and Jaap Havinga. 1991. The time course of lexical access in speech production: A study of picture naming. *Psychological review* 98(1):122.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2015. Generation and comprehension of unambiguous object descriptions. *ArXiv / CoRR* abs/1511.02283. http://arxiv.org/abs/1511.02283.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. Curran Associates Inc., USA, NIPS'13, pages 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959.

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. 2013. Zero-shot learning by convex combination of semantic embeddings. *International Conference on Learning Representations (ICLR)* .

Vicente Ordonez, Wei Liu, Jia Deng, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. 2016. Learning to name objects. *Commun. ACM* 59(3):108–115.

Eleanor Rosch. 1978. Principles of Categorization. In Eleanor Rosch and Barbara B. Lloyd, editors, *Cognition and Categorization*, Lawrence Erlbaum, Hillsdale, N.J., USA, pages 27—-48.

Deb Roy. 2005. Grounding words in perception and action: Computational insights. *Trends in Cognitive Sciene* 9(8):389–396.

Deb Roy, Peter Gorniak, Niloy Mukherjee, and Josh Juster. 2002. A trainable spoken language understanding system for visual object selection. In *Proceedings of the International Conference on Speech and Language Processing 2002 (ICSLP 2002)*. Colorado, USA.

Deb K. Roy. 2002. Learning visually-grounded words and syntax for a scene description task. *Computer Speech and Language* 16(3).

David Schlangen, Sina Zarriess, and Casey Kennington. 2016. Resolving references to objects in photographs using the words-as-classifiers model. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 721–732. http://www.aclweb.org/anthology/P14-1068.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*. pages 935–943.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR 2015*. Boston, MA, USA.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37(1):141–188.

Sina Zarrieß and David Schlangen. 2016. Easy things first: Installments improve referring expression generation for objects in photographs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 610–620. http://www.aclweb.org/anthology/P16-1058.

Sina Zarrieß and David Schlangen. 2017. Is this a child, a girl or a car? exploring the contribution of distributional similarity to learning referential word meanings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, pages 86–91. http://aclweb.org/anthology/E17-2014.

A Vocabulary Splits for Zero-Shot Naming

Hypernyms animal, animals, plant, plants, vehicle, person, persons, food, thing, object, area, things, thingy, toy, anyone, clothes, dish, building, land, structure, item, water

Singulars/Plurals . . .

. . . training on instances of: animals, plants, cars, people, buildings, trees, man, kid, guy, girl, boy, flower, bird, hill, orange, cloud, curtain, window, shrub, apple, light, house, glass, bottle, dude, leg, book, wall, bananas, carrots, pillows, bushes, mountains, bags

. . . testing on instances of: animal, plant, car, person, building, tree, men, kids, guys, girls, boys, flowers, birds, hills, oranges, clouds, curtains, windows, shrubs, apples, lights, houses, glasses, bottles, dudes, legs, books, walls, banana, carrot, pillow, bush, mountain, bag

FOIL it! Find One mismatch between Image and Language caption

Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich,
Aurélie Herbelot, Moin Nabi, Enver Sangineto, Raffaella Bernardi
University of Trento
{firstname.lastname}@unitn.it

Abstract

In this paper, we aim to understand whether current language and vision (LaVi) models truly grasp the interaction between the two modalities. To this end, we propose an extension of the MS-COCO dataset, FOIL-COCO, which associates images with both correct and 'foil' captions, that is, descriptions of the image that are highly similar to the original ones, but contain one single mistake ('foil word'). We show that current LaVi models fall into the traps of this data and perform badly on three tasks: a) caption classification (correct vs. foil); b) foil word detection; c) foil word correction. Humans, in contrast, have near-perfect performance on those tasks. We demonstrate that merely utilising language cues is not enough to model FOIL-COCO and that it challenges the state-of-the-art by requiring a fine-grained understanding of the relation between text and image.

1 Introduction

Most human language understanding is grounded in perception. There is thus growing interest in combining information from language and vision in the NLP and AI communities.

So far, the primary testbeds of Language and Vision (LaVi) models have been 'Visual Question Answering' (VQA) (e.g. Antol et al. (2015); Malinowski and Fritz (2014); Malinowski et al. (2015); Gao et al. (2015); Ren et al. (2015)) and 'Image Captioning' (IC) (e.g. Hodosh et al. (2013); Fang et al. (2015); Chen and Lawrence Zitnick (2015); Donahue et al. (2015); Karpathy and Fei-Fei (2015); Vinyals et al. (2015)). Whilst some models have seemed extremely successful on those tasks, it remains unclear how the reported results should be interpreted and what those models are actually learning. There is an emerging feeling in the community that the VQA task should be revisited, especially as many current dataset can be handled by 'blind' models which use language input only, or by simple concatenation of language and vision features (Agrawal et al., 2016; Jabri et al., 2016; Zhang et al., 2016; Goyal et al., 2016a). In IC too, Hodosh and Hockenmaier (2016) showed that, contrarily to what prior research had suggested, the task is far from been solved, since IC models are not able to distinguish between a correct and incorrect caption.

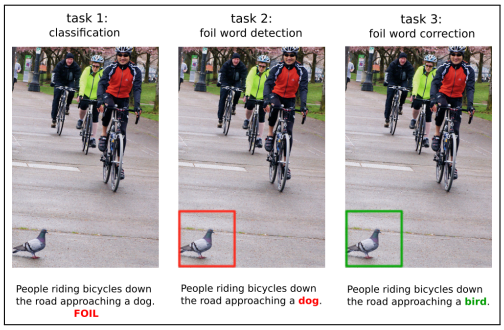


Figure 1: Is the caption correct or foil (T1)? If it is foil, where is the mistake (T2) and which is the word to correct the foil one (T3)?

Such results indicate that in current datasets, *language provides priors* that make LaVi models successful without truly understanding and integrating language and vision. But problems do not stop at biases. Johnson et al. (2016) also point out that current data 'conflate multiple sources of error, making it hard to pinpoint model weaknesses', thus highlighting the need for *diagnostic datasets*. Thirdly, existing IC *evaluation metrics* are sensitive to n-gram overlap and there is a need for measures that better simulate human judgments (Hodosh et al., 2013; Elliott and Keller, 2014; Anderson et al., 2016).

Our paper tackles the identified issues by proposing an automatic method for creating a

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 255–265
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1024

large dataset of real images with *minimal language bias* and some *diagnostic* abilities. Our dataset, FOIL (Find One mismatch between Image and Language caption),[1] consists of images associated with incorrect captions. The captions are produced by introducing one single error (or 'foil') per caption in existing, human-annotated data (Figure 1). This process results in a challenging error-detection/correction setting (because the caption is 'nearly' correct). It also provides us with a ground truth (we know where the error is) that can be used to objectively measure the performance of current models.

We propose three tasks based on widely accepted evaluation measures: we test the ability of the system to a) compute whether a caption is compatible with the image (T1); b) when it is incompatible, highlight the mismatch in the caption (T2); c) correct the mistake by replacing the foil word (T3).

The dataset presented in this paper (Section 3) is built on top of MS-COCO (Lin et al., 2014), and contains 297,268 datapoints and 97,847 images. We will refer to it as FOIL-COCO. We evaluate two state-of-the-art VQA models: the popular one by Antol et al. (2015), and the attention-based model by Lu et al. (2016), and one popular IC model by (Wang et al., 2016). We show that those models perform close to chance level, while humans can perform the tasks accurately (Section 4). Section 5 provides an analysis of our results, allowing us to diagnose three failures of LaVi models. First, their coarse representations of language and visual input do not encode suitably structured information to spot mismatches between an utterance and the corresponding scene (tested by T1). Second, their language representation is not fine-grained enough to identify the part of an utterance that causes a mismatch with the image as it is (T2). Third, their visual representation is also too poor to spot and name the visual area that corresponds to a captioning error (T3).

2 Related Work

The image captioning (IC) and visual question answering (VQA) tasks are the most relevant to our work. In IC (Fang et al., 2015; Chen and Lawrence Zitnick, 2015; Donahue et al., 2015; Karpathy and Fei-Fei, 2015; Vinyals et al., 2015; Wang et al., 2016), the goal is to generate a caption for a given image, such that it is both semantically and syntactically correct, and properly describes the content of that image. In VQA (Antol et al., 2015; Malinowski and Fritz, 2014; Malinowski et al., 2015; Gao et al., 2015; Ren et al., 2015), the system attempts to answer open-ended questions related to the content of the image. There is a wealth of literature on both tasks, but we only discuss here the ones most related to our work and refer the reader to the recent surveys by (Bernardi et al., 2016; Wu et al., 2016).

Despite their success, it remains unclear whether state-of-the-art LaVi models capture vision and language in a truly integrative fashion. We could identify three types of arguments surrounding the high performance of LaVi models:

(i) Triviality of the LaVi tasks: Recent work has shown that LaVi models heavily rely on language priors (Ren et al., 2015; Agrawal et al., 2016; Kafle and Kanan, 2016). Even simple correlation and memorisation can result in good performance, without the underlying models truly understanding visual content (Zhou et al., 2015; Jabri et al., 2016; Hodosh and Hockenmaier, 2016). Zhang et al. (2016) first unveiled that there exists a huge bias in the popular VQA dataset by Antol et al. (2015): they showed that almost half of all the questions in this dataset could be answered correctly by using the question alone and ignoring the image completely. In the same vein, Zhou et al. (2015) proposed a simple baseline for the task of VQA. This baseline simply concatenates the Bag of Words (BoW) features from the question and Convolutional Neural Networks (CNN) features from the image to predict the answer. They showed that such a simple method can achieve comparable performance to complex and deep architectures. Jabri et al. (2016) proposed a similar model for the task of multiple choice VQA, and suggested a cross-dataset generalization scheme as an evaluation criterion for VQA systems. We complement this research by introducing three new tasks with different levels of difficulty, on which LaVi models can be evaluated sequentially.

(ii) Need for diagnostics: To overcome the bias uncovered in previous datasets, several research groups have started proposing tasks which involve distinguishing distractors from a ground-truth caption for an image. Zhang et al. (2016) introduced a binary VQA task along with a dataset

[1]The dataset is available from https://foilunitn.github.io/

composed of sets of similar artificial images, allowing for more precise diagnostics of a system's errors. Goyal et al. (2016a) balanced the dataset of Antol et al. (2015), collecting a new set of complementary natural images which are similar to existing items in the original dataset, but result in different answers to a common question. Hodosh and Hockenmaier (2016) also proposed to evaluate a number of state-of-the-art LaVi algorithms in the presence of distractors. Their evaluation was however limited to a small dataset (namely, Flickr30K (Young et al., 2014)) and the caption generation was based on a hand-crafted scheme using only inter-dataset distractors.

Most related to our paper is the work by Ding et al. (2016). Like us, they propose to extend the MS-COCO dataset by generating decoys from human-created image captions. They also suggest an evaluation apparently similar to our T1, requiring the LaVi system to detect the true target caption amongst the decoys. Our efforts, however, differ in some substantial ways. First, their technique to create incorrect captions (using BLEU to set an upper similarity threshold) is so that many of those captions will differ from the gold description in more than one respect. For instance, the caption *two elephants standing next to each other in a grass field* is associated with the decoy *a herd of giraffes standing next to each other in a dirt field* (errors: *herd*, *giraffe*, *dirt*) or with *animals are gathering next to each other in a dirt field* (error: *dirt*; infelicities: *animals* and *gathering*, which are both pragmatically odd). Clearly, the more the caption changes in the decoy, the easier the task becomes. In contrast, the foil captions we propose only differ from the gold description by *one* word and are thus more challenging. Secondly, the automatic caption generation of Ding et al means that 'correct' descriptions can be produced, resulting in some confusion in human responses to the task. We made sure to prevent such cases, and human performance on our dataset is thus close to 100%. We note as well that our task does not require any complex instructions for the annotation, indicating that it is intuitive to human beings (see §4). Thirdly, their evaluation is a multiple-choice task, where the system has to compare all captions to understand which one is *closest* to the image. This is arguably a simpler task than the one we propose, where a caption is given and the system is asked to classify it as correct or foil: as we show in §4, detecting a *correct* caption is much easier than detecting foils. So evaluating precision on both gold and foil items is crucial.

Finally, (Johnson et al., 2016) proposed CLEVR, a dataset for the diagnostic evaluation of VQA systems. This dataset was designed with the explicit goal of enabling detailed analysis of different aspects of visual reasoning, by minimising dataset biases and providing rich ground-truth representations for both images and questions.

(iii) Lack of objective evaluation metrics: The evaluation of Natural Language Generation (NLG) systems is known to be a hard problem. It is further unclear whether the quality of LaVi models should be measured using metrics designed for language-only tasks. Elliott and Keller (2014) performed a sentence-level correlation analysis of NLG evaluation measures against expert human judgements in the context of IC. Their study revealed that most of those metrics were only weakly correlated with human judgements. In the same line of research, Anderson et al. (2016) showed that the most widely-used metrics for IC fail to capture semantic propositional content, which is an essential component of human caption evaluation. They proposed a semantic evaluation metric called SPICE, that measures how effectively image captions recover objects, attributes and the relations between them. In this paper, we tackle this problem by proposing tasks which can be evaluated based on objective metrics for classification/detection error.

3 Dataset

In this section, we describe how we automatically generate FOIL-COCO datapoints, i.e. image, original and foil caption triples. We used the training and validation Microsoft's Common Objects in Context (MS-COCO) dataset (Lin et al., 2014) (2014 version) as our starting point. In MS-COCO, each image is described by at least five descriptions written by humans via Amazon Mechanical Turk (AMT). The images contains 91 common object categories (e.g. *dog, elephant, bird, ...* and *car, bicycle, airplane, ...*), from 11 supercategories (*Animal*, *Vehicle*, resp.), with 82 of them having more than 5K labeled instances. In total there are 123,287 images with captions (82,783 for training and 40,504 for validation).[2]

Our data generation process consists of four

[2]The MS-COCO test set is not available for download.

	nr. of datapoints	nr. unique images	nr. of tot. captions	nr. target::foil pairs
Train	197,788	65,697	395,576	256
Test	99,480	32,150	198,960	216

Table 1: Composition of FOIL-COCO.

main steps, as described below. The last two steps are illustrated in Figure 2.

1. Generation of replacement word pairs We want to replace one noun in the original caption (the *target*) with an incorrect but similar word (the *foil*). To do this, we take the labels of MS-COCO categories, and we pair together words belonging to the same supercategory (e.g., bicycle::motorcycle, bicycle::car, bird::dog). We use as our vocabulary 73 out of the 91 MS-COCO categories, leaving out those categories that are multi-word expressions (e.g. traffic light). We thus obtain 472 target::foil pairs.

2. Splitting of replacement pairs into training and testing To avoid the models learning trivial correlations due to replacement frequency, we randomly split, within each supercategory, the candidate target::foil pairs which are used to generate the captions of the training vs. test sets. We obtain 256 pairs, built out of 72 target and 70 foil words, for the training set, and 216 pairs, containing 73 target and 71 foil words, for the test set.

3. Generation of foil captions We would like to generate foil captions by replacing only target words which refer to *visually salient objects*. To this end, given an image, we replace only those target words that occur in more than one MS-COCO caption associated with that image. Moreover, we want to use foils which are *not visually present*, i.e. that refer to visual content not present in the image. Hence, given an image, we only replace a word with foils that are not among the labels (objects) annotated in MS-COCO for that image. We use the images from the MS-COCO training and validation sets to generate our training and test sets, respectively. We obtain 2,229,899 for training and 1,097,012 captions for testing.

4. Mining the hardest foil caption for each image To eliminate possible visual-language dataset bias, out of all foil captions generated in step 3, we select only the hardest one. For this purpose, we need to model the visual-language bias of the dataset. To this end, we use Neuraltalk[3] (Karpathy and Fei-Fei, 2015), one of the state-of-the-art image captioning systems, pre-trained on MS-COCO. Neuraltalk is based on an LSTM which takes as input an image and generates a sentence describing its content. We obtain a neural network $\mathcal{N}$ that implicitly represents the visual-language bias through its weights. We use $\mathcal{N}$ to approximate the conditional probability of a caption C given a dataset T and and an image I ($P(C|I,T)$). This is obtained by simply using the loss $l(C,\mathcal{N}(I))$ i.e., the error obtained by comparing the pseudo-ground truth C with the sentence predicted by $\mathcal{N}$: $P(C|I,T) = 1 - l(C,\mathcal{N}(I))$ (we refer to (Karpathy and Fei-Fei, 2015) for more details on how $l()$ is computed). $P(C|I,T)$ is used to select the hardest foil among all the possible foil captions, i.e. the one with the highest probability according to the dataset bias learned by $\mathcal{N}$. Through this process, we obtain 197,788 and 99,480 original::foil caption pairs for the training and test sets, respectively. None of the target::foil word pairs are filtered out by this mining process.

The final FOIL-COCO dataset consists of 297,268 datapoints (197,788 in training and 99,480 in test set). All the 11 MS-COCO supercategories are represented in our dataset and contain 73 categories from the 91 MS-COCO ones (4.8 categories per supercategory on average.) Further details are reported in Table 1.

4 Experiments and Results

We conduct three tasks, as presented below:

Task 1 (T1): Correct vs. foil classification Given an image and a caption, the model is asked to mark whether the caption is correct or wrong. The aim is to understand whether LaVi models can spot mismatches between their coarse representations of language and visual input.

Task 2 (T2): Foil word detection Given an image and a foil caption, the model has to detect the foil word. The aim is to evaluate the understanding of the system at the word level. In order to systematically check the system's performance with different prior information, we test two different set-

[3] https://github.com/karpathy/neuraltalk

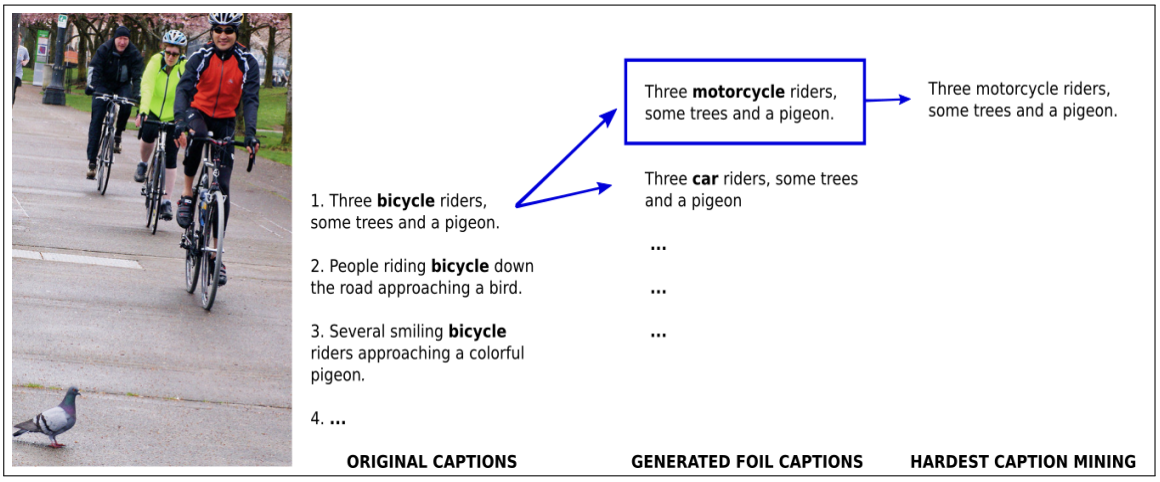


Figure 2: The main aspects of the foil caption generation process. Left column: some of the original COCO captions associated with an image. In bold we highlight one of the target words (bicycle), chosen because it is mentioned by more than one annotator. Middle column: For each original caption and each chosen target word, different foil captions are generated by replacing the target word with all possible candidate foil replacements. Right column: A single caption is selected amongst all foil candidates. We select the 'hardest' caption, according to Neuraltalk model, trained using only the original captions.

tings: the foil has to be selected amongst (a) only the nouns or (b) all content words in the caption.

Task 3 (T3): Foil word correction Given an image, a foil caption and the foil word, the model has to detect the foil and provide its correction. The aim is to check whether the system's visual representation is fine-grained enough to be able to extract the information necessary to correct the error. For efficiency reasons, we operationalise this task by asking models to select a correction from the set of target words, rather than the whole dataset vocabulary (viz. more than 10K words).

4.1 Models

We evaluate both VQA and IC models against our tasks. For the former, we use two of the three models evaluated in (Goyal et al., 2016a) against a balanced VQA dataset. For the latter, we use the multimodal bi-directional LSTM, proposed in (Wang et al., 2016), and adapted for our tasks.

LSTM + norm I: We use the best performing VQA model in (Antol et al., 2015) (deeper LSTM + norm I). This model uses a two stack Long-Short Term Memory (LSTM) to encode the questions and the last fully connected layer of VGGNet to encode images. Both image embedding and caption embedding are projected into a 1024-dimensional feature space. Following (Antol et al., 2015), we have normalised the image feature before projecting it. The combination of these two projected embeddings is performed by a point-wise multiplication. The multi-model representation thus obtained is used for the classification, which is performed by a multi-layer perceptron (MLP) classifier.

HieCoAtt: We use the Hierarchical Co-Attention model proposed by (Lu et al., 2016) that co-attends to both the image and the question to solve the task. In particular, we evaluate the 'alternate' version, i.e. the model that sequentially alternates between generating some attention over the image and question. It does so in a hierarchical way by starting from the word-level, then going to the phrase and then to the entire sentence-level. These levels are combined recursively to produce the distribution over the foil vs. correct captions.

IC-Wang: Amongst the IC models, we choose the multimodal bi-directional LSTM (Bi-LSTM) model proposed in (Wang et al., 2016). This model predicts a word in a sentence by considering both the past and future context, as sentences are fed to the LSTM in forward and backward order. The model consists of three modules: a CNN for encoding image inputs, a Text-LSTM (T-LSTM) for encoding sentence inputs, a Multimodal LSTM (M-LSTM) for embedding visual and textual vectors to a common semantic space and decoding to sentence. The bidirectional LSTM is implemented with two separate LSTM layers.

Baselines: We compare the SoA models above against two baselines. For the classification task, we use a **Blind** LSTM model followed by a fully connected layer and softmax and train it only on captions as input to predict the answer. In addition, we evaluate the **CNN+LSTM** model, where visual and textual features are simply concatenated.

The models at work on our three tasks For the *classification task* (T1), the baselines and VQA models can be applied directly. We adapt the generative IC model to perform the classification task as follows. Given a test image I and a test caption, for each word w_t in the test caption, we remove the word and use the model to generate new captions in which the w_t has been replaced by the word v_t predicted by the model (w_1,...,w_{t-1}, v_t, w_{t-1},...,w_n). We then compare the conditional probability of the test caption with all the captions generated from it by replacing w_t with v_t. When all the conditional probabilities of the generated captions are lower than the one assigned to the test caption the latter is classified as good, otherwise as foil. For the other tasks, the models have been trained on T1. To perform the *foil word detection task* (T2), for the VQA models, we apply the occlusion method. Following (Goyal et al., 2016b), we systematically occlude subsets of the language input, forward propagate the masked input through the model, and compute the change in the probability of the answer predicted with the unmasked original input. For the IC model, similarly to T1, we sequentially generate new captions from the foil one by replacing, one by one, the words in it and computing the conditional probability of the foil caption and the one generated from it. The word whose replacement generate the caption with the highest conditional probabilities is taken to be the foil word. Finally, to evaluate the models on the *error correction task* (T3), we apply the linear regression method over all the target words and select the target word which has the highest probability of making that wrong caption correct with respect to the given image.

Upper-bound Using Crowdflower, we collected human answers from 738 native English speakers for 984 image-caption pairs randomly selected from the test set. Subjects were given an image and a caption and had to decide whether it was correct or wrong (T1). If they thought it was wrong, they were required to mark the error in the caption (T2). We collected 2952 judgements (i.e. 3 judgements per pair and 4 judgements per rater) and computed human accuracy in T1 when considering as answer (a) the one provided by at least 2 out of 3 annotators (*majority*) and (b) the one provided by all 3 annotators (*unanimity*). The same procedure was adopted for computing accuracies in T2. Accuracies in both T1 an T2 are reported in Table 2. As can be seen, in the *majority* setting annotators are quasi-perfect in classifying captions (92.89%) and detecting foil words (97.00%). Though lower, accuracies in the *unanimity* setting are still very high, with raters providing the correct answer in 3 out of 4 cases in both tasks. Hence, although we have collected human answers only on a rather small subset of the test set, we believe their results are representative of how easy the tasks are for humans.

4.2 Results

As shown in Table 2, the FOIL-COCO dataset is challenging. On T1, for which the chance level is 50.00%, the 'blind', language-only model, does badly with an accuracy of 55.62% (25.04% on foil captions), demonstrating that language bias is minimal. By adding visual information, CNN+LSTM, the overall accuracy increases by 5.45% (7.94% on foil captions.) reaching 61.07% (resp. 32.98%). Both SoA VQA and IC models do significantly worse than humans on both T1 and T2. The VQA systems show a strong bias towards correct captions and poor overall performance. They only identify 34.51% (LSTM +norm I) and 36.38% (HieCoAtt) of the incorrect captions (T1). On the other hand, the IC model tends to be biased toward the foil captions, on which it achieves an accuracy of 45.44%, higher than the VQA models. But the overall accuracy (42.21%) is poorer than the one obtained by the two baselines. On the foil word detection task, when considering only nouns as possible foil word, both the IC and the LSTM+norm I models perform close to chance level, and the HieCoAtt performs somewhat better, reaching 38.79%. Similar results are obtained when considering all words in the caption as possible foil. Finally, the VQA models' accuracy on foil word correction (T3) is extremely low, at 4.7% (LSTM +norm I) and 4.21% (HieCoAtt). The result on T3 makes it clear that the VQA systems are unable to extract from the image rep-

resentation the information needed to correct the foil: despite being told which element in the caption is wrong, they are not able to zoom into the correct part of the image to provide a correction, or if they are, cannot name the object in that region. The IC model performs better compared to the other models, having an accuracy that is 20,78% higher than chance level.

T1: Classification task			
	Overall	Correct	Foil
Blind	55.62	86.20	25.04
CNN+LSTM	61.07	89.16	32.98
IC-Wang	42.21	38.98	45.44
LSTM + norm I	63.26	**92.02**	34.51
HieCoAtt	**64.14**	91.89	**36.38**
Human (*majority*)	92.89	91.24	94.52
Human (*unanimity*)	76.32	73.73	78.90

T2: Foil word detection task		
	nouns	all content words
Chance	23.25	15.87
IC-Wang	27.59	23.32
LSTM + norm I	26.32	24.25
HieCoAtt	**38.79**	**33.69**
Human (*majority*)		97.00
Human (*unanimity*)		73.60

T3: Foil word correction task	
	all target words
Chance	1.38
IC-Wang	**22.16**
LSTM + norm I	4.7
HieCoAtt	4.21

Table 2: **T1:** Accuracy for the *classification* task, relatively to all image-caption pairs (overall) and by type of caption (correct vs. foil); **T2:** Accuracy for the *foil word detection* task, when the foil is known to be among the nouns only or when it is known to be among all the content words; **T3:** Accuracy for the *foil word correction* task when the correct word has to be chosen among any of the target words.

5 Analysis

We performed a mixed-effect logistic regression analysis in order to check whether the behavior of the best performing models in T1, namely the VQA models, can be predicted by various linguistic variables. We included: 1) semantic similarity between the original word and the foil (computed as the cosine between the two corresponding `word2vec` embeddings (Mikolov et al., 2013)); 2) frequency of original word in FOIL-COCO captions; 3) frequency of the foil word in FOIL-COCO captions; 4) length of the caption (number of words). The mixed-effect model was performed to get rid of possible effects due to either object supercategory (indoor, food, vehicle, etc.) or target::foil pair (e.g., zebra::giraffe, boat::airplane, etc.). For both LSTM + norm I and HieCoAtt, `word2vec` similarity, frequency of the original word, and frequency of the foil word turned out to be highly reliable predictors of the model's response. The higher the values of these variables, the more the models tend to provide the wrong output. That is, when the foil word (e.g. *cat*) is semantically very similar to the original one (e.g. *dog*), the models tend to wrongly classify the caption as 'correct'. The same holds for frequency values. In particular, the higher the frequency of both the original word and the foil one, the more the models fail. This indicates that systems find it difficult to distinguish related concepts at the text-vision interface, and also that they may tend to be biased towards frequently occurring concepts, 'seeing them everywhere' even when they are not present in the image. Caption length turned out to be only a partially reliable predictor in the LSTM + norm I model, whereas it is a reliable predictor in HieCoAtt. In particular, the longer the caption, the harder for the model to spot that there is a foil word that makes the caption wrong.

As revealed by the fairly high variance explained by the random effect related to target::foil pairs in the regression analysis, both models perform very well on some target::foil pairs, but fail on some others (see leftmost part of Table 4 for same examples of easy/hard target::foil pairs). Moreover, the variance explained by the random effect related to object supercategory is reported in Table 3. As can be seen, for some supercategories accuracies are significatively higher than for others (compare, e.g., 'electronic' and 'outdoor').

In a separate analysis, we also checked whether there was any correlation between results and the position of the foil in the sentence, to ensure the models did not profit from any undesirable artifacts of the data. We did not find any such correlation.

Super-category	No. of object	No. of foil captions	Acc. using LSTM + norm I	Acc. using HieCoAtt
outdoor	2	107	2.80	0.93
food	9	10407	22.00	26.59
indoor	6	4911	30.74	27.97
appliance	5	2811	32.72	34.54
sports	10	16276	31.57	31.61
animal	10	21982	39.03	43.18
vehicle	8	16514	34.38	40.09
furniture	5	13625	33.27	33.13
accessory	5	3040	49.53	31.80
electronic	6	5615	45.82	43.47
kitchen	7	4192	38.19	45.34

Table 3: Classification Accuracy of foil captions by Super Categories (T1). The No. of the objects and the No. of foil captions refer to the test set. The training set has a similar distribution.

Top-5		Bottom-5		Top-5		Bottom-5	
T1: LSTM + norm I				T2: LSTM + norm I			
racket::glove	100	motorcycle::airplane	0	drier::scissors	100	glove::skis	0
racket::kite	97.29	bicycle::airplane	0	zebra::giraffe	88.98	snowboard::racket	0
couch::toilet	97.11	drier::scissors	0	boat::airplane	87.87	donut::apple	0
racket::skis	95.23	bus::airplane	0.35	truck::airplane	85.71	glove::surfboard	0
giraffe::sheep	95.09	zebra::giraffe	0.43	train::airplane	81.93	spoon::bottle	0
T1: HieCoAtt				T2: HieCoAtt			
tie::handbag	100	drier::scissors	0	zebra::elephant	94.92	drier::scissors	0
snowboard::glove	100	fork::glass	0	backpack::handbag	94.44	handbag::tie	0
racket::skis	100	handbag::tie	0	cow::zebra	93.33	broccoli:orange	1.47
racket::glove	100	motorcycle::airplane	0	bird::sheep	93.11	zebra::giraffe	1.96
backpack::handbag	100	train::airplane	0	orange::carrot	92.37	boat::airplane	2.09

Table 4: Easiest and hardest target::foil pairs: T1 (caption classification) and T2 (foil word detection).

To better understand results on T2, we performed an analysis investigating the performance of the VQA models on different target::foil pairs. As reported in Table 4 (right), both models perform nearly perfectly with some pairs and very badly with others. At first glance, it can be noticed that LSTM + norm I is very effective with pairs involving vehicles (*airplane*, *truck*, etc.), whereas HieCoAtt seems more effective with pairs involving animate nouns (i.e. animals), though more in depth analysis is needed on this point. More interestingly, some pairs that are found to be predicted almost perfectly by LSTM + I norm, namely boat::airplane, zebra::giraffe, and drier::scissors, turn out to be among the Bottom-5 cases in HieCoAtt. This suggests, on the one hand, that the two VQA models use different strategies to perform the task. On the other hand, it shows that our dataset does not contain cases that are a priori easy for any model.

The results of IC-Wang on T3 are much higher than LSTM + norm I and HieCoAtt, although it is outperformed by or is on par with HieCoAtton on T1-T2. Our interpretation is that this behaviour is related to the discriminative/generative nature of our tasks. Specifically, T1 and T2 are discriminative tasks and LSTM + norm I and HieCoAtt are discriminative models. Conversely, T3 is a generative task (a word needs to be generated) and IC-Wang is a generative model. It would be interesting to test other IC models on T3 and compare their results against the ones reported here. However, note that IC-Wang is 'tailored' for T3 because it takes as input the whole sentence (minus the word to be generated), while common sequential IC approaches can only generate a word depending on the previous words in the sentence.

As far as human performance is concerned, both T1 and T2 turn out to be extremely easy. In T1, image-caption pairs were correctly judged as correct/wrong in overall 914 out of 984 cases (92.89%) in the *majority* setting. In the *unanim-*

ity setting, the correct response was provided in 751 out of 984 cases (76.32%). Judging foil captions turns out to be slightly easier than judging correct captions in both settings, probably due to the presence of typos and misspellings that sometimes occur in the original caption (e.g. raters judge as wrong the original caption *People playing ball with a drown and white dog*, where 'brown' was misspelled as 'drown'). To better understand which factors contribute to make the task harder, we qualitatively analyse those cases where all annotators provided a wrong judgement for an image-caption pair. As partly expected, almost all cases where original captions (thus correct for the given image) are judged as being wrong are cases where the original caption is indeed incorrect. For example, a caption using the word 'motorcycle' to refer to a bicycle in the image is judged as wrong. More interesting are those cases where all raters agreed in considering as correct image-caption pairs that are instead foil. Here, it seems that vagueness as well as certain metaphorical properties of language are at play: human annotators judged as correct a caption describing *Blue and banana large birds on tree with metal pot* (see Fig 3, left), where 'banana' replaced 'orange'. Similarly, all raters judged as correct the caption *A cat laying on a bed next to an opened keyboard* (see Fig 3, right), where the cat is instead laying next to an opened laptop.

Focusing on T2, it is interesting to report that among the correctly-classified foil cases, annotators provided the target word in 97% and 73.6% of cases in the *majority* and *unanimity* setting, respectively. This further indicates that finding the foil word in the caption is a rather trivial task for humans.

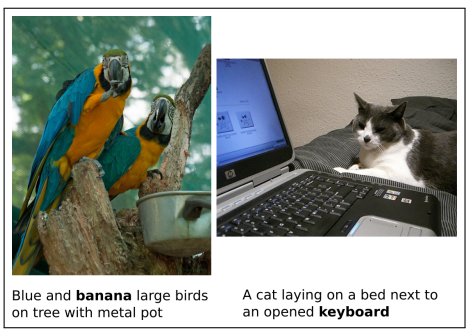


Figure 3: Two cases of foil image-caption pairs that are judged as correct by all annotators.

6 Conclusion

We have introduced FOIL-COCO, a large dataset of images associated with both correct and foil captions. The error production is automatically generated, but carefully thought out, making the task of spotting foils particularly challenging. By associating the dataset with a series of tasks, we allow for diagnosing various failures of current LaVi systems, from their coarse understanding of the correspondence between text and vision to their grasp of language and image structure.

Our hypothesis is that systems which, like humans, deeply integrate the language and vision modalities, should spot foil captions quite easily. The SoA LaVi models we have tested fall through that test, implying that they fail to integrate the two modalities. To complete the analysis of these results, we plan to carry out a further task, namely ask the system to detect in the image the area that produces the mismatch with the foil word (the red box around the bird in Figure 1.) This extra step would allow us to fully diagnose the failure of the tested systems and confirm what is implicit in our results from task 3: that the algorithms are unable to map particular elements of the text to their visual counterparts. We note that the addition of this extra step will move this work closer to the textual/visual explanation research (e.g., (Park et al., 2016; Selvaraju et al., 2016)). We will then have a pipeline able to not only test whether a mistake can be detected, but also whether the system can explain its decision: 'the wrong word is *dog* because the cyclists are in fact approaching a bird, there, in the image'.

LaVi models are a great success of recent research, and we are impressed by the amount of ideas, data and models produced in this stimulating area. With our work, we would like to push the community to think of ways that models can better merge language and vision modalites, instead of merely using one to supplement the other.

Acknowledgments

We are greatful to the Erasmus Mundus European Master in Language and Communication Technologies (EM LCT) for the scholarship provided to the third author. Moreover, we gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used in our research.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the behavior of visual question answering models. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: Semantic Propositional Image Caption Evaluation. In *In Proceedings of the European Conference on Computer Vision (ECCV)*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*. https://github.com/VT-vision-lab/VQA_LSTM_CNN.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *J. Artif. Intell. Res.(JAIR)* 55:409–442.

Xinlei Chen and C Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 2422–2431.

Nan Ding, Sebastian Goodman, Fei Sha, and Radu Soricut. 2016. Understanding image and text simultaneously: a dual vision-language machine comprehension task. *arXiv preprint arXiv:1612.07833* .

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 2625–2634.

Desmond Elliott and Frank Keller. 2014. Comparing automatic evaluation measures for image description. In *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Short Papers*. pages 452–457.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 1473–1482.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. 2015. Are you talking to a machine? dataset and methods for multilingual image question. In *Advances in Neural Information Processing Systems*. pages 2296–2304.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016a. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. *arXiv preprint arXiv:1612.00837* .

Yash Goyal, Akrit Mohapatra, Devi Parikh, and Dhruv Batra. 2016b. Towards Transparent AI Systems: Interpreting Visual Question Answering Models . In *In Proceedings of ICML Visualization Workshop*.

Micah Hodosh and Julia Hockenmaier. 2016. Focused evaluation for image description with binary forced-choice tasks. In *Proceedings of the 5th Workshop on Vision and Language (VL'16)*.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research* 47:853–899.

Allan Jabri, Armand Joulin, and Laurens van der Maaten. 2016. Revisiting visual question answering baselines. In *Proceedings of the European Conference on Computer Vision (ECCV)*. pages 727–739.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2016. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. ArXiv:1612.06890.

Kushal Kafle and Christopher Kanan. 2016. Visual question answering: Datasets, algorithms, and future challenges. *arXiv preprint arXiv:1610.01465* .

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3128–3137.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*. Springer, pages 740–755.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *In Proceedings of NIPS 2016*. https://github.com/jiasenlu/HieCoAttenVQA.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*. pages 1682–1690.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2015. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 1–9.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Trevor Darrell Bernt Schiele, and Marcus Rohrbach. 2016. Attentive explanations: Justifying decisions and pointing to the evidence. ArXiv:1612.04757.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems (NIPS 2015)*.

Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2016. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. ArXiv:1610.02391v2.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3156–3164.

Cheng Wang, Haojin Yang, Christian Bartz, and Christoph Meinel. 2016. Image captioning with deep bidirectional LSTMs. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, pages 988–997.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2016. Visual question answering: A survey of methods and datasets. *arXiv preprint arXiv:1607.05910* .

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* 2:67–78.

Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Yin and yang: Balancing and answering binary visual questions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 5014–5022.

Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2015. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167* .

VERB PHYSICS: Relative Physical Knowledge of Actions and Objects

Maxwell Forbes **Yejin Choi**
Paul G. Allen School of Computer Science & Engineering
University of Washington
{mbforbes,yejin}@cs.washington.edu

Abstract

Learning commonsense knowledge from natural language text is nontrivial due to *reporting bias*: people rarely state the obvious, e.g., "My house is *bigger* than me." However, while rarely stated explicitly, this trivial everyday knowledge does influence the way people talk about the world, which provides indirect clues to reason about the world. For example, a statement like, "Tyler *entered* his house" implies that his house is *bigger* than Tyler.

In this paper, we present an approach to infer relative physical knowledge of actions and objects along five dimensions (e.g., size, weight, and strength) from unstructured natural language text. We frame knowledge acquisition as joint inference over two closely related problems: learning (1) relative physical knowledge of object pairs and (2) physical implications of actions when applied to those object pairs. Empirical results demonstrate that it is possible to extract knowledge of actions and objects from language and that joint inference over different types of knowledge improves performance.

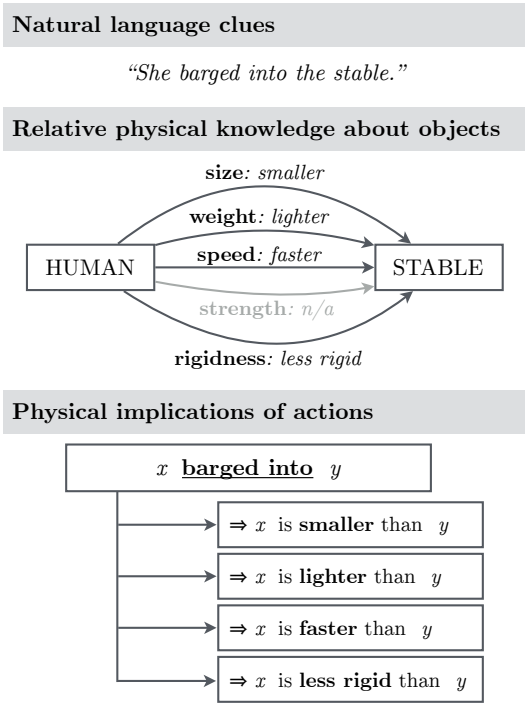


Figure 1: An overview of our approach. A verb's usage in language (top) implies physical relations between objects it takes as arguments. This allows us to reason about properties of specific objects (middle), as well as the knowledge implied by the verb itself (bottom).

1 Introduction

Reading and reasoning about natural language text often requires trivial knowledge about everyday physical actions and objects. For example, given a sentence *"Shanice could fit the trophy into the suitcase,"* we can trivially infer that the trophy must be smaller than the suitcase even though it is not stated explicitly. This reasoning requires knowledge about the action *"fit"*—in particular, typical preconditions that need to be satisfied in order to perform the action. In addition, reasoning about the applicability of various physical actions in a given situation often requires background knowledge about objects in the world, for example, that people are usually *smaller* than houses, that cars generally move *faster* than humans walk, or that a brick probably is *heavier* than a feather.

In fact, the potential use of such knowledge about everyday actions and objects can go beyond language understanding and reasoning. Many open challenges in computer vision and robotics may also benefit from such knowledge, as shown

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 266–276
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1025

in recent work that requires visual reasoning and entailment (Izadinia et al., 2015; Zhu et al., 2014). Ideally, an AI system should acquire such knowledge through direct physical interactions with the world. However, such a physically interactive system does not seem feasible in the foreseeable future.

In this paper, we present an approach to acquire trivial physical knowledge from unstructured natural language text as an alternative knowledge source. In particular, we focus on acquiring relative physical knowledge of actions and objects organized along five dimensions: size, weight, strength, rigidness, and speed. Figure 1 illustrates example knowledge of (1) relative physical relations of object pairs and (2) physical implications of actions when applied to those object pairs.

While natural language text is a rich source to obtain broad knowledge about the world, compiling trivial commonsense knowledge from unstructured text is a nontrivial feat. The central challenge lies in *reporting bias*: people rarely states the obvious (Gordon and Van Durme, 2013; Sorower et al., 2011; Misra et al., 2016; Zhang et al., 2017), since it goes against Grice's conversational maxim on the quantity of information (Grice, 1975).

In this work, we demonstrate that it is possible to overcome reporting bias and still extract the unspoken knowledge from language. The key insight is this: there is consistency in the way people describe how they interact with the world, which provides vital clues to reverse engineer the common knowledge shared among people. More concretely, we frame knowledge acquisition as joint inference over two closely related puzzles: inferring relative physical knowledge about object pairs while simultaneously reasoning about physical implications of actions.

Importantly, four of five dimensions of knowledge in our study—weight, strength, rigidness, and speed—are either not visual or not easily recognizable by image recognition using currently available computer vision techniques. Thus, our work provides unique value to complement recent attempts to acquire commonsense knowledge from web images (Izadinia et al., 2015; Bagherinezhad et al., 2016; Sadeghi et al., 2015).

In sum, our contributions are threefold:

- We introduce a new task in the domain of commonsense knowledge extraction from language, focusing on the physical implications of actions and the relative physical relations among objects, organized along five dimensions.
- We propose a model that can infer relations over grounded object pairs together with first order relations implied by physical verbs.
- We develop a new dataset VERBPHYSICS that compiles crowdsourced knowledge of actions and objects.[1]

The rest of the paper is organized as follows. We first provide the formal definition of knowledge we aim to learn in Section 2. We then describe our data collection in Section 3 and present our inference model in Section 4. Empirical results are given in Section 5 and discussed in Section 6. We review related work in Section 7 and conclude in Section 8.

2 Representation of Relative Physical Knowledge

2.1 Knowledge Dimensions

We consider five dimensions of relative physical knowledge in this work: *size, weight, strength, rigidness,* and *speed.* "Strength" in our work refers to the physical durability of an object (e.g., "diamond" is stronger than "glass"), while "rigidness" refers to the physical flexibility of an object (e.g., "glass" is more rigid than a "wire"). When considered in verb implications, *size, weight, strength,* and *rigidness* concern individual-level semantics; the relative properties implied by verbs in these dimensions are true in general. On the other hand, *speed* concerns stage-level semantics; its implied relations hold only during a window surrounding the verb.[2]

2.2 Relative physical knowledge

Let us first consider the problem of representing relative physical knowledge between two objects. We can write a single piece of knowledge like "A person is larger than a basketball" as

$$\texttt{person} >^{\text{size}} \texttt{basketball}$$

Any propositional statement can have exceptions and counterexamples. Moreover, we need to cope

[1] https://uwnlp.github.io/verbphysics/

[2] We thank reviewer two for pointing us to this terminology and for the illustrative example: "When a person throws a ball, the ball is faster than the person (stage-level) but it's not true in general that balls are faster than people (individual-level)."

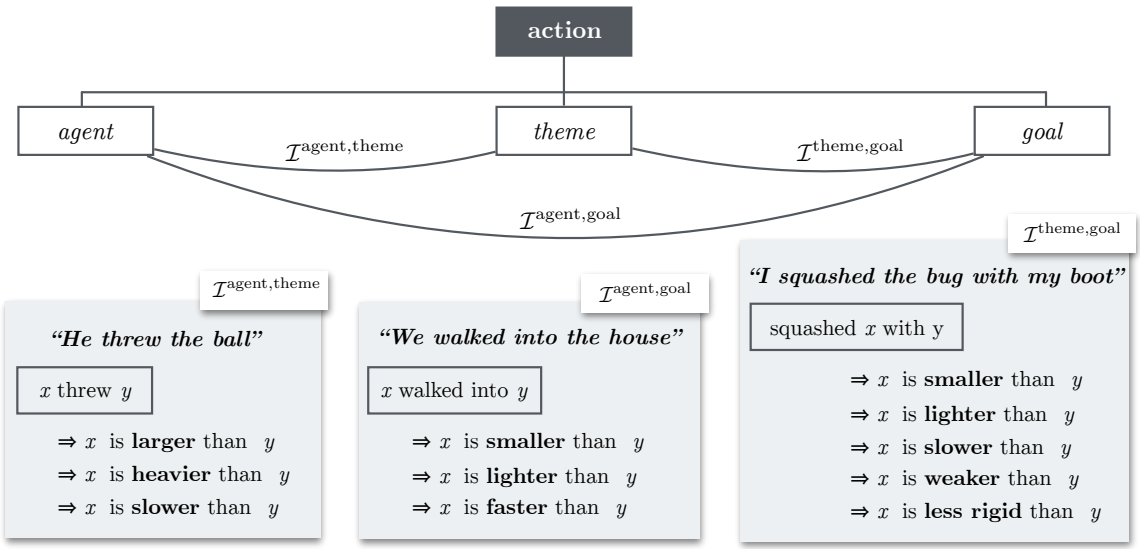


Figure 2: Example physical implications represented as frame relations between a pair of arguments.

with uncertainties involved in knowledge acquisition. Therefore, we assume each piece of knowledge is associated with a probability distribution. More formally, given objects x and y, we define a random variable $O^a_{x,y}$ whose range is $\{\boxed{>}, \boxed{<}, \boxed{\simeq}\}$ with respect to a knowledge dimension $a \in \{\text{SIZE}, \text{WEIGHT}, \text{STRENGTH}, \text{RIGIDNESS}, \text{SPEED}\}$ so that:

$$\mathbb{P}(O^a_{x,y} = r), r \in \{\boxed{>}, \boxed{<}, \boxed{\simeq}\}.$$

This immediately provides two simple properties:

$$\mathbb{P}(O_{x,y} = \boxed{>}) = \mathbb{P}(O_{y,x} = \boxed{<})$$
$$\mathbb{P}(O_{x,x} = \boxed{\simeq}) = 1$$

2.3 Physical Implications of Verbs

Next we consider representing relative physical implications of actions applied over two objects. For example, consider an action frame "x threw y." In general, following implications are likely to be true:

$$\text{“}x \text{ threw } y\text{”} \implies x >^{\text{size}} y$$
$$\text{“}x \text{ threw } y\text{”} \implies x >^{\text{weight}} y$$
$$\text{“}x \text{ threw } y\text{”} \implies x <^{\text{speed}} y$$

Again, in order to cope with exceptions and uncertainties, we assume a probability distribution associated with each implication. More formally, we define a random variable F^a_v to denote the implication of the action verb v when applied over its arguments x and y with respect to a knowledge dimension a so that:

$$\mathbb{P}(F^{\text{size}}_{\text{threw}} = \boxed{>}) := \mathbb{P}(\text{“}x \text{ threw } y\text{”} \Rightarrow x >^{\text{size}} y)$$
$$\mathbb{P}(F^{\text{wgt}}_{\text{threw}} = \boxed{>}) := \mathbb{P}(\text{“}x \text{ threw } y\text{”} \Rightarrow x >^{\text{wgt}} y)$$

where the range of F^{size}_{threw} is $\{\boxed{>}, \boxed{<}, \boxed{\simeq}\}$. Intuitively, F^{size}_{threw} represents the likely first order relation implied by "throw" over ungrounded (i.e., variable) object pairs.

The above definition assumes that there is only a single implication relation for any given verb with respect to a specific knowledge dimension. This is generally not true, since a verb, especially a common action verb, can often invoke a number of different frames according to frame semantics (Fillmore, 1976). Thus, given a number of different frame relations $v_1 ... v_T$ associated with a verb v, we define random variables F with respect to a specific frame relation v_t, i.e., $F^a_{v_t}$. We use this notation going forward.

Frame Perspective on Verb Implications: Figure 2 illustrates the frame-centric view of physical implication knowledge we aim to learn. Importantly, the key insight of our work is inspired by Fillmore's original manuscript on frame semantics (Fillmore, 1976). Fillmore has argued that "frames"—the contexts in which utterances are situated—should be considered as a third primitive of describing a language, along with a grammar and lexicon. While existing frame annotations such as FrameNet (Baker et al., 1998), PropBank (Palmer et al., 2005), and VerbNet (Kipper et al., 2000) provide rich frame knowledge associated

with a predicate, none of them provide the exact kind of physical implications we consider in our paper. Thus, our work can potentially contribute to these resources by investigating new approaches to automatically recover richer frame knowledge from language. In addition, our work is motivated by the formal semantics of Dowty (1991), as the task of learning verb implications is essentially that of extracting lexical entailments for verbs.

3 Data and Crowdsourced Knowledge

Action Verbs: We pick 50 classes of Levin verbs from both "alternation classes" and "verb classes" (Levin, 1993), which corresponds to about 1100 unique verbs. We sort this list by frequency of occurrence in our frame patterns in the Google Syntax Ngrams corpus (Goldberg and Orwant, 2013) and pick the top 100 verbs.

Action Frames: Figure 2 illustrates examples of action frame relations. Because we consider implications over pairwise argument relations for each frame, there are sometimes multiple frame relations we consider for a single frame. To enumerate action frame relations for each verb, we use syntactic patterns based on dependency parse by extracting the core components (subject, verb, direct object, prepositional object) of an action, then map the subject to an agent, the direct object to a theme, and the prepositional object to a goal.[3] For those frames that involve an argument in a prepositional phrase, we create a separate frame for each preposition based on the statistics observed in the Google Syntax Ngram corpus.

Because the syntax ngram corpus provides only tree snippets without context, this way of enumerating potential frame patterns tend to overgenerate. Thus we refine our prepositions for each frame by taking either the intersection or union with the top 5 Google Surface Ngrams (Michel et al., 2011), depending on whether the frame was under- or over-generating. We also add an additional crowdsourcing step where we ask crowd workers to judge whether a frame pattern with a particular verb and preposition could plausibly be found in a sentence. This process results in 813 frame templates, an average of 8.13 per verb.

[3]Future research could use an SRL parser instead. We use dependency parse to benefit from the Google Syntax Ngram dataset that provides language statistics over an extremely large corpus, which does not exist for SRL.

Data collected		
	Total	**Seed / dev / test**
$\text{Verbs}_{5\%}$	100	5 / 45 / 50
$\text{Verbs}_{20\%}$	"	20 / 30 / 50
$\text{Frames}_{5\%}$	813	65 / 333 / 415
$\text{Frames}_{20\%}$	"	188 / 210 / 415
$\text{Object pairs}_{5\%}$	3656	183 / 1645 / 1828
$\text{Object pairs}_{20\%}$	"	733 / 1096 / 1828

Per attribute frame statistics				
	Agreement		*Counts (usable)*	
	2/3	**3/3**	**Verbs**	**Frames**
size	0.91	0.41	96	615
weight	0.90	0.33	97	562
strength	0.88	0.25	95	465
rigidness	0.87	0.26	89	432
speed	0.93	0.36	88	420

Per attribute object pair statistics				
	Agreement		*Counts (usable)*	
	2/3	**3/3**	**Distinct objs**	**Pairs**
size	0.95	0.59	210	2552
weight	0.95	0.56	212	2586
strength	0.92	0.43	208	2335
rigidness	0.91	0.39	212	2355
speed	0.90	0.38	209	2184

Table 1: Statistics of crowdsourced knowledge. Frames are partitioned by verb. Counts are shown for *usable* data, which includes only $\geq$ 2/3 agreement and removes all with "no relation." Each prediction task (frames or object pairs) is given 5% of that domain's data as seed. We compare models using either 5% or 20% of the *other* domain's data as seed.

Object Pairs: To provide a source of ground truth relations between objects, we select the object pairs that occur in the 813 frame templates with positive pointwise mutual information (PMI) across the Google Syntax Ngram corpus. After replacing a small set of "human" nouns with a generic HUMAN object, filtering out nouns labeled as abstract by WordNet (Miller, 1995), and distilling all surface forms to their lemmas (also with WordNet), the result is 3656 object pairs.

3.1 Crowdsourcing Knowledge

We collect human judgements of the frame knowledge implications to use as a small set of seed knowledge (5%), a development set (45%), and a test set (50%). Crowd workers are given with a frame template such as "**x** threw **y**," and then asked to list a few plausible objects (including people and animals) for the missing slots (e.g., **x** and **y**).[4]

[4]This step is to prime them for thinking about the particular template; we do not use the objects they provided.

We then ask them to rate the general relationship that the arguments of the frame exhibit with respect to all knowledge dimensions (size, weight, etc.). For each knowledge dimension, or attribute, a, workers select an answer from (1) $\mathbf{x} >^a \mathbf{y}$, (2) $\mathbf{x} <^a \mathbf{y}$, (3) $\mathbf{x} \simeq^a \mathbf{y}$, or (4) no general relation.

We conduct a similar crowdsourcing step for the set of object pairs. We ask crowd workers to compare each of the 3656 object pairs along the five knowledge dimensions we consider, selecting an answer from the same options above as with frames. We reserve 50% of the data as a test set, and split the remainder up either 5% / 45% or 20% / 30% (seed / development) to investigate the effects of different seed knowledge sizes on the model.

Statistics for the dataset are provided in Table 1. About 90% of the frames as well as object pairs had 2/3 agreement between workers. After removing frame/attribute combinations and object pairs that received less than 2/3 agreement, or were selected by at least 2/3 workers to have no relation, we end up with roughly 400–600 usable frames and 2100–2500 usable object pairs per attribute.

4 Model

We model knowledge acquisition as probabilistic inference over a factor graph of knowledge. As shown in Figure 3, the graph consists of multiple substrates (page-wide boxes) corresponding to different knowledge dimensions (shown only three of them —strength, size, weight—for brevity). Each substrate consists of two types of sub-graphs: verb subgraphs and object subgraphs, which are connected through factors that quantify action–object compatibilities. Connecting across substrates are factors that model inter-dependencies across different knowledge dimensions. In what follows, we describe each graph component.

4.1 Nodes

The factor graph contains two types of nodes in order to capture two classes of knowledge. The first type of nodes are object pair nodes. Each object pair node is a random variable $O^a_{x,y}$ which captures the relative strength of an attribute a between objects x and y.

The second type of nodes are frame nodes. Each frame node is a random variable $F^a_{v_t}$. This corresponds to the verb v used in a particular type of frame t, and captures the implied knowledge the frame v_t holds along an attribute a.

All random variables take on the values $\{\boxed{>}, \boxed{<}, \boxed{\simeq}\}$. For an object pair node $O^a_{x,y}$, the value represents the belief about the relation between x and y along the attribute a. For a frame node $F^a_{v_t}$, the value represents the belief about the relation along the attribute a between *any* two objects that might be used in the frame v_t.

We denote the sets of all object pair and frame random variables $\mathcal{O}$ and $\mathcal{F}$, respectively.

4.2 Action–Object Compatibility

The key aspect of our work is to reason about two types of knowledge simultaneously: relative knowledge of grounded object pairs, and implications of actions related to those objects. Thus we connect the verb subgraphs and object subgraphs through selectional preference factors ψ_s between two such nodes $O^a_{x,y}$ and $F^a_{v_t}$ if we find evidence from text that suggests objects x and y are used in the frame v_t. These factors encourage both random variables to agree on the same value.

As an example, consider a node $O^{size}_{p,b}$ which represents the relative size of a person and a basketball, and a node $F^{size}_{threw_{dobj}}$ which represents the relative size implied by an *"x threw y"* frame. If we find significant evidence in text that *"[person] threw [basketball]"* occurs, we would add a selectional preference factor to connect $O^{size}_{p,b}$ with $F^{size}_{threw_{dobj}}$ and encourage them towards the same value. This means that if it is discovered that people are larger than basketballs (the value $\boxed{>}$), then we would expect the frame *"x threw y"* to entail $x >^{size} y$ (also the value $\boxed{>}$).

4.3 Semantic Similarities

Some frames have relatively sparse text evidences to support their corresponding knowledge acquisition. Thus, we include several types of factors based on semantic similarities as described below.

Cross-Verb Frame Similarity: We add a group of factors ψ_v between two verbs v and u (to connect a specific frame of v with a corresponding frame of u) based on the verb-level similarities.

Within-Verb Frame Similarity: Within each verb v, which consists of a set of frame relations $v_1, ...v_T$, we also include frame-level similarity factors ψ_f between v_i and v_j. This gives us more evidence over a broader range of frames when textual evidence might be sparse.

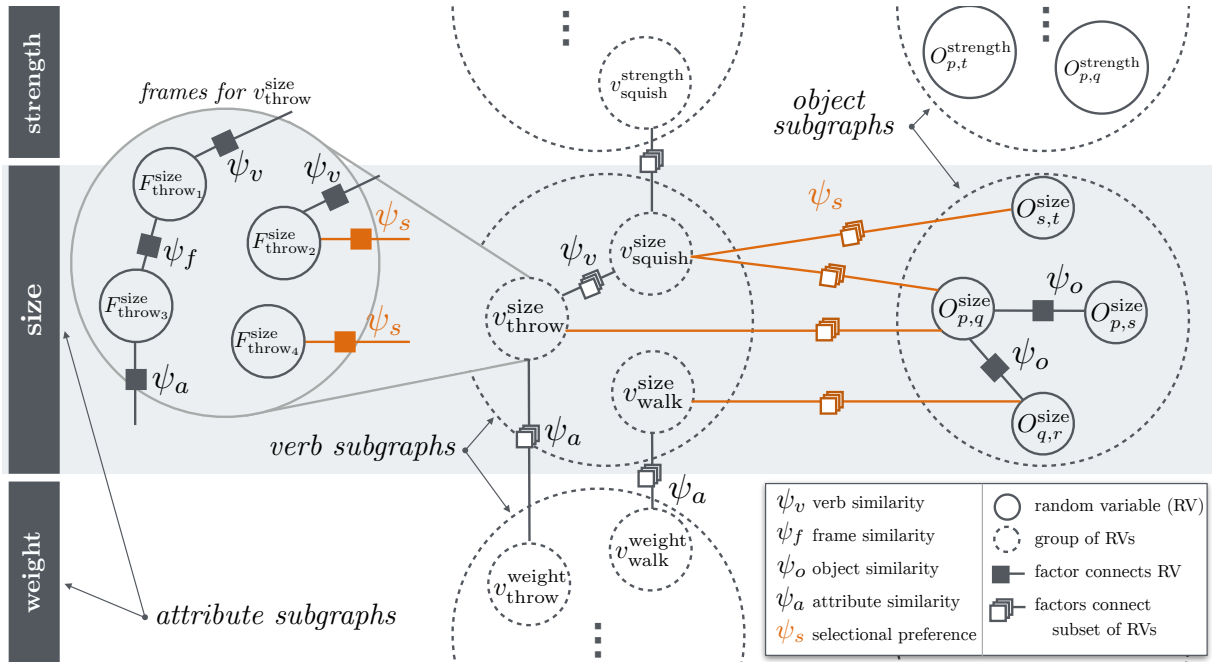


Figure 3: High level view of the factor graph model. Performance on both learning relative knowledge about objects (right), as well as entailed knowledge from verbs (center) via realized frames (left), is improved by modeling their interplay (orange). Unary seed (ψ_{seed}) and embedding (ψ_{emb}) factors are omitted for clarity.

Object Similarity: As with verbs, we add factors ψ_o that encourage similar pairs of objects to take the same value. Given that each node represents a pair of objects, finding that x and y are similar yields two main cases in how to add factors (aside from the trivial case where the variable $O^a_{x,y}$ is given a unary factor to encourage the value $\boxed{\simeq}$).

1. If nodes $O_{x,z}$ and $O_{y,z}$ exist, we expect objects x and y to both have a similar relation to z. We add a factor that encourages $O_{x,z}$ and $O_{y,z}$ to take the same value. The same is true if nodes $O_{z,x}$ and $O_{z,y}$ exist.
2. On the other hand, if nodes $O_{x,z}$ and $O_{z,y}$ exist, we expect these two nodes to reach the opposite decision. In this case, we add a factor that encourages one node to take the value $\boxed{>}$ if the other prefers the value $\boxed{<}$, and vice versa. (For the case of $\boxed{\simeq}$, if one prefers that value, then both should.)

4.4 Cross-Knowledge Correlation

Some knowledge dimensions, such as size and weight, have a significant correlation in their implied relations. For two such attributes a and b, if the same frame $F^a_{v_i}$ and $F^b_{v_i}$ exists in both graphs, we add a factor ψ_a between them to push them towards taking the same value.

4.5 Seed Knowledge

In order to kick off learning, we provide a small set of seed knowledge among the random variables in $\{\mathcal{O}, \mathcal{F}\}$ with seed factors ψ_{seed}. These unary seed factors push the belief for its associated random variable strongly towards the seed label.

4.6 Potential Functions

Unary Factors: For all frame and object pair random variables in the training set, we train a maximum entropy classifier to predict the value of the variable. We then use the probabilities of the classifier as potentials for seed factors given to all random variables in their class (frame or object pair). Each log-linear classifier is trained separately per attribute on a featurized vector of the variable:

$$\mathbb{P}(r|X^a) \propto e^{w_a \cdot f(X^a)}$$

The feature function is defined differently according to the node type:

$$f(O^a_{p,q}) := \langle g(p), g(q) \rangle$$
$$f(F^a_{v_t}) := \langle h(t), g(v), g(t) \rangle$$

Algorithm	Development						Test					
	size	weight	stren	rigid	speed	*overall*	size	weight	stren	rigid	speed	*overall*
RANDOM	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33
MAJORITY	0.38	0.41	0.42	0.18	**0.83**	0.43	0.35	0.35	0.43	0.20	0.88	0.44
EMB-MAXENT	0.62	0.64	0.60	**0.83**	**0.83**	0.69	0.55	0.55	0.59	0.79	0.88	0.66
OUR MODEL (A)	0.71	0.63	0.61	0.82	**0.83**	0.71	0.55	0.55	0.55	0.79	**0.89**	0.65
OUR MODEL (B)	**0.75**	**0.68**	**0.68**	0.82	0.78	**0.74**	**0.74**	**0.71**	**0.65**	**0.80**	0.87	**0.75**

Algorithm	Development						Test					
	size	weight	stren	rigid	speed	*overall*	size	weight	stren	rigid	speed	*overall*
RANDOM	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33
MAJORITY	0.50	0.54	0.51	0.50	0.53	0.51	0.51	0.55	0.52	0.49	0.50	0.51
EMB-MAXENT	0.68	0.66	0.64	0.67	0.65	0.66	0.71	0.67	0.64	0.65	**0.63**	0.66
OUR MODEL (A)	0.74	0.69	0.67	**0.68**	**0.66**	0.69	0.68	0.70	0.66	**0.66**	0.60	0.66
OUR MODEL (B)	**0.75**	**0.74**	**0.71**	**0.68**	**0.66**	**0.71**	**0.75**	**0.76**	**0.72**	0.65	0.61	**0.70**

Table 2: Accuracy of baselines and our model on both tasks. Top: frame prediction task; bottom: object pair prediction task. In both tasks 5% of in-domain data (frames or object pairs, respectively) are available as seed data. We compare providing the other type of data (object pairs or frames, respectively) as seed knowledge, trying 5% (OUR MODEL (A)) and 20% (OUR MODEL (B)).

Here $g(x)$ is the GloVe word embedding (Pennington et al., 2014) for the word x (t is the frame relation's preposition, and $g(t)$ is simply set to the zero vector if there is no preposition) and $h(t)$ is a one-hot vector of the frame relation type. We use GloVe vectors of 100 dimensions for verbs and 50 dimensions for objects and prepositions (the dimensions picked based on development set).

Binary Factors: In the case of all other factors, we use a "soft 1" agreement matrix with strong signal down the diagonals:

$$\begin{bmatrix} & > & \simeq & < \\ > & \mathbf{0.7} & 0.1 & 0.2 \\ \simeq & 0.15 & \mathbf{0.7} & 0.15 \\ < & 0.2 & 0.1 & \mathbf{0.7} \end{bmatrix}$$

4.7 Inference

After our full graph is constructed, we use belief propagation to infer the assignments of frames and object pairs not in our training data. Each message μ is a vector where each element is the probability that a random variable takes on each value $x \in \{\boxed{>}, \boxed{<}, \boxed{\simeq}\}$. A message passed from a random variable v to a neighboring factor f about the value x is the product of the messages from its other neighboring factors about x:

$$\mu_{v \to f}(x) \propto \prod_{f' \in N(v) \setminus \{f\}} \mu_{f' \to v}(x)$$

A message passed from a factor f with potential ψ to a random variable v about its value x is a marginalized belief about v taking value x from the other neighboring random variables combined with its potential:

$$\mu_{f \to v}(x) \propto \sum_{\mathbf{x}: \mathbf{x}[v]=x} \psi(\mathbf{x}) \prod_{v' \in N(f) \setminus \{v\}} \mu_{v' \to f}(\mathbf{x}[v'])$$

After stopping belief propagation, the marginals for a node can be computed and used as a decision for that random variable. The marginal for v taking value x is the product of its surrounding factors' messages:

$$v(x) \propto \prod_{f \in N(v)} \mu_{f \to v}(x)$$

5 Experimental Results

Factor Graph Construction: We first need to pick a set of frames and objects to determine our set of random variables. The frames are simply the subset of the frames that were crowdsourced in the given configuration (e.g., seed + dev), with "soft 1" unary seed factors (the gold label indexed row of the binary factor matrix) given only to those in the seed set. The same selection criteria and seed factors are applied to the crowdsourced object pairs.

For lexical similarity factors (ψ_v, ψ_o), we pick connections based on the cosine similarity scores of GloVe vectors thresholded above a value chosen based on development set performance. Attribute similarity factors (ψ_a) are chosen based on sets of frames that reach largely the same decisions on the seed data (95%). Frame similarity factors (ψ_f) are added to pairs of frames with linguistically similar constructions. Finally, selectional preference

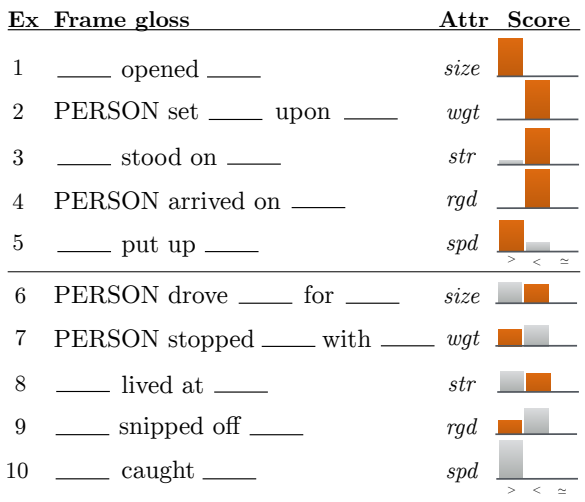


Figure 4: Example model predictions on dev set frames. The model's confidence is shown by the bars on the right. The correct relation is highlighted in orange (6–10 are failure cases for the model). If there are two blanks, the relation is between them. If there is only one blank, the relation is between PERSON and the blank. Note that $\boxed{\simeq}$ receives miniscule weight because it is never the correct value for frames in the seed set.

factors (ψ_s) are picked by using a threshold (also tuned on the development set) of pointwise mutual information (PMI) between the frames and the object pairs' occurrences in the Google Syntax Ngram corpus.

For each task, we consider the set of factors to include in each model a hyperparameter, which is also tuned on the development set.

Baselines: Baselines include making a RANDOM choice, picking between $\boxed{>}$, $\boxed{<}$, and $\boxed{\simeq}$), picking the MAJORITY label, and a maximum entropy classifier based on the embedding representations (EMB-MAXENT) defined in Section 4.6.

Inferring Knowledge of Actions: Our first experiment is to predict knowledge implied by new frames. In this task, 5% of the frames are available as seed knowledge. We experiment with two different sets of seed knowledge for the object pair data: OUR MODEL (A) uses only 5% of the object pair data as seed, and OUR MODEL (B) uses 20%.

The full results for the baseline methods and our model are given in the upper half of Table 2. Our model outperforms the baselines on all attributes except for the speed, which has a highly skewed label distribution to allow the majority baseline to perform well. Ablations are given in Table 3, and sample correct predictions from the development set are shown in examples 1–5 of Figure 4.

Ablated (or added) component	Accuracy
– Verb similarity	0.69
+ Frame similarity	0.62
– Action-object compatibility	0.62
– Object similarity	0.70
+ Attribute similarity	0.62
– Frame embeddings	0.63
– Frame seeds	0.62
– Object embeddings	0.62
– Object seeds	0.62
OUR MODEL (A)	**0.71**

Table 3: Ablation results on *size* attribute for the frame prediction task on the development dataset for OUR MODEL (A) (5% of the object pairs as seed data). We find that different graph configurations improve performance for different tasks and data amounts. In this setting, frame and attribute similarity factors hindered performance.

Inferring Knowledge of Objects: Our second experiment is to predict the correct relations of new object pairs. The data for this task is the inverse of before: 5% of the object pairs are available as seed knowledge, and we experiment with both 5% (OUR MODEL (A)) and 20% (OUR MODEL (B)) frames given as seed data. Again, both are independently tuned on the development data. Results for this task are presented in the lower half of Table 2. While OUR MODEL (A) is competitive with the strongest baseline, introducing the additional frame data allows OUR MODEL (B) to reach the highest accuracy.

6 Discussion

Metaphorical Language: While our frame patterns are intended to capture action verbs, our templates also match senses of those verbs that can be used with abstract or metaphorical arguments, rather than directly physical ones. One example from the development set is "x contained y." While x and y can be real objects, more abstract senses of "contained" could involve y as a "forest fire" or even a "revolution." In these instances, $x >^{\text{size}} y$ is plausible as an abstract notion: if some entity can contain a revolution, we might think that entity as "larger" or "stronger" than the revolution.

Error analysis: Examples 6–10 in Figure 4 highlight failure cases for the model. Example

6 shows a case where the comparison is nonsensical because *"for"* would naturally be followed by a purpose (*"He drove the car for work."*) or a duration (*"She drove the car for hours."*) rather than a concrete object whose size is measurable. Example 7 highlights an underspecified frame. One crowd worker provided the example, "PERSON *stopped the fly with {the jar / a swatter},"* where $\text{fly} <^{\text{weight}} \{\text{jar}, \text{swatter}\}$. However, two crowd workers provided examples like "PERSON *stopped their car with the brake,"* where clearly $\text{car} >^{\text{weight}} \text{brake}$. This example illustrates complex underlying physics we do not model: a brake—the pedal itself—is used to stop a car, but it does so by applying significant force through a separate system.

The next two examples are cases where the model nearly predicts correctly (8, e.g., *"She lived at the office."*) and is just clearly wrong (9, e.g., *"He snipped off a locket of hair"*). Example 10 demonstrates a case of polysemy where the model picks the wrong side. In the phrase, *"She caught the runner in first,"*, it is correct that $\text{she} >^{\text{speed}} \text{runner}$. However, the sense chosen by the crowd workers is that of, *"She caught the baseball,"* where indeed $\text{she} <^{\text{speed}} \text{baseball}$.

7 Related work

Several works straddle the gap between IE, knowledge base completion, and learning commonsense knowledge from text. Earlier works in these areas use large amounts of text to try to extract general statements like "A THING CAN BE READABLE" (Gordon et al., 2010) and frequencies of events (Gordon and Schubert, 2012). Our work focuses on specific domains of knowledge rather than general statements or occurrence statistics, and develops a frame-centric approach to circumvent reporting bias. Other work uses a knowledge base and scores unseen tuples based on similarity to existing ones (Angeli and Manning, 2013; Li et al., 2016). Relatedly, previous work uses natural language inference to infer new facts from a dataset of commonsense facts that can be extracted from unstructured text (Angeli and Manning, 2014). In contrast, we focus on a small number of specific types of knowledge without access to an existing database of knowledge.

A number of recent works combine multimodal input to learn visual attributes (Bruni et al., 2012; Silberer et al., 2013), extract commonsense knowledge from web images (Tandon et al., 2016), and overcome reporting bias (Misra et al., 2016). In contrast, we focus on natural language evidence to reason about attributes that are both in (size) and out (weight, rigidness, etc.) of the scope of computer vision. Yet other works mine numerical attributes of objects (Narisawa et al., 2013; Takamura and Tsujii, 2015; Davidov and Rappoport, 2010) and comparative knowledge from the web (Tandon et al., 2014). Our work uniquely learns verb-centric lexical entailment knowledge.

A handful of works have attempted to learn the types of knowledge we address in this work. One recent work tried to directly predict several binary attributes (such "is large" and "is yellow") from on off-the-shelf word embeddings, noting that accuracy was very low (Rubinstein et al., 2015). Another line of work addressed grounding verbs in the context of robotic tasks. One paper in this line acquires verb meanings by observing state changes in the environment (She and Chai, 2016). Another work in this line does a deep investigation of eleven verbs, modeling their physical effect via annotated images along eighteen attributes (Gao et al., 2016). These works are encouraging investigations into multimodal groundings of a small set of verbs. Our work instead grounds into a fixed set of attributes but leverages language on a broader scale to learn about more verbs in more diverse set of frames.

8 Conclusion

We presented a novel take on verb-centric frame semantics to learn implied physical knowledge latent in verbs. Empirical results confirm that by modeling changes in physical attributes entailed by verbs together with objects that exhibit these properties, we are able to better infer new knowledge in both domains.

Acknowledgements

This research is supported in part by the National Science Foundation Graduate Research Fellowship, DARPA CwC program through ARO (W911NF-15-1-0543), the NSF grant (IIS-1524371), and gifts by Google and Facebook. The authors thank the anonymous reviewers for their thorough and insightful comments.

References

Gabor Angeli and Christopher D Manning. 2013. Philosophers are mortal: Inferring the truth of unseen facts. In *CoNLL*. pages 133–142.

Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *EMNLP*. pages 534–545.

Hessam Bagherinezhad, Hannaneh Hajishirzi, Yejin Choi, and Ali Farhadi. 2016. Are elephants bigger than butterflies? reasoning about sizes of objects. *arXiv preprint arXiv:1602.00753* .

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 86–90.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 136–145.

Dmitry Davidov and Ari Rappoport. 2010. Extraction and approximation of numerical attributes from the web. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1308–1317.

David Dowty. 1991. Thematic proto-roles and argument selection. *language* pages 547–619.

Charles J Fillmore. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences* 280(1):20–32.

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Y Chai. 2016. Physical causality of action verbs in grounded language understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. volume 1, pages 1814–1824.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*. volume 1, pages 241–247.

Jonathan Gordon and Lenhart K Schubert. 2012. Using textual patterns to learn expected event frequencies. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, pages 122–127.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 workshop on Automated knowledge base construction*. ACM, pages 25–30.

Jonathan Gordon, Benjamin Van Durme, and Lenhart K Schubert. 2010. Learning from the web: Extracting general world knowledge from noisy text. In *Collaboratively-Built Knowledge Sources and AI*.

HP Grice. 1975. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics*. Academic Press, New York, volume 3: Speech Acts.

Hamid Izadinia, Fereshteh Sadeghi, Santosh K Divvala, Hannaneh Hajishirzi, Yejin Choi, and Ali Farhadi. 2015. Segment-phrase table for semantic segmentation, visual entailment and paraphrasing. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 10–18.

Karin Kipper, Hoa Trang Dang, Martha Palmer, et al. 2000. Class-based construction of a verb lexicon. *AAAI/IAAI* 691:696.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Berlin, Germany, August. Association for Computational Linguistics*.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. 2011. Quantitative analysis of culture using millions of digitized books. *science* 331(6014):176–182.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Ishan Misra, C Lawrence Zitnick, Margaret Mitchell, and Ross Girshick. 2016. Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 2930–2939.

Katsuma Narisawa, Yotaro Watanabe, Junta Mizuno, Naoaki Okazaki, and Kentaro Inui. 2013. Is a 204 cm man tall or small? acquisition of numerical common sense from the web. In *ACL (1)*. pages 382–391.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31(1):71–106.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *ACL (2)*. pages 726–730.

Fereshteh Sadeghi, Santosh K Kumar Divvala, and Ali Farhadi. 2015. Viske: Visual knowledge extraction and question answering by visual verification of relation phrases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 1456–1464.

Lanbo She and Joyce Y Chai. 2016. Incremental acquisition of verb hypothesis space towards physical world interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2013. Models of semantic representation with visual attributes. In *ACL (1)*. pages 572–582.

Mohammad S Sorower, Janardhan R Doppa, Walker Orr, Prasad Tadepalli, Thomas G Dietterich, and Xiaoli Z Fern. 2011. Inverting grice's maxims to learn rules from natural language extractions. In *Advances in neural information processing systems*. pages 1053–1061.

Hiroya Takamura and Jun'ichi Tsujii. 2015. Estimating numerical attributes by bringing together fragmentary clues. In *HLT-NAACL*. pages 1305–1310.

Niket Tandon, Gerard De Melo, and Gerhard Weikum. 2014. Acquiring comparative commonsense knowledge from the web. In *AAAI*. pages 166–172.

Niket Tandon, Charles Hariman, Jacopo Urbani, Anna Rohrbach, Marcus Rohrbach, and Gerhard Weikum. 2016. Commonsense in parts: Mining part-whole relations from the web and image tags. In *AAAI*. pages 243–250.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2017. Ordinal common-sense inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yuke Zhu, Alireza Fathi, and Li Fei-Fei. 2014. Reasoning about object affordances in a knowledge base representation. In *European conference on computer vision*. Springer, pages 408–424.

A* CCG Parsing with a Supertag and Dependency Factored Model

Masashi Yoshikawa and **Hiroshi Noji** and **Yuji Matsumoto**
Graduate School of Information and Science
Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara, 630-0192, Japan
{ masashi.yoshikawa.yh8, noji, matsu }@is.naist.jp

Abstract

We propose a new A* CCG parsing model in which the probability of a tree is decomposed into factors of CCG categories and its syntactic dependencies both defined on bi-directional LSTMs. Our factored model allows the precomputation of all probabilities and runs very efficiently, while modeling sentence structures explicitly via dependencies. Our model achieves the state-of-the-art results on English and Japanese CCG parsing.[1]

1 Introduction

Supertagging in lexicalized grammar parsing is known as *almost parsing* (Bangalore and Joshi, 1999), in that each supertag is syntactically informative and most ambiguities are resolved once a correct supertag is assigned to every word. Recently this property is effectively exploited in A* Combinatory Categorial Grammar (CCG; Steedman (2000)) parsing (Lewis and Steedman, 2014; Lewis et al., 2016), in which the probability of a CCG tree $\boldsymbol{y}$ on a sentence $\boldsymbol{x}$ of length N is the product of the probabilities of supertags (categories) c_i (locally factored model):

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i \in [1,N]} P_{tag}(c_i|\boldsymbol{x}). \quad (1)$$

By not modeling every combinatory rule in a derivation, this formulation enables us to employ efficient A* search (see Section 2), which finds the most probable supertag sequence that can build a well-formed CCG tree.

Although much ambiguity is resolved with this supertagging, some ambiguity still remains. Figure 1 shows an example, where the two CCG parses are derived from the same supertags. Lewis et al.'s approach to this problem is resorting to some deterministic rule. For example, Lewis et al. (2016) employ the attach low heuristics, which is motivated by the right-branching tendency of English, and always prioritizes (b) for this type of ambiguity. Though for English it empirically works well, an obvious limitation is that it does not always derive the correct parse; consider a phrase "*a house in Paris with a garden*", for which the correct parse has the structure corresponding to (a) instead.

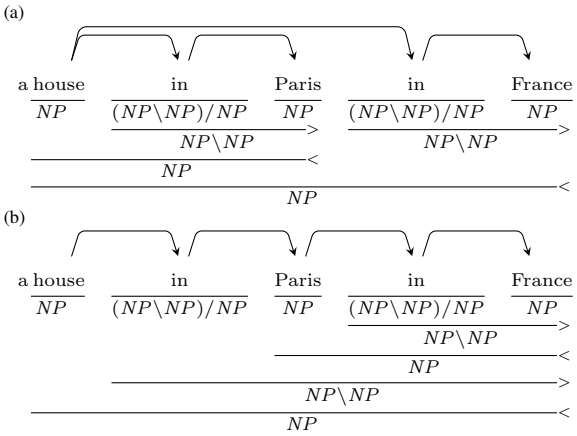


Figure 1: CCG trees that are equally likely under Eq. 1. Our model resolves this ambiguity by modeling the head of every word (dependencies).

In this paper, we provide a way to resolve these remaining ambiguities under the locally factored model, by explicitly modeling bilexical dependencies as shown in Figure 1. Our joint model is still locally factored so that an efficient A* search can be applied. The key idea is to predict the head of every word independently as in Eq. 1 with a strong unigram model, for which we utilize the scoring model in the recent successful graph-based dependency parsing on LSTMs (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016). Specif-

[1] Our software and the pretrained models are available at: https://github.com/masashi-y/depccg.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 277–287
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1026

ically, we extend the bi-directional LSTM (bi-LSTM) architecture of Lewis et al. (2016) predicting the supertag of a word to predict the head of it at the same time with a bilinear transformation.

The importance of modeling structures beyond supertags is demonstrated in the performance gain in Lee et al. (2016), which adds a recursive component to the model of Eq. 1. Unfortunately, this formulation loses the efficiency of the original one since it needs to compute a recursive neural network every time it searches for a new node. Our model does not resort to the recursive networks while modeling tree structures via dependencies.

We also extend the tri-training method of Lewis et al. (2016) to learn our model with dependencies from unlabeled data. On English CCGbank test data, our model with this technique achieves 88.8% and 94.0% in terms of labeled and unlabeled F1, which mark the best scores so far.

Besides English, we provide experiments on Japanese CCG parsing. Japanese employs freer word order dominated by the case markers and a deterministic rule such as the attach low method may not work well. We show that this is actually the case; our method outperforms the simple application of Lewis et al. (2016) in a large margin, 10.0 points in terms of clause dependency accuracy.

2 Background

Our work is built on A* CCG parsing (Section 2.1), which we extend in Section 3 with a head prediction model on bi-LSTMs (Section 2.2).

2.1 Supertag-factored A* CCG Parsing

CCG has a nice property that since every category is highly informative about attachment decisions, assigning it to every word (*supertagging*) resolves most of its syntactic structure. Lewis and Steedman (2014) utilize this characteristics of the grammar. Let a CCG tree $\boldsymbol{y}$ be a list of categories $\langle c_1, \ldots, c_N \rangle$ and a derivation on it. Their model looks for the most probable $\boldsymbol{y}$ given a sentence $\boldsymbol{x}$ of length N from the set $Y(\boldsymbol{x})$ of possible CCG trees under the model of Eq. 1:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in Y(\boldsymbol{x})} \sum_{i \in [1,N]} \log P_{tag}(c_i | \boldsymbol{x}).$$

Since this score is factored into each supertag, they call the model a *supertag-factored* model.

Exact inference of this problem is possible by A* parsing (Klein and D. Manning, 2003), which uses the following two scores on a chart:

$$b(C_{i,j}) = \sum_{c_k \in \boldsymbol{c}_{i,j}} \log P_{tag}(c_k | \boldsymbol{x}),$$

$$a(C_{i,j}) = \sum_{k \in [1,N] \setminus [i,j]} \max_{c_k} \log P_{tag}(c_k | \boldsymbol{x}),$$

where $C_{i,j}$ is a chart item called an *edge*, which abstracts parses spanning interval $[i, j]$ rooted by category C. The chart maps each edge to the derivation with the highest score, i.e., the Viterbi parse for $C_{i,j}$. $\boldsymbol{c}_{i,j}$ is the sequence of categories on such Viterbi parse, and thus b is called the Viterbi inside score, while a is the approximation (upper bound) of the Viterbi outside score.

A* parsing is a kind of CKY chart parsing augmented with an agenda, a priority queue that keeps the edges to be explored. At every step it pops the edge e with the highest priority $b(e) + a(e)$ and inserts that into the chart, and enqueue any edges that can be built by combining e with other edges in the chart. The algorithm terminates when an edge $C_{1,N}$ is popped from the agenda.

A* search for this model is quite efficient because both b and a can be obtained from the unigram category distribution on every word, which can be precomputed before search. The heuristics a gives an upper bound on the true Viterbi outside score (i.e., admissible). Along with this the condition that the inside score never increases by expansion (monotonicity) guarantees that the first found derivation on $C_{1,N}$ is always optimal. $a(C_{i,j})$ matches the true outside score if the one-best category assignments on the outside words ($\arg\max_{c_k} \log P_{tag}(c_k | \boldsymbol{x})$) can comprise a well-formed tree with $C_{i,j}$, which is generally not true.

Scoring model For modeling P_{tag}, Lewis and Steedman (2014) use a log-linear model with features from a fixed window context. Lewis et al. (2016) extend this with bi-LSTMs, which encode the complete sentence and capture the long range syntactic information. We base our model on this bi-LSTM architecture, and extend it to modeling a head word at the same time.

Attachment ambiguity In A* search, an edge with the highest priority $b + a$ is searched first, but as shown in Figure 1 the same categories (with the same priority) may sometimes derive more than

one tree. In Lewis and Steedman (2014), they prioritize the parse with longer dependencies, which they judge with a conversion rule from a CCG tree to a dependency tree (Section 4). Lewis et al. (2016) employ another heuristics prioritizing low attachments of constituencies, but inevitably these heuristics cannot be flawless in any situations. We provide a simple solution to this problem by explicitly modeling bilexical dependencies.

2.2 Bi-LSTM Dependency Parsing

For modeling dependencies, we borrow the idea from the recent graph-based neural dependency parsing (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016) in which each dependency arc is scored directly on the outputs of bi-LSTMs. Though the model is first-order, bi-LSTMs enable conditioning on the entire sentence and lead to the state-of-the-art performance. Note that this mechanism is similar to modeling of the supertag distribution discussed above, in that for each word the distribution of the head choice is unigram and can be precomputed. As we will see this keeps our joint model still locally factored and A* search tractable. For score calculation, we use an extended bilinear transformation by Dozat and Manning (2016) that models the prior headness of each token as well, which they call *biaffine*.

3 Proposed Method

3.1 A* parsing with Supertag and Dependency Factored Model

We define a CCG tree $\boldsymbol{y}$ for a sentence $\boldsymbol{x} = \langle x_i, \ldots, x_N \rangle$ as a triplet of a list of CCG categories $\boldsymbol{c} = \langle c_1, \ldots, c_N \rangle$, dependencies $\boldsymbol{h} = \langle h_1, \ldots, h_N \rangle$, and the derivation, where h_i is the head index of x_i. Our model is defined as follows:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i \in [1,N]} P_{tag}(c_i|\boldsymbol{x}) \prod_{i \in [1,N]} P_{dep}(h_i|\boldsymbol{x}). \tag{2}$$

The added term P_{dep} is a unigram distribution of the head choice.

A* search is still tractable under this model. The search problem is changed as:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in Y(\boldsymbol{x})} \left(\sum_{i \in [1,N]} \log P_{tag}(c_i|\boldsymbol{x}) + \sum_{i \in [1,N]} \log P_{dep}(h_i|\boldsymbol{x}) \right),$$

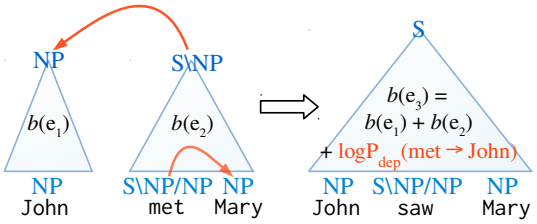


Figure 2: Viterbi inside score for edge e_3 under our model is the sum of those of e_1 and e_2 and the score of dependency arc going from the head of e_2 to that of e_1 (the head direction changes according to the child categories).

and the inside score is given by:

$$b(C_{i,j}) = \sum_{c_k \in \boldsymbol{c}_{i,j}} \log P_{tag}(c_k|\boldsymbol{x}) + \sum_{k \in [i,j] \setminus \{root(\boldsymbol{h}^C_{i,j})\}} \log P_{dep}(h_k|\boldsymbol{x}), \tag{3}$$

where $\boldsymbol{h}^C_{i,j}$ is a dependency subtree for the Viterbi parse on $C_{i,j}$ and $root(\boldsymbol{h})$ returns the root index. We exclude the head score for the subtree root token since it cannot be resolved inside $[i, j]$. This causes the mismatch between the goal inside score $b(C_{1,N})$ and the true model score (log of Eq. 2), which we adjust by adding a special unary rule that is always applied to the popped goal edge $C_{1,N}$.

We can calculate the dependency terms in Eq. 3 on the fly when expanding the chart. Let the currently popped edge be $A_{i,k}$, which will be combined with $B_{k,j}$ into $C_{i,j}$. The key observation is that only one dependency arc (between $root(\boldsymbol{h}^A_{i,k})$ and $root(\boldsymbol{h}^B_{k,j})$) is resolved at every combination (see Figure 2). For every rule $C \to A\ B$ we can define the head direction (see Section 4) and P_{dep} is obtained accordingly. For example, when the right child B becomes the head, $b(C_{i,j}) = b(A_{i,k}) + b(B_{k,j}) + \log P_{dep}(h_l = m|\boldsymbol{x})$, where $l = root(\boldsymbol{h}^A_{i,k})$ and $m = root(\boldsymbol{h}^B_{k,j})$ ($l < m$).

The Viterbi outside score is changed as:

$$a(C_{i,j}) = \sum_{k \in [1,N] \setminus [i,j]} \max_{c_k} \log P_{tag}(c_k|\boldsymbol{x}) + \sum_{k \in L} \max_{h_k} \log P_{dep}(h_k|\boldsymbol{x}),$$

where $L = [1, N] \setminus [k' | k' \in [i, j], root(\boldsymbol{h}^C_{i,j}) \neq k']$. We regard $root(\boldsymbol{h}^C_{i,j})$ as an outside word since its head is undefined yet. For every outside word we independently assign the weight of its argmax

head, which may not comprise a well-formed dependency tree. We initialize the agenda by adding an item for every supertag C and word x_i with the score $a(C_{i,i}) = \sum_{k \in I \setminus \{i\}} \max \log P_{tag}(c_k|\boldsymbol{x}) + \sum_{k \in I} \max \log P_{dep}(h_k|\boldsymbol{x})$. Note that the dependency component of it is the same for every word.

3.2 Network Architecture

Following Lewis et al. (2016) and Dozat and Manning (2016), we model P_{tag} and P_{dep} using bi-LSTMs for exploiting the entire sentence to capture the long range phenomena. See Figure 3 for the overall network architecture, where P_{tag} and P_{dep} share the common bi-LSTM hidden vectors.

First we map every word x_i to their hidden vector $\boldsymbol{r}_i$ with bi-LSTMs. The input to the LSTMs is word embeddings, which we describe in Section 6. We add special start and end tokens to each sentence with the trainable parameters following Lewis et al. (2016). For P_{dep}, we use the biaffine transformation in Dozat and Manning (2016):

$$\begin{aligned} \boldsymbol{g}_i^{dep} &= MLP_{child}^{dep}(\boldsymbol{r}_i), \\ \boldsymbol{g}_{h_i}^{dep} &= MLP_{head}^{dep}(\boldsymbol{r}_{h_i}), \\ &P_{dep}(h_i|\boldsymbol{x}) \\ &\quad \propto \exp((\boldsymbol{g}_i^{dep})^{\mathrm{T}} W_{dep} \boldsymbol{g}_{h_i}^{dep} + \boldsymbol{w}_{dep} \boldsymbol{g}_{h_i}^{dep}), \end{aligned} \tag{4}$$

where MLP is a multilayered perceptron. Though Lewis et al. (2016) simply use an MLP for mapping $\boldsymbol{r}_i$ to P_{tag}, we additionally utilize the hidden vector of the most probable head $h_i = \arg\max_{h'_i} P_{dep}(h'_i|\boldsymbol{x})$, and apply $\boldsymbol{r}_i$ and $\boldsymbol{r}_{h_i}$ to a bilinear function:[2]

$$\begin{aligned} \boldsymbol{g}_i^{tag} &= MLP_{child}^{tag}(\boldsymbol{r}_i), \\ \boldsymbol{g}_{h_i}^{tag} &= MLP_{head}^{tag}(\boldsymbol{r}_{h_i}), \\ \boldsymbol{\ell} &= (\boldsymbol{g}_i^{tag})^{\mathrm{T}} \boldsymbol{U}_{tag} \boldsymbol{g}_{h_i}^{tag} + W_{tag} \begin{bmatrix} \boldsymbol{g}_i^{tag} \\ \boldsymbol{g}_{h_i}^{tag} \end{bmatrix} + \boldsymbol{b}_{tag}, \\ P_{tag}(c_i|\boldsymbol{x}) &\propto \exp(\ell_c), \end{aligned} \tag{5}$$

where $\boldsymbol{U}_{tag}$ is a third order tensor. As in Lewis et al. these values can be precomputed before search, which makes our A* parsing quite efficient.

4 CCG to Dependency Conversion

Now we describe our conversion rules from a CCG tree to a dependency one, which we use in two purposes: 1) creation of the training data for the dependency component of our model; and 2) extraction of a dependency arc at each combinatory rule during A* search (Section 3.1). Lewis and Steedman (2014) describe one way to extract dependencies from a CCG tree (LEWISRULE). Below in addition to this we describe two simpler alternatives (HEADFIRST and HEADFINAL), and see the effects on parsing performance in our experiments (Section 6). See Figure 4 for the overview.

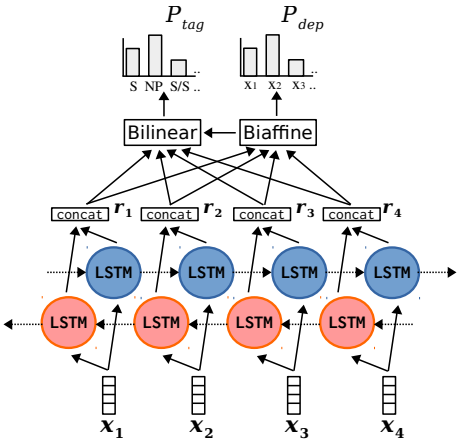


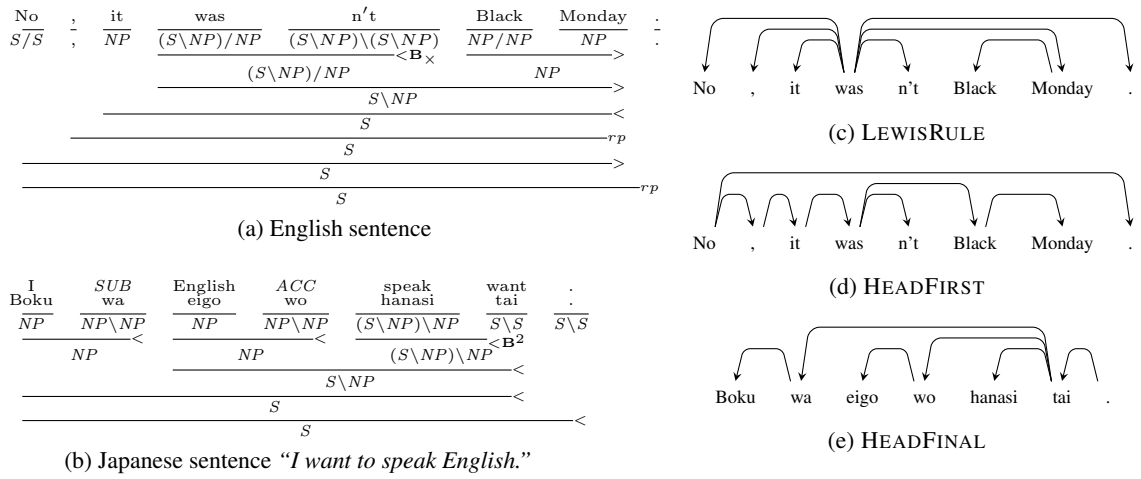
Figure 3: Neural networks of our supertag and dependency factored model. First we map every word x_i to a hidden vector $\boldsymbol{r}_i$ by bi-LSTMs, and then apply biaffine (Eq. 4) and bilinear (Eq. 5) transformations to obtain the distributions of dependency heads (P_{dep}) and supertags (P_{tag}).

LEWISRULE This is the same as the conversion rule in Lewis and Steedman (2014). As shown in Figure 4c the output looks a familiar English dependency tree.

For forward application and (generalized) forward composition, we define the head to be the left argument of the combinatory rule, unless it matches either X/X or $X/(X\backslash Y)$, in which case the right argument is the head. For example, on *"Black Monday"* in Figure 4a we choose *Monday* as the head of *Black*. For the backward rules, the conversions are defined as the reverse of the corresponding forward rules. For other rules, *RemovePunctuation* (*rp*) chooses the non punctuation argument as the head, while *Conjunction* (Φ) chooses the right argument.[3]

[2] This is inspired by the formulation of label prediction in Dozat and Manning (2016), which performs the best among other settings that remove or reverse the dependence between the head model and the supertag model.

[3] When applying LEWISRULE to Japanese, we ignore the feature values in determining the head argument, which we find often leads to a more natural dependency structure. For example, in "tabe ta" (eat *PAST*), the category of auxiliary verb "ta" is $S_{f_1}\backslash S_{f_2}$ with $f_1 \neq f_2$, and thus $S_{f_1} \neq S_{f_2}$. We choose "tabe" as the head in this case by removing the feature values, which makes the category $X\backslash X$.



(a) English sentence

(b) Japanese sentence *"I want to speak English."*

(c) LEWISRULE

(d) HEADFIRST

(e) HEADFINAL

Figure 4: Examples of applying conversion rules in Section 4 to English and Japanese sentences.

One issue when applying this method for obtaining the training data is that due to the mismatch between the rule set of our CCG parser, for which we follow Lewis and Steedman (2014), and the grammar in English CCGbank (Hockenmaier and Steedman, 2007) we cannot extract dependencies from some of annotated CCG trees.[4] For this reason, we instead obtain the training data for this method from the original dependency annotations on CCGbank. Fortunately the dependency annotations of CCGbank matches LEWISRULE above in most cases and thus they can be a good approximation to it.

HEADFINAL Among SOV languages, Japanese is known as a strictly head final language, meaning that the head of every word always follows it. Japanese dependency parsing (Uchimoto et al., 1999; Kudo and Matsumoto, 2002) has exploited this property explicitly by only allowing left-to-right dependency arcs. Inspired by this tradition, we try a simple HEADFINAL rule in Japanese CCG parsing, in which we always select the right argument as the head. For example we obtain the head final dependency tree in Figure 4e from the Japanese CCG tree in Figure 4b.

HEADFIRST We apply the similar idea as HEADFINAL into English. Since English has the opposite, SVO word order, we define the simple "head first" rule, in which the left argument always becomes the head (Figure 4d).

Though this conversion may look odd at first sight it also has some advantages over LEWISRULE. First, since the model with LEWISRULE is trained on the CCGbank dependencies, at inference, occasionally the two components P_{dep} and P_{tag} cause some conflicts on their predictions. For example, the true Viterbi parse may have a lower score in terms of dependencies, in which case the parser slows down and may degrade the accuracy. HEADFIRST, in contract, does not suffer from such conflicts. Second, by fixing the direction of arcs, the prediction of heads becomes easier, meaning that the dependency predictions become more reliable. Later we show that this is in fact the case for existing dependency parsers (see Section 5), and in practice, we find that this simple conversion rule leads to the higher parsing scores than LEWISRULE on English (Section 6).

5 Tri-training

We extend the existing tri-training method to our models and apply it to our English parsers.

Tri-training is one of the semi-supervised methods, in which the outputs of two parsers on unlabeled data are intersected to create (silver) new training data. This method is successfully applied to dependency parsing (Weiss et al., 2015) and CCG supertagging (Lewis et al., 2016).

We simply combine the two previous approaches. Lewis et al. (2016) obtain their silver data annotated with the high quality supertags. Since they make this data publicly available [5], we obtain our silver data by assigning dependency

[4] For example, the combinatory rules in Lewis and Steedman (2014) do not contain $N_{conj} \rightarrow N\ N$ in CCGbank. Another difficulty is that in English CCGbank the name of each combinatory rule is not annotated explicitly.

[5] https://github.com/uwnlp/taggerflow

structures on top of them.[6]

We train two very different dependency parsers from the training data extracted from CCGbank Section 02-21. This training data differs depending on our dependency conversion strategies (Section 4). For LEWISRULE, we extract the original dependency annotations of CCGbank. For HEADFIRST, we extract the head first dependencies from the CCG trees. Note that we cannot annotate dependency labels so we assign a dummy "none" label to every arc. The first parser is graph-based `RBGParser` (Lei et al., 2014) with the default settings except that we train an unlabeled parser and use word embeddings of Turian et al. (2010). The second parser is transition-based `lstm-parser` (Dyer et al., 2015) with the default parameters.

On the development set (Section 00), with LEWISRULE dependencies `RBGParser` shows 93.8% unlabeled attachment score while that of `lstm-parser` is 92.5% using gold POS tags. Interestingly, the parsers with HEADFIRST dependencies achieve higher scores: 94.9% by `RBGParser` and 94.6% by `lstm-parser`, suggesting that HEADFIRST dependencies are easier to parse. For both dependencies, we obtain more than 1.7 million sentences on which two parsers agree.

Following Lewis et al. (2016), we include 15 copies of CCGbank training set when using these silver data. Also to make effects of the tri-train samples smaller we multiply their loss by 0.4.

6 Experiments

We perform experiments on English and Japanese CCGbanks.

6.1 English Experimental Settings

We follow the standard data splits and use Sections 02-21 for training, Section 00 for development, and Section 23 for final evaluation. We report labeled and unlabeled F1 of the extracted CCG semantic dependencies obtained using `generate` program supplied with `C&C` parser.

For our models, we adopt the pruning strategies in Lewis and Steedman (2014) and allow at most 50 categories per word, use a variable-width beam with $\beta = 0.00001$, and utilize a tag dictionary, which maps frequent words to the possible supertags[7]. Unless otherwise stated, we only allow normal form parses (Eisner, 1996; Hockenmaier and Bisk, 2010), choosing the same subset of the constraints as Lewis and Steedman (2014).

We use as word representation the concatenation of word vectors initialized to GloVe[8] (Pennington et al., 2014), and randomly initialized prefix and suffix vectors of the length 1 to 4, which is inspired by Lewis et al. (2016). All affixes appearing less than two times in the training data are mapped to "UNK".

Other model configurations are: 4-layer bi-LSTMs with left and right 300-dimensional LSTMs, 1-layer 100-dimensional MLPs with ELU non-linearity (Clevert et al., 2015) for all MLP^{dep}_{child}, MLP^{dep}_{head}, MLP^{tag}_{child} and MLP^{tag}_{head}, and the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.9$, L2 norm ($1e^{-6}$), and learning rate decay with the ratio 0.75 for every 2,500 iteration starting from $2e^{-3}$, which is shown to be effective for training the biaffine parser (Dozat and Manning, 2016).

6.2 Japanese Experimental Settings

We follow the default train/dev/test splits of Japanese CCGbank (Uematsu et al., 2013). For the baselines, we use an existing shift-reduce CCG parser implemented in an NLP tool Jigg[9] (Noji and Miyao, 2016), and our implementation of the supertag-factored model using bi-LSTMs.

For Japanese, we use as word representation the concatenation of word vectors initialized to Japanese Wikipedia Entity Vector[10], and 100-dimensional vectors computed from randomly initialized 50-dimensional character embeddings through convolution (dos Santos and Zadrozny, 2014). We do not use affix vectors as affixes are less informative in Japanese. All characters appearing less than two times are mapped to "UNK". We use the same parameter settings as English for bi-LSTMs, MLPs, and optimization.

One issue in Japanese experiments is evaluation. The Japanese CCGbank is encoded in a different format than the English bank, and no standalone script for extracting semantic dependencies is available yet. For this reason, we evaluate the parser outputs by converting them to *bunsetsu*

[6] We annotate POS tags on this data using Stanford POS tagger (Toutanova et al., 2003).

[7] We use the same tag dictionary provided with their bi-LSTM model.

[8] http://nlp.stanford.edu/projects/glove/

[9] https://github.com/mynlp/jigg

[10] http://www.cl.ecei.tohoku.ac.jp/~m-suzuki/jawiki_vector/

Method	Labeled	Unlabeled
CCGbank		
LEWISRULE w/o dep	85.8	91.7
LEWISRULE	86.0	92.5
HEADFIRST w/o dep	85.6	91.6
HEADFIRST	**86.6**	**92.8**
Tri-training		
LEWISRULE	86.9	93.0
HEADFIRST	**87.6**	**93.3**

Table 1: Parsing results (F1) on English development set. "w/o dep" means that the model discards dependency components at prediction.

Method	Labeled	Unlabeled	# violations
CCGbank			
LEWISRULE w/o dep	85.8	91.7	2732
LEWISRULE	85.4	92.2	283
HEADFIRST w/o dep	85.6	91.6	2773
HEADFIRST	**86.8**	**93.0**	**89**
Tri-training			
LEWISRULE	86.7	92.8	253
HEADFIRST	**87.7**	**93.5**	**66**

Table 2: Parsing results (F1) on English development set when excluding the normal form constraints. # violations is the number of combinations violating the constraints on the outputs.

dependencies, the syntactic representation ordinary used in Japanese NLP (Kudo and Matsumoto, 2002). Given a CCG tree, we obtain this by first segment a sentence into bunsetsu (chunks) using CaboCha[11] and extract dependencies that cross a bunsetsu boundary after obtaining the word-level, head final dependencies as in Figure 4b. For example, the sentence in Figure 4e is segmented as "*Boku wa* | *eigo wo* | *hanashi tai*", from which we extract two dependencies (*Boku wa*) ← (*hanashi tai*) and (*eigo wo*) ← (*hanashi tai*). We perform this conversion for both gold and output CCG trees and calculate the (unlabeled) attachment accuracy. Though this is imperfect, it can detect important parse errors such as attachment errors and thus can be a good proxy for the performance as a CCG parser.

6.3 English Parsing Results

Effect of Dependency We first see how the dependency components added in our model affect the performance. Table 1 shows the results on the development set with the several configurations, in which "w/o dep" means discarding the dependency terms of the model and applying the attach low heuristics (Section 1) instead (i.e., a supertag-factored model; Section 2.1). We can see that for both LEWISRULE and HEADFIRST, adding dependency terms improves the performance.

[11] http://taku910.github.io/cabocha/

Method	Labeled	Unlabeled
CCGbank		
C&C (Clark and Curran, 2007)	85.5	91.7
w/ LSTMs (Vaswani et al., 2016)	**88.3**	-
EasySRL (Lewis et al., 2016)	87.2	-
EasySRL_reimpl	86.8	92.3
HEADFIRST w/o NF (Ours)	87.7	**93.4**
Tri-training		
EasySRL (Lewis et al., 2016)	88.0	92.9
neuralccg (Lee et al., 2016)	88.7	93.7
HEADFIRST w/o NF (Ours)	**88.8**	**94.0**

Table 3: Parsing results (F1) on English test set (Section 23).

Choice of Dependency Conversion Rule To our surprise, our simple HEADFIRST strategy always leads to better results than the linguistically motivated LEWISRULE. The absolute improvements by tri-training are equally large (about 1.0 points), suggesting that our model with dependencies can also benefit from the silver data.

Excluding Normal Form Constraints One advantage of HEADFIRST is that the direction of arcs is always right, making the structures simpler and more parsable (Section 5). From another viewpoint, this fixed direction means that the constituent structure behind a (head first) dependency tree is unique. Since the constituent structures of CCGbank trees basically follow the normal form (NF), we hypothesize that the model learned with HEADFIRST has an ability to force the outputs in NF automatically. We summarize the results without the NF constraints in Table 2, which shows that the above argument is correct; the number of violating NF rules on the outputs of HEADFIRST is much smaller than that of LEWISRULE (89 vs. 283). Interestingly the scores of HEADFIRST slightly increase from the models with NF (e.g., 86.8 vs. 86.6 for CCGbank), suggesting that the NF constraints hinder the search of HEADFIRST models occasionally.

Results on Test Set Parsing results on the test set (Section 23) are shown in Table 3, where we compare our best performing HEADFIRST dependency model without NF constraints with the several existing parsers. In the CCGbank experi-

	EasySRL_reimpl	neuralccg	Ours
Tagging	24.8	21.7	16.6
A Search*	185.2	16.7	114.6
Total	21.9	9.33	14.5

Table 4: Results of the efficiency experiment, where each number is the number of sentences processed per second. We compare our proposed parser against neuralccg and our reimplementation of EasySRL.

ment, our parser shows the better result than all the baseline parsers except C&C with an LSTM supertagger (Vaswani et al., 2016). Our parser outperforms EasySRL by 0.5% and our reimplementation of that parser (EasySRL_reimpl) by 0.9% in terms of labeled F1. In the tri-training experiment, our parser shows much increased performance of 88.8% labeled F1 and 94.0% unlabeled F1, outperforming the current state-of-the-art neuralccg (Lee et al., 2016) that uses recursive neural networks by 0.1 point and 0.3 point in terms of labeled and unlabeled F1. This is the best reported F1 in English CCG parsing.

Efficiency Comparison We compare the efficiency of our parser with neuralccg and EasySRL_reimpl.[12] The results are shown in Table 4. For the overall speed (the third row), our parser is faster than neuralccg although lags behind EasySRL_reimpl. Inspecting the details, our supertagger runs slower than those of neuralccg and EasySRL_reimpl, while in A* search our parser processes over 7 times more sentences than neuralccg. The delay in supertagging can be attributed to several factors, in particular the differences in network architectures including the number of bi-LSTM layers (4 vs. 2) and the use of bilinear transformation instead of linear one. There are also many implementation differences in our parser (C++ A* parser with neural network model implemented with Chainer (Tokui et al., 2015)) and neuralccg (Java parser with C++ TensorFlow (Abadi et al., 2015) supertagger and recursive neural model in C++ DyNet (Neubig et al., 2017)).

6.4 Japanese Parsing Result

We show the results of the Japanese parsing experiment in Table 5. The simple application of Lewis et al. (2016) (Supertag model) is not effective for Japanese, showing the lowest attachment score of 81.5%. We observe a performance boost with our method, especially with HEADFINAL dependencies, which outperforms the baseline shift-reduce parser by 1.1 points on category assignments and 4.0 points on bunsetsu dependencies.

[12] This experiment is performed on a laptop with 4-thread 2.0 GHz CPU.

Method	Category	Bunsetsu Dep.
Noji and Miyao (2016)	93.0	87.5
Supertag model	93.7	81.5
LEWISRULE (Ours)	93.8	90.8
HEADFINAL (Ours)	**94.1**	**91.5**

Table 5: Results of Japanese CCGbank.

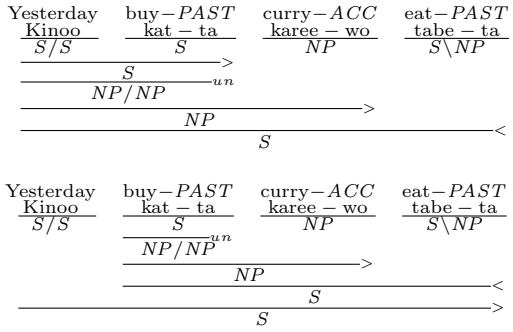


Figure 5: Examples of ambiguous Japanese sentence given fixed supertags. The English translation is *"I ate the curry I bought yesterday"*.

The degraded results of the simple application of the supertag-factored model can be attributed to the fact that the structure of a Japanese sentence is still highly ambiguous given the supertags (Figure 5). This is particularly the case in constructions where phrasal adverbial/adnominal modifiers (with the supertag S/S) are involved. The result suggests the importance of modeling dependencies in some languages, at least Japanese.

7 Related Work

There is some past work that utilizes dependencies in lexicalized grammar parsing, which we review briefly here.

For Head-driven Phrase Structure Grammar (HPSG; Pollard and Sag (1994)), there are studies to use the predicted dependency structure to improve HPSG parsing accuracy. Sagae et al. (2007) use dependencies to constrain the form of the output tree. As in our method, for every rule (schema) application they define which child becomes the head and impose a soft constraint that these dependencies agree with the output of the dependency parser. Our method is different

in that we do not use the one-best dependency structure alone, but rather we search for a CCG tree that is optimal in terms of dependencies and CCG supertags. Zhang et al. (2010) use the syntactic dependencies in a different way, and show that dependency-based features are useful for predicting HPSG supertags.

In the CCG parsing literature, some work optimizes a *dependency model*, instead of supertags or a derivation (Clark and Curran, 2007; Xu et al., 2014). This approach is reasonable given that the objective matches the evaluation metric. Instead of modeling dependencies alone, our method finds a CCG derivation that has a higher dependency score. Lewis et al. (2015) present a joint model of CCG parsing and semantic role labeling (SRL), which is closely related to our approach. They map each CCG semantic dependency to an SRL relation, for which they give the A* upper bound by the score from a predicate to the most probable argument. Our approach is similar; the largest difference is that we instead model syntactic dependencies from each token to its head, and this is the key to our success. Since dependency parsing can be formulated as independent head selections similar to tagging, we can build the entire model on LSTMs to exploit features from the whole sentence. This formulation is not straightforward in the case of multi-headed semantic dependencies in their model.

8 Conclusion

We have presented a new A* CCG parsing method, in which the probability of a CCG tree is decomposed into local factors of the CCG categories and its dependency structure. By explicitly modeling the dependency structure, we do not require any deterministic heuristics to resolve attachment ambiguities, and keep the model locally factored so that all the probabilities can be precomputed before running the search. Our parser efficiently finds the optimal parse and achieves the state-of-the-art performance in both English and Japanese parsing.

Acknowledgments

We are grateful to Mike Lewis for answering our questions and your Github repository from which we learned many things. We also thank Yuichiro Sawai for the faster LSTM implementation. This work was in part supported by JSPS KAKENHI Grant Number 16H06981, and also by JST CREST Grant Number JPMJCR1301.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. http://tensorflow.org/.

Srinivas Bangalore and Aravind K Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational linguistics* 25(2):237–265.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics, Volume 33, Number 4, December 2007* http://aclweb.org/anthology/J07-4004.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *CoRR* abs/1511.07289. http://arxiv.org/abs/1511.07289.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning Character-level Representations for Part-of-Speech Tagging. ICML.

Timothy Dozat and Christopher D. Manning. 2016. Deep Biaffine Attention for Neural Dependency Parsing. *CoRR* abs/1611.01734. http://arxiv.org/abs/1611.01734.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and A. Noah Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 334–343. https://doi.org/10.3115/v1/P15-1033.

Jason Eisner. 1996. Efficient Normal-Form Parsing for Combinatory Categorial Grammar. In *34th Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P96-1011.

Julia Hockenmaier and Yonatan Bisk. 2010. Normal-form parsing for Combinatory Categorial Grammars with generalized composition and type-raising. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, pages 465–473. http://aclweb.org/anthology/C10-1053.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics* 33(3):355–396. http://www.aclweb.org/anthology/J07-3004.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://www.transacl.org/ojs/index.php/tacl/article/view/885.

Dan Klein and Christopher D. Manning. 2003. A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. http://aclweb.org/anthology/N03-1016.

Taku Kudo and Yuji Matsumoto. 2002. Japanese Dependency Analysis using Cascaded Chunking. In *Proceedings of the 6th Conference on Natural Language Learning, CoNLL 2002, Held in cooperation with COLING 2002, Taipei, Taiwan, 2002*. http://aclweb.org/anthology/W/W02/W02-2016.pdf.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2016. Global Neural CCG Parsing with Optimality Guarantees. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2366–2376. http://aclweb.org/anthology/D16-1262.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-Rank Tensors for Scoring Dependency Structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1381–1391. https://doi.org/10.3115/v1/P14-1130.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG Parsing and Semantic Role Labelling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1444–1454. https://doi.org/10.18653/v1/D15-1169.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 221–231. https://doi.org/10.18653/v1/N16-1026.

Mike Lewis and Mark Steedman. 2014. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 990–1000. https://doi.org/10.3115/v1/D14-1107.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980* .

Hiroshi Noji and Yusuke Miyao. 2016. Jigg: A Framework for an Easy Natural Language Processing Pipeline. In *Proceedings of ACL-2016 System Demonstrations*. Association for Computational Linguistics, pages 103–108. https://doi.org/10.18653/v1/P16-4018.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Kenji Sagae, Yusuke Miyao, and Jun'ichi Tsujii. 2007. HPSG Parsing with Shallow Dependency Constraints. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, pages 624–631. http://aclweb.org/anthology/P07-1079.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a Next-Generation Open Source Framework for Deep Learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*. http://learningsys.org/papers/LearningSys_2015_paper_33.pdf.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational*

Linguistics. http://www.aclweb.org/anthology/N03-1033.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 384–394. http://aclweb.org/anthology/P10-1040.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 1999. Japanese Dependency Structure Analysis Based on Maximum Entropy Models. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*. http://aclweb.org/anthology/E99-1026.

Sumire Uematsu, Takuya Matsuzaki, Hiroki Hanaoka, Yusuke Miyao, and Hideki Mima. 2013. Integrating Multiple Dependency Corpora for Inducing Wide-coverage Japanese CCG Resources. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1042–1051. http://www.aclweb.org/anthology/P13-1103.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging With LSTMs. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 232–237. https://doi.org/10.18653/v1/N16-1027.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured Training for Neural Network Transition-Based Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 323–333. https://doi.org/10.3115/v1/P15-1032.

Wenduan Xu, Stephen Clark, and Yue Zhang. 2014. Shift-Reduce CCG Parsing with a Dependency Model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 218–227. https://doi.org/10.3115/v1/P14-1021.

Yao-zhong Zhang, Takuya Matsuzaki, and Jun'ichi Tsujii. 2010. A Simple Approach for HPSG Supertagging Using Dependency Information. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 645–648. http://aclweb.org/anthology/N10-1090.

A Full Non-Monotonic Transition System for Unrestricted Non-Projective Parsing

Daniel Fernández-González and **Carlos Gómez-Rodríguez**
Universidade da Coruña
FASTPARSE Lab, LyS Research Group, Departamento de Computación
Campus de Elviña, s/n, 15071 A Coruña, Spain
d.fgonzalez@udc.es, carlos.gomez@udc.es

Abstract

Restricted non-monotonicity has been shown beneficial for the projective arc-eager dependency parser in previous research, as posterior decisions can repair mistakes made in previous states due to the lack of information. In this paper, we propose a novel, fully non-monotonic transition system based on the non-projective Covington algorithm. As a non-monotonic system requires exploration of erroneous actions during the training process, we develop several non-monotonic variants of the recently defined dynamic oracle for the Covington parser, based on tight approximations of the loss. Experiments on datasets from the CoNLL-X and CoNLL-XI shared tasks show that a non-monotonic dynamic oracle outperforms the monotonic version in the majority of languages.

1 Introduction

Greedy transition-based dependency parsers are widely used in different NLP tasks due to their speed and efficiency. They parse a sentence from left to right by greedily choosing the highest-scoring transition to go from the current parser configuration or state to the next. The resulting sequence of transitions incrementally builds a parse for the input sentence. The scoring of the transitions is provided by a statistical model, previously trained to approximate an *oracle*, a function that selects the needed transitions to parse a gold tree.

Unfortunately, the greedy nature that grants these parsers their efficiency also represents their main limitation. McDonald and Nivre (2007) show that greedy transition-based parsers lose accuracy to error propagation: a transition erroneously chosen by the greedy parser can place it in an incorrect and unknown configuration, causing more mistakes in the rest of the transition sequence. Training with a dynamic oracle (Goldberg and Nivre, 2012) improves robustness in these situations, but in a monotonic transition system, erroneous decisions made in the past are permanent, even when the availability of further information in later states might be useful to correct them.

Honnibal et al. (2013) show that allowing some degree of non-monotonicity, by using a limited set of non-monotonic actions that can repair past mistakes and replace previously-built arcs, can increase the accuracy of a transition-based parser. In particular, they present a modified arc-eager transition system where the Left-Arc and Reduce transitions are non-monotonic: the former is used to repair invalid attachments made in previous states by replacing them with a leftward arc, and the latter allows the parser to link two words with a rightward arc that were previously left unattached due to an erroneous decision. Since the Right-Arc transition is still monotonic and leftward arcs can never be repaired because their dependent is removed from the stack by the arc-eager parser and rendered inaccessible, this approach can only repair certain kinds of mistakes: namely, it can fix erroneous rightward arcs by replacing them with a leftward arc, and connect a limited set of unattached words with rightward arcs. In addition, they argue that non-monotonicity in the training oracle can be harmful for the final accuracy and, therefore, they suggest to apply it only as a fallback component for a monotonic oracle, which is given priority over the non-monotonic one. Thus, this strategy will follow the path dictated by the monotonic oracle the majority of the time. Honnibal and Johnson (2015) present an extension of this transition system with an Unshift transition allowing it some extra flexibility to correct past errors. However, the restriction that only rightward

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 288–298
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1027

arcs can be deleted, and only by replacing them with leftward arcs, is still in place. Furthermore, both versions of the algorithm are limited to projective trees.

In this paper, we propose a non-monotonic transition system based on the non-projective Covington parser, together with a dynamic oracle to train it with erroneous examples that will need to be repaired. Unlike the system developed in (Honnibal et al., 2013; Honnibal and Johnson, 2015), we work with full non-monotonicity. This has a twofold meaning: (1) our approach can repair previous erroneous attachments regardless of their original direction, and it can replace them either with a rightward or leftward arc as both arc transitions are non-monotonic;[1] and (2) we use exclusively a non-monotonic oracle, without the interferences of monotonic decisions. These modifications are feasible because the non-projective Covington transition system is less rigid than the arc-eager algorithm, as words are never deleted from the parser's data structures and can always be revisited, making it a better option to exploit the full potencial of non-monotonicity. To our knowledge, the presented system is the first non-monotonic parser that can produce non-projective dependency analyses. Another novel aspect is that our dynamic oracle is *approximate*, i.e., based on efficiently-computable approximations of the loss due to the complexity of calculating its actual value in a non-monotonic and non-projective scenario. However, this is not a problem in practice: experimental results show how our parser and oracle can use non-monotonic actions to repair erroneous attachments, outperforming the monotonic version developed by Gómez-Rodríguez and Fernández-González (2015) in a large majority of the datasets tested.

2 Preliminaries

2.1 Non-Projective Covington Transition System

The non-projective Covington parser was originally defined by Covington (2001), and then recast by Nivre (2008) under the transition-based parsing framework.

[1]The only restriction is that parsing must still proceed in left-to-right order. For this reason, a leftward arc cannot be repaired with a rightward arc, because this would imply going back in the sentence. The other three combinations (replacing leftward with leftward, rightward with leftward or rightward with rightward arcs) are possible.

The transition system that defines this parser is as follows: each parser configuration is of the form $c = \langle \lambda_1, \lambda_2, B, A \rangle$, such that λ_1 and λ_2 are lists of partially processed words, B is another list (called the buffer) containing currently unprocessed words, and A is the set of dependencies that have been built so far. Suppose that our input is a string $w_1 \cdots w_n$, whose word occurrences will be identified with their indices $1 \cdots n$ for simplicity. Then, the parser will start at an initial configuration $c_s(w_1 \dots w_n) = \langle [], [], [1 \dots n], \emptyset \rangle$, and execute transitions chosen from those in Figure 1 until a terminal configuration of the form $\{\langle \lambda_1, \lambda_2, [], A \rangle \in C\}$ is reached. At that point, the sentence's parse tree is obtained from A.[2]

These transitions implement the same logic as the double nested loop traversing word pairs in the original formulation by Covington (2001). When the parser's configuration is $\langle \lambda_1 | i, \lambda_2, j | B, A \rangle$, we say that it is considering the **focus words** i and j, located at the end of the first list and at the beginning of the buffer. At that point, the parser must decide whether these two words should be linked with a leftward arc $i \leftarrow j$ (Left-Arc transition), a rightward arc $i \rightarrow j$ (Right-Arc transition), or not linked at all (No-Arc transition). However, the two transitions that create arcs will be disallowed in configurations where this would cause a violation of the **single-head constraint** (a node can have at most one incoming arc) or the **acyclicity constraint** (the dependency graph cannot have cycles). After applying any of these three transitions, i is moved to the second list to make $i - 1$ and j the focus words for the next step. As an alternative, we can instead choose to execute a Shift transition which lets the parser read a new input word, placing the focus on j and $j + 1$.

The resulting parser can generate any possible dependency tree for the input, including arbitrary non-projective trees. While it runs in quadratic worst-case time, in theory worse than linear-time transition-based parsers (e.g. (Nivre, 2003; Gómez-Rodríguez and Nivre, 2013)), it has been shown to outspeed linear algorithms in practice, thanks to feature extraction optimizations that cannot be implemented in other parsers (Volokh and Neumann, 2012). In fact, one of the fastest dependency parsers ever reported uses this algorithm

[2]In general A is a forest, but it can be converted to a tree by linking headless nodes as dependents of an artificial root node at position 0. When we refer to parser outputs as trees, we assume that this transformation is being implicitly made.

Shift:	$\langle \lambda_1, \lambda_2, j\|B, A\rangle \Rightarrow \langle \lambda_1 \cdot \lambda_2\|j, [], B, A\rangle$
No-Arc:	$\langle \lambda_1\|i, \lambda_2, B, A\rangle \Rightarrow \langle \lambda_1, i\|\lambda_2, B, A\rangle$
Left-Arc:	$\langle \lambda_1\|i, \lambda_2, j\|B, A\rangle \Rightarrow \langle \lambda_1, i\|\lambda_2, j\|B, A \cup \{j \rightarrow i\}\rangle$ only if $\nexists k \mid k \rightarrow i \in A$ (single-head) and $i \rightarrow^* j \notin A$ (acyclicity).
Right-Arc:	$\langle \lambda_1\|i, \lambda_2, j\|B, A\rangle \Rightarrow \langle \lambda_1, i\|\lambda_2, j\|B, A \cup \{i \rightarrow j\}\rangle$ only if $\nexists k \mid k \rightarrow j \in A$ (single-head) and $j \rightarrow^* i \notin A$ (acyclicity).

Figure 1: Transitions of the monotonic Covington non-projective dependency parser. The notation $i \rightarrow^* j \in A$ means that there is a (possibly empty) directed path from i to j in A.

(Volokh, 2013).

2.2 Monotonic Dynamic Oracle

A dynamic oracle is a function that maps a configuration c and a gold tree t_G to the set of transitions that can be applied in c and lead to some parse tree t minimizing the Hamming loss with respect to t_G (the amount of nodes whose head is different in t and t_G). Following Goldberg and Nivre (2013), we say that an arc set A is **reachable** from configuration c, and we write $c \rightsquigarrow A$, if there is some (possibly empty) path of transitions from c to some configuration $c' = \langle \lambda_1, \lambda_2, B, A'\rangle$, with $A \subseteq A'$. Then, we can define the loss of configuration c as

$$\ell(c) = \min_{t|c \rightsquigarrow t} \mathcal{L}(t, t_G),$$

and therefore, a correct dynamic oracle will return the set of transitions

$$o_d(c, t_G) = \{\tau \mid \ell(c) - \ell(\tau(c)) = 0\},$$

i.e., the set of transitions that do not increase configuration loss, and thus lead to the best parse (in terms of loss) reachable from c. Hence, implementing a dynamic oracle reduces to computing the loss $\ell(c)$ for each configuration c.

Goldberg and Nivre (2013) show a straightforward method to calculate loss for parsers that are **arc-decomposable**, i.e., those where every arc set A that can be part of a well-formed parse verifies that if $c \rightsquigarrow (i \rightarrow j)$ for every $i \rightarrow j \in A$ (i.e., each of the individual arcs of A is reachable from a given configuration c), then $c \rightsquigarrow A$ (i.e., the set A as a whole is reachable from c). If this holds, then the loss of a configuration c equals the number of gold arcs that are *not* individually reachable from c, which is easy to compute in most parsers.

Gómez-Rodríguez and Fernández-González (2015) show that the non-projective Covington parser is not arc-decomposable because sets of individually reachable arcs may form cycles together with already-built arcs, preventing them from being jointly reachable due to the acyclicity constraint. In spite of this, they prove that a dynamic oracle for the Covington parser can be efficiently built by counting individually unreachable arcs, and correcting for the presence of such cycles. Concretely, the loss is computed as:

$$\ell(c) = |\mathcal{U}(c, t_G)| + n_c(A \cup \mathcal{I}(c, t_G))$$

where $\mathcal{I}(c, t_G) = \{x \rightarrow y \in t_G \mid c \rightsquigarrow (x \rightarrow y)\}$ is the set of **individually reachable arcs** of t_G from configuration c; $\mathcal{U}(c, t_G)$ is the set of **individually unreachable arcs** of t_G from c, computed as $t_G \backslash \mathcal{I}(c, t_G)$; and $n_c(G)$ denotes the number of cycles in a graph G.

Therefore, to calculate the loss of a configuration c, we only need to compute the two terms $|\mathcal{U}(c, t_G)|$ and $n_c(A \cup \mathcal{I}(c, t_G))$. To calculate the first term, given a configuration c with focus words i and j (i.e., $c = \langle \lambda_1|i, \lambda_2, j|B, A\rangle$), an arc $x \rightarrow y$ will be in $\mathcal{U}(c, t_G)$ if it is not in A, and at least one of the following holds:

- $j > \max(x, y)$, (i.e., we have read too far in the string and can no longer get $\max(x, y)$ as right focus word),
- $j = \max(x, y) \wedge i < \min(x, y)$, (i.e., we have $\max(x, y)$ as the right focus word but the left focus word has already moved left past $\min(x, y)$, and we cannot go back),
- there is some $z \neq 0, z \neq x$ such that $z \rightarrow y \in A$, (i.e., we cannot create $x \rightarrow y$ because it would violate the single-head constraint),
- x and y are on the same weakly connected component of A (i.e., we cannot create $x \rightarrow y$ due to the acyclicity constraint).

The second term of the loss, $n_c(A \cup \mathcal{I}(c, t_G))$, can be computed by first obtaining $\mathcal{I}(c, t_G)$ as $t_G \setminus \mathcal{U}(c, t_G)$. Since the graph $\mathcal{I}(c, t_G)$ has in-degree 1, the algorithm by Tarjan (1972) can then be used to find and count the cycles in $O(n)$ time.

Algorithm 1 Computation of the loss of a configuration in the monotonic oracle.

function LOSS($c = \langle \lambda_1 | i, \lambda_2, j | B, A \rangle, t_G$)
  $U \leftarrow \emptyset$ ▷ Variable U is for $\mathcal{U}(c, t_G)$
  for each $x \rightarrow y \in (t_G \setminus A)$ **do**
    $left \leftarrow \min(x, y)$
    $right \leftarrow \max(x, y)$
    if $j > right \vee$
      $(j = right \wedge i < left) \vee$
      $(\exists z > 0, z \neq x : z \rightarrow y \in A) \vee$
      WEAKLYCONNECTED(A, x, y) **then**
        $U \leftarrow u \cup \{x \rightarrow y\}$
  $I \leftarrow t_G \setminus U$ ▷ Variable I is for $\mathcal{I}(c, t_G)$
  return $|U| +$ COUNTCYCLES$(A \cup I)$

Algorithm 1 shows the resulting loss calculation algorithm, where COUNTCYCLES is a function that counts the number of cycles in the given graph and WEAKLYCONNECTED returns whether two given nodes are weakly connected in A.

3 Non-Monotonic Transition System for the Covington Non-Projective Parser

We now define a non-monotonic variant of the Covington non-projective parser. To do so, we allow the Right-Arc and Left-Arc transitions to create arcs between any pair of nodes without restriction. If the node attached as dependent already had a previous head, the existing attachment is discarded in favor of the new one. This allows the parser to correct erroneous attachments made in the past by assigning new heads, while still enforcing the single-head constraint, as only the most recent head assigned to each node is kept.

To enforce acyclicity, one possibility would be to keep the logic of the monotonic algorithm, forbidding the creation of arcs that would create cycles. However, this greatly complicates the definition of the set of individually unreachable arcs, which is needed to compute the loss bounds that will be used by the dynamic oracle. This is because a gold arc $x \rightarrow y$ may superficially seem unreachable due to forming a cycle together with arcs in A, but it might in fact be reachable if there is some transition sequence that first breaks the cycle using non-monotonic transitions to remove arcs from A, to then create $x \rightarrow y$. We do not know of a way to characterize the conditions under which such a transition sequence exists, and thus cannot estimate the loss efficiently.

Instead, we enforce the acyclicity constraint in a similar way to the single-head constraint: Right-Arc and Left-Arc transitions are always allowed, even if the prospective arc would create a cycle in A. However, if the creation of a new arc $x \rightarrow y$ generates a cycle in A, we immediately remove the arc of the form $z \rightarrow x$ from A (which trivially exists, and is unique due to the single-head constraint). This not only enforces the acyclicity constraint while keeping the computation of $\mathcal{U}(c, t_G)$ simple and efficient, but also produces a straightforward, coherent algorithm (arc transitions are always allowed, and both constraints are enforced by deleting a previous arc) and allows us to exploit non-monotonicity to the maximum (we can not only recover from assigning a node the wrong head, but also from situations where previous errors together with the acyclicity constraint prevent us from building a gold arc, keeping with the principle that later decisions override earlier ones).

In Figure 2, we can see the resulting non-monotonic transition system for the non-projective Covington algorithm, where, unlike the monotonic version, all transitions are allowed at each configuration, and the single-head and acyclicity constraints are kept in A by removing offending arcs.

4 Non-Monotonic Approximate Dynamic Oracle

To successfully train a non-monotonic system, we need a dynamic oracle with error exploration, so that the parser will be put in erroneous states and need to apply non-monotonic transitions in order to repair them. To achieve that, we modify the dynamic oracle defined by Gómez-Rodríguez and Fernández-González (2015) so that it can deal with non-monotonicity. Our modification is an **approximate** dynamic oracle: due to the extra flexibility added to the algorithm by non-monotonicity, we do not know of an efficient way of obtaining an exact calculation of the loss of a given configuration. Instead, we use upper or lower bounds on the loss, which we empirically show to be very tight (less that 1% relative error with respect to the real loss) and are sufficient for the algorithm to provide better accuracy than the exact monotonic oracle.

First of all, we adapt the computation of the set of **individually unreachable arcs** $\mathcal{U}(c, t_G)$ to the new algorithm. In particular, if c has focus words i and j (i.e., $c = \langle \lambda_1 | i, \lambda_2, j | B, A \rangle$), then an arc $x \rightarrow y$ is in $\mathcal{U}(c, t_G)$ if it is not in A, and at least one of the following holds:

- $j > \max(x, y)$, (i.e., we have read too far in the string and can no longer get $\max(x, y)$ as

Shift:	$\langle \lambda_1, \lambda_2, j\|B, A\rangle \Rightarrow \langle \lambda_1 \cdot \lambda_2\|j, [], B, A\rangle$
No-Arc:	$\langle \lambda_1\|i, \lambda_2, B, A\rangle \Rightarrow \langle \lambda_1, i\|\lambda_2, B, A\rangle$
Left-Arc:	$\langle \lambda_1\|i, \lambda_2, j\|B, A\rangle \Rightarrow \langle \lambda_1, i\|\lambda_2, j\|B, (A \cup \{j \rightarrow i\}) \setminus \{x \rightarrow i \in A\} \setminus \{k \rightarrow j \in A \mid i \rightarrow^* k \in A\}\rangle$
Right-Arc:	$\langle \lambda_1\|i, \lambda_2, j\|B, A\rangle \Rightarrow \langle \lambda_1, i\|\lambda_2, j\|B, A \cup \{i \rightarrow j\} \setminus \{x \rightarrow j \in A\} \setminus \{k \rightarrow i \in A \mid j \rightarrow^* k \in A\}\rangle$

Figure 2: Transitions of the non-monotonic Covington non-projective dependency parser. The notation $i \rightarrow^* j \in A$ means that there is a (possibly empty) directed path from i to j in A.

right focus word),

- $j = \max(x, y) \wedge i < \min(x, y)$ (i.e., we have $\max(x, y)$ as the right focus word but the left focus word has already moved left past $\min(x, y)$, and we cannot move it back).

Note that, since the head of a node can change during the parsing process and arcs that produce cycles in A can be built, the two last conditions present in the monotonic scenario for computing $\mathcal{U}(c, t_G)$ are not needed when we use non-monotonicity and, as a consequence, the set of **individually reachable arcs** $\mathcal{I}(c, t_G)$ is larger: due to the greater flexibility provided by non-monotonicity, we can reach arcs that would be unreachable for the monotonic version.

Since arcs that are in this new $\mathcal{U}(c, t_G)$ are unreachable even by the non-monotonic parser, $|\mathcal{U}(c, t_G)|$ is trivially a **lower bound** of the loss $\ell(c)$. It is worth noting that there always exists at least one transition sequence that builds every arc in $\mathcal{I}(c, t_G)$ at some point (although not all of them necessarily appear in the final tree, due to non-monotonicity). This can be easily shown based on the fact that the non-monotonic parser does not forbid transitions at any configuration. Thanks to this, we can can generate one such sequence by just applying the original Covington (2001) criteria (choose an arc transition whenever the focus words are linked in $\mathcal{I}(c, t_G)$, and otherwise Shift or No-Arc depending on whether the left focus word is the first word in the sentence or not), although this sequence is not necessarily optimal in terms of loss. In such a transition sequence, the gold arcs that are missed are (1) those in $\mathcal{U}(c, t_G)$, and (2) those that are removed by the cycle-breaking in Left-Arc and Right-Arc transitions. In practice configurations where (2) is needed are uncommon, so this lower bound is a very close approximation of the real loss, as will be seen empirically below.

This reasoning also helps us calculate an upper bound of the loss: in a transition sequence as described, if we *only* build the arcs in $\mathcal{I}(c, t_G)$ and none else, the amount of arcs removed by breaking cycles (2) cannot be larger than the number of cycles in $A \cup \mathcal{I}(c, t_G)$. This means that $|\mathcal{U}(c, t_G)| + n_c(A \cup \mathcal{I}(c, t_G))$ is an **upper bound** of the loss $\ell(c)$. Note that, contrary to the monotonic case, this expression does not always give us the exact loss, for several reasons: firstly, $A \cup \mathcal{I}(c, t_G)$ can have non-disjoint cycles (a node may have different heads in A and $\mathcal{I}$ since attachments are not permanent, contrary to the monotonic version) and thus removing a single arc may break more than one cycle; secondly, the removed arc can be a non-gold arc of A and therefore not incur loss; and thirdly, there may exist alternative transition sequences where a cycle in $A \cup \mathcal{I}(c, t_G)$ is broken early by non-monotonic configurations that change the head of a wrongly-attached node in A to a different (and also wrong) head,[3] removing the cycle before the cycle-breaking mechanism needs to come into play without incurring in extra errors. Characterizing the situations where such an alternative exists is the main difficulty for an exact calculation of the loss.

However, it is possible to obtain a closer upper bound to the real loss if we consider the following: for each cycle in $A \cup \mathcal{I}(c, t_G)$ that will be broken by the transition sequence described above, we can determine exactly which is the arc removed by cycle-breaking (if $x \rightarrow y$ is the arc that will close the cycle according to the Covington arc-building order, then the affected arc is the one of the form $z \rightarrow x$). The cycle can only cause the loss of a gold arc if that arc $z \rightarrow x$ is gold, which can be trivially checked. Hence, if we call cycles where that holds **problematic cycles**, then the expression

[3] Note that, in this scenario, the new head must also be wrong because otherwise the newly created arc would be an arc of $\mathcal{I}(c, t_G)$ (and therefore, would not be breaking a cycle in $A \cup \mathcal{I}(c, t_G)$). However, replacing a wrong attachment with another wrong attachment need not increase loss.

Language	average value				relative difference to loss		
	lower	loss	pc upper	upper	lower	pc upper	upper
Arabic	0.66925	0.67257	**0.67312**	0.68143	0.00182	**0.00029**	0.00587
Basque	**0.58260**	0.58318	0.58389	0.62543	**0.00035**	0.00038	0.02732
Catalan	0.58009	0.58793	**0.58931**	0.60644	0.00424	**0.00069**	0.00961
Chinese	**0.56515**	0.56711	0.57156	0.62921	**0.00121**	0.00302	0.03984
Czech	**0.57521**	0.58357	0.59401	0.62883	**0.00476**	0.00685	0.02662
English	0.55267	0.56383	**0.56884**	0.59494	0.00633	**0.00294**	0.01767
Greek	0.56123	0.57443	**0.57983**	0.61256	0.00731	**0.00296**	0.02256
Hungarian	**0.46495**	0.46672	0.46873	0.48797	**0.00097**	0.00114	0.01165
Italian	0.62033	0.62612	**0.62767**	0.64356	0.00307	**0.00082**	0.00883
Turkish	**0.60143**	0.60215	0.60660	0.63560	**0.00060**	0.00329	0.02139
Bulgarian	0.61415	0.62257	**0.62433**	0.64497	0.00456	**0.00086**	0.01233
Danish	0.67350	0.67904	**0.68119**	0.69436	0.00291	**0.00108**	0.00916
Dutch	0.69201	0.70600	**0.71105**	0.74008	0.00709	**0.00251**	0.01862
German	**0.54581**	0.54755	0.55080	0.58182	**0.00104**	0.00208	0.02033
Japanese	**0.60515**	0.60515	**0.60515**	0.60654	**0.00000**	**0.00000**	0.00115
Portuguese	0.58880	0.60063	**0.60185**	0.61780	0.00651	**0.00067**	0.00867
Slovene	0.56155	0.56860	**0.57135**	0.60373	0.00396	**0.00153**	0.01979
Spanish	0.58247	0.59119	**0.59277**	0.61273	0.00487	**0.00089**	0.01197
Swedish	0.57543	0.58636	**0.58933**	0.61104	0.00585	**0.00153**	0.01383
Average	0.59009	0.59656	**0.59954**	0.62416	0.00355	**0.00176**	0.01513

Table 1: Average value of the different bounds and the loss, and of the relative differences from each bound to the loss, on CoNLL-XI (first block) and CoNLL-X (second block) datasets during 100,000 transitions. For each language, we show in boldface the average value and relative difference of the bound that is closer to the loss.

$|\mathcal{U}(c, t_G)| + n_{pc}(A \cup \mathcal{I}(c, t_G))$, where "pc" stands for problematic cycles, is a closer **upper bound** to the loss $\ell(c)$ and the following holds:

$$|\mathcal{U}(c, t_G)| \leq \ell(c) \leq |\mathcal{U}(c, t_G)| + n_{pc}(A \cup \mathcal{I}(c, t_G))$$

$$\leq |\mathcal{U}(c, t_G)| + n_c(A \cup \mathcal{I}(c, t_G))$$

As mentioned before, unlike the monotonic approach, a node can have a different head in A than in $\mathcal{I}(c, t_G)$ and, as a consequence, the resulting graph $A \cup \mathcal{I}(c, t_G)$ has maximum in-degree 2 rather than 1, and there can be overlapping cycles. Therefore, the computation of the non-monotonic terms $n_c(A \cup \mathcal{I}(c, t_G))$ and $n_{pc}(A \cup \mathcal{I}(c, t_G))$ requires an algorithm such as the one by Johnson (1975) to find all elementary cycles in a directed graph. This runs in $O((n + e)(c + 1))$, where n is the number of vertices, e is the number of edges and c is the number of elementary cycles in the graph. This implies that the calculation of the two non-monotonic upper bounds is less efficient than the linear loss computation in the monotonic scenario. However, a non-monotonic algorithm that uses the lower bound as loss expression is the fastest option (even faster than the monotonic approach) as the oracle does not need to compute cycles at all, speeding up the training process.

Algorithm 2 shows the non-monotonic variant of Algorithm 1, where CountRelevantCycles is a function that counts the number of cycles or problematic cycles in the given graph, depending on the upper bound implemented, and will return 0 in case we use the lower bound.

Algorithm 2 Computation of the approximate loss of a non-monotonic configuration.

```
1: function Loss(c = ⟨λ1|i, λ2, j|B, A⟩, t_G)
2:     U ← ∅                          ▷ Variable U is for U(c, t_G)
3:     for each x → y ∈ (t_G \ A) do
4:         left ← min(x, y)
5:         right ← max(x, y)
6:         if j > right ∨
7:           (j = right ∧ i < left) then
8:             U ← u ∪ {x → y}
9:     I ← t_G \ U                    ▷ Variable I is for I(c, t_G)
10:    return |U| + CountRelevantCycles(A ∪ I)
```

5 Evaluation of the Loss Bounds

To determine how close the lower bound $|\mathcal{U}(c, t_G)|$ and the upper bounds $|\mathcal{U}(c, t_G)| + n_{pc}(A \cup \mathcal{I}(c, t_G))$ and $|\mathcal{U}(c, t_G)| + n_c(A \cup \mathcal{I}(c, t_G))$ are to the actual loss in practical scenarios, we use exhaustive search to calculate the real loss of a given configuration, to then compare it with the bounds. This is feasible because the lower and upper bounds allow us to prune the search space: if an upper and a lower bound coincide for a configuration we already know the loss and need not keep searching, and if we can branch to two configurations such that the lower bound of one is greater or equal than an upper bound of the other, we can discard the former as it will never lead to smaller loss than the latter. Therefore, this ex-

Unigrams
L_0w; L_0p; L_0wp; L_0l; $L_{0h}w$; $L_{0h}p$; $L_{0h}l$; $L_{0l'}w$; $L_{0l'}p$; $L_{0l'}l$; $L_{0r'}w$; $L_{0r'}p$; $L_{0r'}l$; $L_{0h2}w$; $L_{0h2}p$; $L_{0h2}l$; $L_{0l}w$; $L_{0l}p$; $L_{0l}l$; $L_{0r}w$; $L_{0r}p$; $L_{0r}l$; L_0wd; L_0pd; L_0wv_r; L_0pv_r; L_0wv_l; L_0pv_l; L_0ws_l; L_0ps_l; L_0ws_r; L_0ps_r; L_1w; L_1p; L_1wp; R_0w; R_0p; R_0wp; $R_{0h}w$; $R_{0h}p$;$R_{0h}l$; $R_{0h2}w$; $R_{0h2}p$; $R_{0l'}w$; $R_{0l'}p$; $R_{0l'}l$; $R_{0l}w$; $R_{0l}p$; $R_{0l}l$; R_0wd; R_0pd; R_0wv_l; R_0pv_l; R_0ws_l; R_0ps_l; R_1w; R_1p; R_1wp; R_2w; R_2p; R_2wp; CLw; CLp; $CLwp$; CRw; CRp; $CRwp$;
Pairs
L_0wp+R_0wp; L_0wp+R_0w; L_0w+R_0wp; L_0wp+R_0p; L_0p+R_0wp; L_0w+R_0w; L_0p+R_0p; R_0p+R_1p; L_0w+R_0wd; L_0p+R_0pd;
Triples
R_0p+R_1p+R_2p; L_0p+R_0p+R_1p; $L_{0h}p$+L_0p+R_0p; L_0p+$L_{0l'}p$+R_0p; L_0p+$L_{0r'}p$+R_0p; L_0p+R_0p+$R_{0l'}p$; L_0p+$L_{0l'}p$+$L_{0l}p$; L_0p+$L_{0r'}p$+$L_{0r}p$; L_0p+$L_{0h}p$+$L_{0h2}p$; R_0p+$R_{0l'}p$+$R_{0l}p$;

Table 2: Feature templates. L_0 and R_0 denote the left and right focus words; $L_1, L_2, \ldots$ are the words to the left of L_0 and $R_1, R_2, \ldots$ those to the right of R_0. X_{ih} means the head of X_i, X_{ih2} the grandparent, X_{il} and $X_{il'}$ the farthest and closest left dependents, and X_{ir} and $X_{ir'}$ the farthest and closest right dependents, respectively. CL and CR are the first and last words between L_0 and R_0 whose head is not in the interval $[L_0, R_0]$. Finally, w stands for word form; p for PoS tag; l for dependency label; d is the distance between L_0 and R_0; v_l, v_r are the left/right valencies (number of left/right dependents); and s_l, s_r the left/right label sets (dependency labels of left/right dependents).

haustive search with pruning guarantees to find the exact loss.

Due to the time complexity of this process, we undertake the analysis of only the first 100,000 transitions on each dataset of the nineteen languages available from CoNLL-X and CoNLL-XI shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007). In Table 1, we present the average values for the lower bound, both upper bounds and the loss, as well as the relative differences from each bound to the real loss. After those experiments, we conclude that the lower and the closer upper bounds are a tight approximation of the loss, with both bounds incurring relative errors below 0.8% in all datasets. If we compare them, the real loss is closer to the upper bound $|\mathcal{U}(c, t_G)| + n_{pc}(A \cup \mathcal{I}(c, t_G))$ in the majority of datasets (12 out of 18 languages, excluding Japanese where both bounds were exactly equal to the real loss in the whole sample of configurations). This means that the term $n_{pc}(A \cup \mathcal{I}(c, t_G))$ provides a close approximation of the gold arcs missed by the presence of cycles in A. Regarding the upper bound $|\mathcal{U}(c, t_G)| + n_c(A \cup \mathcal{I}(c, t_G))$, it presents a more variable relative error, ranging from 0.1% to 4.0%.

Thus, although we do not know an algorithm to obtain the exact loss which is fast enough to be practical, any of the three studied loss bounds can be used to obtain a feasible approximate dynamic oracle with full non-monotonicity.

6 Experiments

To prove the usefulness of our approach, we implement the static, dynamic monotonic and non-monotonic oracles for the non-projective Covington algorithm and compare their accuracies on nine datasets[4] from the CoNLL-X shared task (Buchholz and Marsi, 2006) and all datasets from the CoNLL-XI shared task (Nivre et al., 2007). For the non-monotonic algorithm, we test the three different loss expressions defined in the previous section. We train an averaged perceptron model for 15 iterations and use the same feature templates for all languages[5] which are listed in detail in Table 2.

6.1 Results

The accuracies obtained by the non-projective Covington parser with the three available oracles are presented in Table 3, in terms of Unlabeled (UAS) and Labeled Attachment Score (LAS). For the non-monotonic dynamic oracle, three variants are shown, one for each loss expression implemented. As we can see, the novel non-monotonic oracle improves over the accuracy of the monotonic version on 14 out of 19 languages (0.32 in UAS on average) with the best loss calculation being $|\mathcal{U}(c, t_G)| + n_c(A \cup \mathcal{I}(c, t_G))$, where 6 of these improvements are statistically significant at the .05 level (Yeh, 2000). The other two loss calculation methods also achieve good results, outperforming the monotonic algorithm on 12 out of 19 datasets tested.

The loss expression $|\mathcal{U}(c, t_G)| + n_c(A \cup \mathcal{I}(c, t_G))$ obtains greater accuracy on average than the other two loss expressions, including the more adjusted upper bound that is provably closer to the real loss. This could be explained by the fact that

[4] We excluded the languages from CoNLL-X that also appeared in CoNLL-XI, i.e., if a language was present in both shared tasks, we used the latest version.

[5] No feature optimization is performed since our priority in this paper is not to compete with state-of-the-art systems, but to prove, under uniform experimental settings, that our approach outperforms the baseline system.

	static		dynamic monotonic		dynamic non-monotonic					
					lower		pc upper		upper	
Language	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Arabic	80.67	66.51	82.76*	68.48*	83.29*	69.14*	83.18*	69.05*	**83.40**†	**69.29**†
Basque	76.55	66.05	**77.49**†	**67.31**†	74.61	65.31	74.69	65.18	74.27	64.78
Catalan	90.52	85.09	**91.37***	**85.98***	90.51	85.35	90.40	85.30	90.44	85.35
Chinese	84.93	80.80	85.82	82.15	86.55*	**82.53***	86.29*	82.27*	**86.60***	82.51*
Czech	78.49	61.77	80.21*	63.52*	81.32†	64.89†	81.33†	64.81†	**81.49**†	**65.18**†
English	85.35	84.29	87.47*	86.55*	88.44†	87.37†	88.23†	87.22†	**88.50**†	**87.55**†
Greek	79.47	69.35	80.76	70.43	80.90	70.46	80.84	70.34	**81.02***	**70.49***
Hungarian	77.65	68.32	**78.84***	**70.16***	78.67*	69.83*	78.47*	69.66*	78.65*	69.74*
Italian	84.06	79.79	84.30	80.17	84.38	80.30	**84.64**	**80.52**	84.47	80.32
Turkish	**81.28**	70.97	81.14	**71.38**	80.65	71.15	80.80	71.29	80.60	71.07
Bulgarian	89.13	85.30	90.45*	86.86*	91.36†	87.88†	91.33†	87.89†	**91.73**†	**88.26**†
Danish	86.00	81.49	86.91*	**82.75***	86.83*	82.63*	86.89*	82.74*	**86.94***	82.68*
Dutch	81.54	78.46	82.07	79.26	82.78*	79.64*	82.80*	79.68*	**83.02**†	**79.92**†
German	86.97	83.91	**87.95***	**85.17***	87.31	84.37	87.18	84.22	87.48	84.54
Japanese	93.63	92.20	93.67	92.33	**94.02**	**92.68**	**94.02**	**92.68**	93.97	92.66
Portuguese	86.55	82.61	**87.45***	83.62*	87.17*	83.47*	87.12*	83.45*	87.40*	**83.71***
Slovene	76.76	63.53	77.86	64.43	80.39†	67.04†	**80.56**†	**67.10**†	80.47†	**67.10**†
Spanish	79.20	76.00	80.12*	77.24*	**81.36***	**78.30***	81.12*	77.99*	81.33*	78.16*
Swedish	87.43	81.77	88.05*	82.77*	88.20*	83.02*	88.09*	82.87*	**88.36***	**83.16***
Average	83.48	76.75	84.46	77.92	84.67	78.18	84.63	78.12	**84.74**	**78.24**

Table 3: Parsing accuracy (UAS and LAS, including punctuation) of the Covington non-projective parser with static, and dynamic monotonic and non-monotonic oracles on CoNLL-XI (first block) and CoNLL-X (second block) datasets. For the dynamic non-monotonic oracle, we show the performance with the three loss expressions, where *lower* stands for the lower bound $|\mathcal{U}(c, t_G)|$, *pc upper* for the upper bound $|\mathcal{U}(c, t_G)| + n_{pc}(A \cup \mathcal{I}(c, t_G))$, and *upper* for the upper bound $|\mathcal{U}(c, t_G)| + n_c(A \cup \mathcal{I}(c, t_G))$. For each language, we run five experiments with the same setup but different seeds and report the averaged accuracy. Best results for each language are shown in boldface. Statistically significant improvements ($\alpha = .05$) of both dynamic oracles are marked with * if they are only over the static oracle, and with † if they are over the opposite dynamic oracle too.

identifying problematic cycles is a difficult task to learn for the parser, and for this reason a more straightforward approach, which tries to avoid all kinds of cycles (regardless of whether they will cost gold arcs or not), can perform better. This also leads us to hypothesize that, even if it were feasible to build an oracle with the exact loss, it would not provide practical improvements over these approximate oracles; as it appears difficult for a statistical model to learn the situations where replacing a wrong arc with another indirectly helps due to breaking prospective cycles.

It is also worth mentioning that the non-monotonic dynamic oracle with the best loss expression accomplishes an average improvement over the static version (1.26 UAS) greater than that obtained by the monotonic oracle (0.98 UAS), resulting in 13 statistically significant improvements achieved by the non-monotonic variant over the static oracle in comparison to the 12 obtained by the monotonic system. Finally, note that, despite this remarkable performance, the non-monotonic version (regardless of the loss expression implemented) has an inexplicable drop in accuracy in Basque in comparison to the other two oracles.

6.2 Comparison

In order to provide a broader contextualization of our approach, Table 4 presents a comparison of the average accuracy and parsing speed obtained by some well-known transition-based systems with dynamic oracles. Concretely, we include in this comparison both monotonic (Goldberg and Nivre, 2012) and non-monotonic (Honnibal et al., 2013) versions of the arc-eager parser, as well as the original monotonic Covington system (Gómez-Rodríguez and Fernández-González, 2015). The three of them were ran with our own implementation so the comparison is homogeneous. We also report the published accuracy of the non-projective Attardi algorithm (Gómez-Rodríguez et al., 2014) on the nineteen datasets used in our experiments. From Table 4 we can see that our approach achieves the best average UAS score, but is slightly slower at parsing time than the monotonic Covington algorithm. This can be explained by the fact that the non-monotonic parser has to take into consideration the whole set of transitions at each configuration (since all are allowed), while the monotonic parser only needs to evaluate a limited set of transitions in some con-

Algorithm	Average value		
	UAS	LAS	sent./s.
G&N 2012	84.32	77.68	833.33
G-R et al. 2014*	83.78	**78.64**	-
G-R&F-G 2015	84.46	77.92	335.63
H et al. 2013	84.28	77.68	**847.33**
This work	**84.74**	78.24	236.74

Table 4: Comparison of the average Unlabeled and Labeled Attachment Scores (including punctuation) achieved by some widely-used transition-based algorithms with dynamic oracles on nine CoNLL-X datasets and all CoNLL-XI datatsets, as well as their average parsing speed (sentences per second across all datasets) measured on a 2.30GHz Intel Xeon processor. The first block corresponds to monotonic parsers, while the second gathers non-monotonic parsers. All algorithms are tested under our own implementation, except for the system developed by Gómez-Rodríguez et al. (2014) (marked with *) where we report the published results.

figurations, speeding up the parsing process.

6.3 Error Analysis

We also carry out some error analysis to provide some insights about how non-monotonicity is improving accuracy with respect to the original Covington parser. In particular, we notice that non-monotonicity tends to be more beneficial on projective than on non-projective arcs. In addition, the non-monotonic algorithm presents a notable performance on long arcs (which are more prone to error propagation): average precision on arcs with length greater than 7 goes from 58.41% in the monotonic version to 63.19% in the non-monotonic parser, which may mean that non-monotonicity is alleviating the effect of error propagation. Finally, we study the effectiveness of non-monotonic arcs (i.e., those that break a previously-created arc), obtaining that, on average across all datasets tested, 36.86% of the arc transitions taken were non-monotonic, replacing an existing arc with a new one. Out of these transitions, 60.31% created a gold arc, and only 5.99% were harmful (i.e., they replaced a previously-built gold arc with an incorrect arc), with the remaining cases creating non-gold arcs without introducing extra errors (replacing a non-gold arc with another). These results back up the usefulness of non-monotonicity in transition-based parsing.

7 Conclusion

We presented a novel, fully non-monotonic variant of the well-known non-projective Covington parser, trained with a dynamic oracle. Due to the unpredictability of a non-monotonic scenario, the real loss of each configuration cannot be computed. To overcome this, we proposed three different loss expressions that closely bound the loss and enable us to implement a practical non-monotonic dynamic oracle.

On average, our non-monotonic algorithm obtains better performance than the monotonic version, regardless of which of the variants of the loss calculation is used. In particular, one of the loss expressions developed proved very promising by providing the best average accuracy, in spite of being the farthest approximation from the actual loss. On the other hand, the proposed lower bound makes the non-monotonic oracle the fastest one among all dynamic oracles developed for the non-projective Covington algorithm.

To our knowledge, this is the first implementation of non-monotonicity for a non-projective parsing algorithm, and the first approximate dynamic oracle that uses close, efficiently-computable approximations of the loss, showing this to be a feasible alternative when it is not practical to compute the actual loss.

While we used a perceptron classifier for our experiments, our oracle could also be used in neural-network implementations of greedy transition-based parsing (Chen and Manning, 2014; Dyer et al., 2015), providing an interesting avenue for future work. We believe that gains from both techniques should be complementary, as they apply to orthogonal components of the parsing system (the scoring model vs. the transition system), although we might see a ”diminishing returns”effect.

Acknowledgments

This research has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 714150 - FASTPARSE). The second author has received funding from the TELEPARES-UDC project (FFI2014-51978-C2-2-R) from MINECO.

References

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*. pages 149–164. http://www.aclweb.org/anthology/W06-2920.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 740–750. http://www.aclweb.org/anthology/D14-1082.

Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*. ACM, New York, NY, USA, pages 95–102.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 334–343. http://www.aclweb.org/anthology/P15-1033.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*. Association for Computational Linguistics, Mumbai, India, pages 959–976. http://www.aclweb.org/anthology/C12-1059.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics* 1:403–414. http://anthology.aclweb.org/Q/Q13/Q13-1033.pdf.

Carlos Gómez-Rodríguez and Daniel Fernández-González. 2015. An efficient dynamic oracle for unrestricted non-projective parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*. pages 256–261. http://aclweb.org/anthology/P/P15/P15-2042.pdf.

Carlos Gómez-Rodríguez and Joakim Nivre. 2013. Divisible transition systems and multiplanar dependency parsing. *Computational Linguistics* 39(4):799–845. http://aclweb.org/anthology/J/J13/J13-4002.pdf.

Carlos Gómez-Rodríguez, Francesco Sartorio, and Giorgio Satta. 2014. A polynomial-time dynamic oracle for non-projective dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 917–927. http://aclweb.org/anthology/D14-1099.

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. A non-monotonic arc-eager transition system for dependency parsing. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*. pages 163–172. http://aclweb.org/anthology/W/W13/W13-3518.pdf.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1373–1378. http://aclweb.org/anthology/D15-1162.

Donald B. Johnson. 1975. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing* 4(1):77–84. https://doi.org/10.1137/0204007.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. pages 122–131. http://www.aclweb.org/anthology/D/D07/D07-1013.pdf.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*. ACL/SIGPARSE, pages 149–160.

Joakim Nivre. 2008. Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics* 34(4):513–553. https://doi.org/10.1162/coli.07-056-R1-07-027.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. pages 915–932. http://www.aclweb.org/anthology/D/D07/D07-1096.pdf.

Robert Endre Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM J. Comput.* 1(2):146–160. http://dblp.uni-trier.de/db/journals/siamcomp/siamcomp1.html.

Alexander Volokh. 2013. *Performance-Oriented Dependency Parsing*. Doctoral dissertation, Saarland University, Saarbrücken, Germany.

Alexander Volokh and Günter Neumann. 2012. Dependency parsing with efficient feature extraction. In Birte Glimm and Antonio Krüger, editors, *KI*. Springer, volume 7526 of *Lecture Notes in Computer Science*, pages 253–256. https://doi.org/10.1007/978-3-642-33347-7.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*. pages 947–953. http://aclweb.org/anthology/C/C00/C00-2137.pdf.

Aggregating and Predicting Sequence Labels from Crowd Annotations

An T. Nguyen[1] **Byron C. Wallace**[2] **Junyi Jessy Li**[3] **Ani Nenkova**[3] **Matthew Lease** [1]

[1]University of Texas at Austin, [2]Northeastern University,
[3]University of Pennsylvania,
atn@cs.utexas.edu, byron@ccs.neu.edu,
{ljunyi|nenkova}@seas.upenn.edu, ml@utexas.edu

Abstract

Despite sequences being core to NLP, scant work has considered how to handle noisy sequence labels from multiple annotators for the same text. Given such annotations, we consider two complementary tasks: (1) aggregating sequential crowd labels to infer a best single set of consensus annotations; and (2) using crowd annotations as training data for a model that can predict sequences in unannotated text. For aggregation, we propose a novel Hidden Markov Model variant. To predict sequences in unannotated text, we propose a neural approach using Long Short Term Memory. We evaluate a suite of methods across two different applications and text genres: Named-Entity Recognition in news articles and Information Extraction from biomedical abstracts. Results show improvement over strong baselines. Our source code and data are available online[1].

1 Introduction

Many important problems in Natural Language Processing (NLP) may be viewed as sequence labeling tasks, such as part-of-speech (PoS) tagging, named-entity recognition (NER), and Information Extraction (IE). As with other machine learning tasks, automatic sequence labeling typically requires annotated corpora on which to train predictive models. While such annotation was traditionally performed by domain experts, crowdsourcing has become a popular means to acquire large labeled datasets at lower cost, though annotations from laypeople may be lower quality than those from domain experts (Snow et al., 2008). It is therefore essential to model crowdsourced label quality, both to estimate individual annotator reliability and to aggregate individual annotations to induce a single set of "reference standard" consensus labels. While many models have been proposed for aggregating crowd labels for binary or multiclass classification problems (Sheshadri and Lease, 2013), far less work has explored crowd-based annotation of sequences (Finin et al., 2010; Hovy et al., 2014; Rodrigues et al., 2014).

In this paper, we investigate two complementary challenges in using sequential crowd labels: how to best aggregate them (Task 1); and how to accurately predict sequences in unannotated text given training data from the crowd (Task 2). For aggregation, one might want to induce a single set of high-quality consensus annotations for various purposes: (i) for direct use at run-time (when a given application requires human-level accuracy in identifying sequences); (ii) for sharing with others; or (iii) for training a predictive model.

When human-level accuracy in tagging of sequences is not crucial, automatic labeling of unannotated text is typically preferable, as it is more efficient, scalable, and cost-effective. Given a training set of crowd labels, how can we best predict sequences in unannotated text? Should we: (i) consider Task 1 as a pre-processing step and train the model using consensus labels; or (ii) instead directly train the model on all of the individual annotations, as done by Yang et al. (2010)? We investigate both directions in this work.

Our approach is to augment existing sequence labeling models such as HMMs (Rabiner and Juang, 1986) and LSTMs (Hochreiter and Schmidhuber, 1997; Lample et al., 2016) by introducing an explicit "crowd component". For HMMs, we model this crowd component by including additional parameters for worker label quality and crowd label variables. For the LSTM, we introduce a vector representation for each annotator. In

[1] Soure code and biomedical abstract data: www.github.com/thanhan/seqcrowd-acl17, www.byronwallace.com/EBM_abstracts_data

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 299–309
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1028

both cases, the crowd component models both the noise from labels and the label quality from each annotator. We find that principled combination of the "crowd component" with the "sequence component" yields strong improvement.

For evaluation, we consider two practical applications in two text genres: NER in news and IE from medical abstracts. Recognizing named-entities such as people, organizations or locations can be viewed as a sequence labeling task in which each label specifies whether each word is Inside, Outside or Beginning (IOB) a named-entity. For this task, we consider the English portion of the CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003), using crowd labels collected by Rodrigues et al. (2014).

For the IE application, we use a set of biomedical abstracts that describe Randomized Controlled Trials (RCTs). The crowdsourced annotations comprise labeled text spans that describe the patient populations enrolled in the corresponding RCTs. For example, an abstract may contain the text: *we recruited and enrolled diabetic patients*. Identifying these sequences is useful for downstream systems that process biomedical literature, e.g., clinical search engines (Huang et al., 2006; Schardt et al., 2007; Wallace et al., 2016).

Contributions. We present a systematic investigation and evaluation of alternative methods for handling and utilizing crowd labels for sequential annotation tasks. We consider both how to best aggregate sequential crowd labels (Task 1) and how to best predict sequences in unannotated text given a training set of crowd annotations (Task 2). As part of this work, we propose novel models for working with noisy sequence labels from the crowd. Reported experiments both benchmark existing state-of-the-art approaches (sequential and non-sequential) and show that our proposed models achieve best-in-class performance. As noted in the Abstract, we have also shared our sourcecode and data online for use by the community.

2 Related Work

We briefly review two separate threads of relevant prior work: (1) sequence labeling models; and (2) aggregation of crowdsourcing annotations.

Sequence labeling. Early work on learning for sequential tasks used HMMs (Bikel et al., 1997). HMMs are a class of generative probabilistic models comprising two components: an emission model from a hidden state to an observation and a transition model from a hidden state to the next hidden state. Later work focused on discriminative models such as Maximum Entropy Models (Chieu and Ng, 2002) and Conditional Random Fields (CRFs) (Lafferty et al., 2001). These were able to achieve strong predictive performance by exploiting arbitrary features, but they may not be the best choice for label aggregation. Also, compared to the simple HMM model, discriminative sequentially structured models require more complex optimization and are generally more difficult to extend. Here we argue for the generative HMMs for our first task of aggregating crowd labels. The generative nature of HMMs is a good fit for existing crowd modeling techniques and also enables very efficient parameter estimation.

In addition to the supervised setting, previous work has studied unsupervised HMMs, e.g., for PoS induction (Goldwater and Griffiths, 2007; Johnson, 2007). These works are similar to our work in trying to infer the hidden states without labeled data. Our graphical model is different in incorporating signal from the crowd labels.

For Task 2 (training predictive models), we consider CRFs and LSTMs. CRFs are undirected, conditional models that can exploit arbitrary features. They have achieved strong performance on many sequence labeling tasks (McCallum and Li, 2003), but they depend on hand-crafted features. Recent work has considered end-to-end neural architectures that *learn* features, e.g., Convolutional Neural Networks (CNNs) (Collobert et al., 2011; Kim, 2014; Zhang and Wallace, 2015) and LSTMs (Lample et al., 2016). Here we modify the LSTM model proposed by Lample et al. (2016) by augmenting the network with '*crowd worker vectors*'.

Crowdsourcing. Acquiring labeled data is critical for training supervised models. Snow et al. (2008) proposed using Amazon Mechanical Turk to collect labels in NLP quickly and at low cost, albeit with some degradation in quality. Subsequent work has developed models for improving aggregate label quality (Raykar et al., 2010; Felt et al., 2015; Kajino et al., 2012; Bi et al., 2014; Liu et al., 2012; Hovy et al., 2013). Sheshadri and Lease (2013) survey and benchmark methods.

However, these models are almost all in the binary or multiclass classification setting; only a few have considered sequence labeling. Dredze et al. (2009) proposed a method for learning a CRF

model from multiple labels (although the identities of the annotators or workers were not used). Rodrigues et al. (2014) extended this approach to account for worker identities, providing a joint "crowd-CRF" model. They collected a dataset of crowdsourced labels for a portion of the CoNLL 2003 dataset. Using this, they showed that their model outperformed Dredze et al. (2009)'s model and other baselines. However, due to the technical difficulty of the joint approach with CRFs, they resorted to strong modeling assumptions. For example, their model assumes that for each word, only one worker provides the correct answer while all others label the word completely randomly. While this assumption captures some aspects of label quality, it is potentially problematic, such as for 'easy words' labeled correctly by all workers.

More recently, **?** proposed HMM models for aggregating crowdsourced discourse segmentation labels. However, they did not consider the general sequence labeling setting. Their method includes task-specific assumptions, e.g., that discourse segment lengths follow some empirical distribution estimated from data. In the absence of a gold standard, they evaluated by checking that workers accuracies are consistent and by comparing their two models to each other. We include their approach along with Rodrigues et al. (2014) as a baseline in our evaluation.

3 Methods

We present our Task 1 HMM approach in Section 3.1 and our Task 2 LSTM approach in Section 3.2.

3.1 HMMs with Crowd Workers

Model: We first define a standard HMM with hidden states h_i, observations v_i, transition parameter vectors $\boldsymbol{\tau}_{h_i}$ and emission parameter vectors $\boldsymbol{\Omega}_{h_i}$:

$$h_{i+1}|h_i \sim \text{Discrete}(\boldsymbol{\tau}_{h_i}) \quad (1)$$

$$v_i|h_i \sim \text{Discrete}(\boldsymbol{\Omega}_{h_i}) \quad (2)$$

The discrete distributions here are governed by Multinomials. In the context of our task, v_i is the word at position i and h_i is the true, latent class of v_i (e.g., entity or non-entity).

For the crowd component, assume there are n classes, and let l_{ij} be the label for word i provided by worker j. Further, let $\mathbf{C}^{(j)}$ be the confusion matrix for worker j, i.e., $\mathbf{C}^{(j)}_k$ is a vector of size n in which element k' is the probability of worker j providing the label k' for a word of true class k:

$$l_{ij}|h_i \sim \text{Discrete}(\mathbf{C}^{(j)}_{h_i}) \quad (3)$$

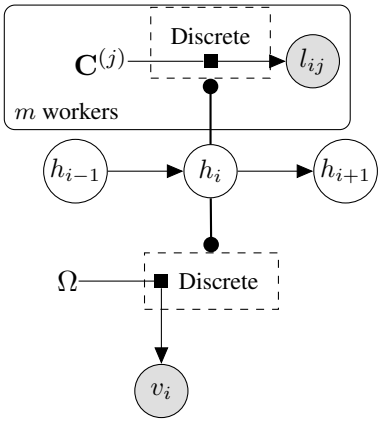


Figure 1: The factor graph for our HMM-Crowd model. Dotted rectangles are gates, where the value of h_i is used to select the parameters for the Multinomial governing the Discrete distribution.

Figure 1 shows the factor graph of this model, which we call HMM-Crowd. Note that we assume that individual crowdworker labels are conditionally independent given the (hidden) true label.

A common problem with crowdsourcing models is data sparsity. For workers who provide only a few labels, it is hard to derive a good estimate of their confusion matrices. This is exacerbated when the label distribution is imbalanced, e.g., most words are not part of a named entity, concentrating the counts in a few confusion matrix entries. Solutions for this problem include hierarchical models of 'worker communities' (Venanzi et al., 2014) or correlations between confusion matrix entries (Nguyen et al., 2016). Although effective, these methods are also quite computationally expensive. For our models, to keep parameter estimation efficient, we use a simpler solution of 'collapsing' the confusion matrix into a 'confusion vector'. For worker j, instead of having the $n \times n$ matrix $\mathbf{C}^{(j)}$, we use the $n \times 1$ vector $\mathbf{C'}^{(j)}$ where $\mathbf{C'}^{(j)}_k$ is the probability of worker j labeling a word with true class k correctly. We also smooth the estimate of $\mathbf{C'}$ with prior counts as in (Liu and Wang, 2012; Kim and Ghahramani, 2012).

Learning: We use the Expectation Maximization (EM) algorithm (Dempster et al., 1977) to learn the parameters $(\boldsymbol{\tau}, \boldsymbol{\Omega}, \mathbf{C'})$, given the observations (all the words $\mathbf{V}$ and all the worker labels $\mathbf{L}$).

In the E-step, given the current estimates of the parameters, we take a forward and a backward

pass in the HMM to infer the hidden states, i.e. to calculate $p(h_i|\mathbf{V},\mathbf{L})$ and $p(h_i, h_{i+1}|\mathbf{V},\mathbf{L})$ for all appropriate i. Let $\alpha(h_i) = p(h_i, v_{1:i}, l_{1:i})$ where $v_{1:i}$ are the words from position 1 to i and $l_{1:i}$ are the crowd labels for these words from all workers. Similarly, let $\beta(h_i) = p(v_{i+1:n}, l_{i+1:n}|h_i)$. We have the recursions:

$$\alpha(h_i) = \sum_{h_{i-1}} p(v_i|h_i)p(h_i|h_{i-1}) \prod_j p(l_{ij}|h_i)\alpha(h_{i-1}) \tag{4}$$

$$\beta(h_i) = \sum_{h_{i+1}} p(h_{i+1}|h_i)p(v_{i+1}|h_{i+1}) \prod_j p(l_{i+1,j}|h_{i+1})\beta(h_{i+1}) \tag{5}$$

These are the standard α and β recursions for HMMs augmented with the crowd model: the product $\prod_j$ over the workers j who have provided labels for word i (or $i+1$). The posteriors can then be easily evaluated: $p(h_i|\mathbf{V},\mathbf{L}) \propto \alpha(h_i)\beta(h_i)$ and $p(h_i, h_{i+1}|\mathbf{V},\mathbf{L}) \propto \alpha(h_i)p(h_{i+1}|h_i)p(v_{i+1}|h_{i+1})\beta(h_{i+1})$

In the standard M-step, the parameters are estimated using maximum likelihood. However, we found a Variational Bayesian (VB) update procedure for the HMM parameters similar to (Johnson, 2007; Beal, 2003) provides some improvement and stability. We first define the Dirichlet priors over the transition and emission parameters:

$$p(\boldsymbol{\tau}_{h_i}) = \mathrm{Dir}(a_t) \tag{6}$$

$$p(\boldsymbol{\Omega}_{h_i}) = \mathrm{Dir}(a_e) \tag{7}$$

With these priors, the variational M-step updates the parameters as follows[2]:

$$\boldsymbol{\tau}_{h'|h} = \frac{\exp\{\Psi(\mathbf{E}_{h'|h} + a_t)\}}{\exp\{\Psi(\mathbf{E}_h + na_t)\}} \tag{8}$$

$$\boldsymbol{\Omega}_{v|h} = \frac{\exp\{\Psi(\mathbf{E}_{v|h} + a_e)\}}{\exp\{\Psi(\mathbf{E}_h + ma_e)\}} \tag{9}$$

where Ψ is the Digamma function, n is the number of states and m is the number of observations. $\mathbf{E}$ denotes the expected counts, calculated from the posteriors inferred in the E-step. $\mathbf{E}_{h'|h}$ is the expected number of times the HMM transitioned from state h to state h', where the expectation is with respect to the posterior distribution $p(h_i, h_{i+1}|\mathbf{V},\mathbf{L})$ that we infer in the E step:

$$\mathbf{E}_{h'|h} = \sum_i p(h_i = h, h_{i+1} = h'|\mathbf{V},\mathbf{L}) \tag{10}$$

[2]See Beal (2003) for the derivation and Johnson (2007) for further discussion for the Variational Bayesian approach.

Similarly, $\mathbf{E}_h$ is the expected number of times the HMM is at state h: $\mathbf{E}_h = \sum_i p(h_i = h|\mathbf{V},\mathbf{L})$ and $\mathbf{E}_{v|h}$ is the expected number of times the HMM emits the observation v from the state h: $\mathbf{E}_{v|h} = \sum_{i,v_i=v} p(h_i = h|\mathbf{V},\mathbf{L})$.

For the crowd parameters $\mathbf{C}'^{(j)}$, we use the (smoothed) maximum likelihood estimate:

$$\mathbf{C}'^{(j)}_k = \frac{\mathbf{E}^{(j)}_{k|k} + a_c}{\mathbf{E}^{(j)}_k + na_c} \tag{11}$$

where a_c is the smoothing parameter and $\mathbf{E}^{(j)}_{k|k}$ is the expected number of times that worker j correctly labeled a word of true class k as k while $\mathbf{E}^{(j)}_k$ is the expected total number of words belonging to class k worker j has labeled. Again, the expectation in $\mathbf{E}$ is taken under the posterior distributions that we infer in the E step.

3.2 Long Short Term Memory with Crowds

For Task 2, we extend the LSTM architecture (Hochreiter and Schmidhuber, 1997) for NER (Lample et al., 2016) to account for noisy crowdsourced labels (this can be easily adapted to other sequence labeling tasks). In this model, the sentence input is first fed into an LSTM block (which includes character- and word-level bi-directional LSTM units). The LSTM block's output then becomes input to a (fully connected) hidden layer, which produces a vector of tags scores for each word. This *tag score* vector is the word-level prediction, representing the likelihood of the word being from each tag. All the tags scores are then fed into a 'CRF layer' that 'connects' the word-level predictions in the sentence and produces the final output: the most likely sequence of tags.

We introduce a crowd representation in which a worker vector represents the noise associated with her labels. In other words, the parameters in the original architecture learns the correct sequence labeling model while the crowd vectors add noise to its predictions to 'explain' the lower quality of the labels. We assume a perfect worker has a zero vector as her representation while an unreliable worker is represented by a large magnitude vector. At test time, we ignore the crowd component and make predictions by feeding the unlabeled sentence into the original LSTM architecture. At train time, an example consists of the labeled sentence and the ID of the worker who provided the labels. Worker IDs are mapped to vector representations and incorporated into the LSTM architecture.

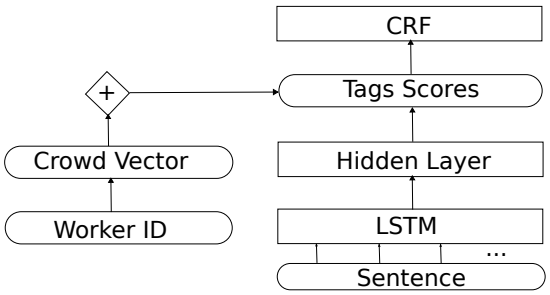


Figure 2: The LSTM-Crowd model. The Crowd Vector is added (element-wise) to the Tags Scores.

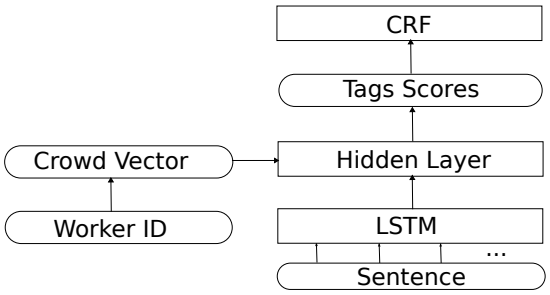


Figure 3: The LSTM-Crowd-cat model. The crowd vectors provide additional input for the Hidden Layer (they are effectively concatenated to the output of the LSTM block).

We propose two strategies for incorporating the crowd vector into the LSTM: (1) adding the crowd vector to the tags scores and (2) concatenating the crowd vector to the output of the LSTM block.

LSTM-Crowd. The first strategy is illustrated in **Figure 2**. We set the dimension of the crowd vectors to be equal to the number of tags and the addition is element-wise. In this strategy, the crowd vectors have a nice interpretation: the tag-conditional noise for the worker. This is useful for worker evaluation and intelligent task routing (i.e. assigning the right work to the right worker).

LSTM-Crowd-cat. The second strategy is illustrated in **Figure 3**. We set the crowd vectors to be additional inputs for the Hidden Layer (along with the LSTM block output). In this way, we are free to set the dimension of the crowd vectors and we have a more flexible model of worker noise. This comes with a cost of reduced interpretability and additional parameters in the hidden layer.

For both strategies, the crowd vectors are randomly initialized and learned in the same LSTM architecture using Back Propagation (Rumelhart et al., 1985) and Stochastic Gradient Descent (SGD) (Bottou, 2010).

Dataset	Application	Size	Gold	Crowd
CoNLL'03	NER	1393	All	400
Medical	IE	5000	200	All

Table 1: Datasets used for each application. We list the total number of articles/abstracts and the number which have Gold/Crowd labels.

4 Evaluation Setup

4.1 Datasets & Tuning

NER. We use the English portion of the CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003), which includes over 21,000 annotated sentences from 1,393 news articles split into 3 sets: train, validation and test. We also use crowd labels collected by Rodrigues et al. (2014) for 400 articles in the train set[3]. For Task 1 (aggregating crowd labels), to avoid overfitting, we split these 400 articles into 50% validation and 50% test[4]. For Task 2 (predicting sequences on unannotated text), we follow Rodrigues et al. (2014) in using the CoNLL validation and test sets.

Biomedical IE. We use 5,000 medical paper abstracts describing randomized control trials (RCTs) involving people. Each abstract is annotated by roughly 5 Amazon Mechanical Turk workers. Annotators were asked to mark all text spans in a given abstract which identify the population enrolled in the clinical trial. The annotations are therefore binary: inside or outside a span. In addition to annotations collected from laypeople via Mechanical Turk, we also use gold annotations by medical students for a small set of 200 abstracts, which we split into 50% validation and 50% test. For Task 1, we run methods being compared on all 5,000 abstracts, but we evaluate them only using the validation/test set. For Task 2, the validation and test sets are held out. **Table 1** presents key statistics of datasets used.

Tuning: In all experiments, validation set results are used to tune the models hyper-parameters. For HMM-Crowd, we have a smoothing parameter and two Dirichlet priors. For our two LSTMs, we have a L2 regularization parameter. For LSTM-Crowd-cat, we also have the crowd vector dimen-

[3] http://www.fprodrigues.com/software/crf-ma-sequence-labeling-with-multiple-annotators/

[4] Rodrigues et al. (2014)'s results on the 'training set' are not directly comparable to ours since they do not partition the crowd labels into validation and test sets.

sion. For each hyper-parameter, we consider a few (less then 5) different parameter settings for light tuning. We report results achieved on the test set.

4.2 Baselines

Task 1. For aggregating crowd labels, we consider the following baselines:

- Majority Voting (MV) at the token level. Rodrigues et al. (2014) show that this generally performs better than MV at the entity level.
- Dawid and Skene (1979) weighted voting at the token level. We tested both a popular public implementation[5] of Dawid-Skene and our own and found that ours performed better (likely due to smoothing), so we report it.
- MACE (Hovy et al., 2013), using the authors' public implementation[6].
- *Dawid-Skene then HMM*. We propose a simple heuristic to aggregate sequential crowd labels: (1) use Dawid and Skene (1979) to induce consensus labels from individual crowd labels; (2) train a HMM using the input text and consensus labels; and then (3) use the trained HMM to predict and output labels for the input text. We also tried using a CRF or LSTM as the sequence labeler but found the HMM performed best. This is not surprising: CRFs and LSTM are good at predicting unseen sequences, whereas the predictions here are on the seen training sequences.
- Rodrigues et al. (2014)'s CRF with Multiple Annotators (CRF-MA). We use the source code provided by the authors.
- **?**'s Interval-dependent (ID) HMM using the authors' source code[7]. Since they assume binary labels, we can only apply this to the biomedical IE task.

For non-sequential aggregation baselines, we evaluate majority voting (MV) and Dawid and Skene (1979) as perhaps the most widely known and used in practice. A recent benchmark evaluation of aggregation methods for (non-sequential) crowd labels found that classic Dawid-Skene was the most consistently strong performing method among those considered, despite its age, while majority voting was often outperformed by other methods (Sheshadri and Lease, 2013).

Dawid and Skene (1979) models a confusion matrix for each annotator, using EM estimation of these matrices as parameters and the true token labels as hidden variables. This is roughly equivalent to our proposed HMM-Crowd model (Section 3), but without the HMM component.

Task 2. To predict sequences on unannotated text when trained on crowd labels, we consider two broad approaches: (1) directly train the model on all individual crowd annotations; and (2) induce consensus labels via Task 1 and train on them.

For approach (1), we report as baselines:

- Rodrigues et al. (2014)'s CRF-MA
- Lample et al. (2016)'s LSTM trained on all crowd labels (ignoring worker IDs)

For approach (2), we report as baselines:

- Majority Voting (MV) then Conditional Random Field (CRF). We train the CRF using the *CRF Suite* package (Okazaki, 2007) with the same features as in Rodrigues et al. (2014), who also report this baseline.
- Lample et al. (2016)'s LSTM trained on Dawid-Skene (DS) consensus labels.

4.3 Metrics

NER. We use the CoNLL 2003 metrics of entity-level precision, recall and F1. The predicted entity must match the gold entity exactly (i.e. no partial credit is given for partial matches).

Biomedical IE. The above metrics are overly strict for the biomedical IR task, in which annotated sequences are typically far longer than named-entities. We therefore 'relax' the metric to credit partial matches as follows. For each predicted positive contiguous text span, we calculate:

$$\text{Precision} = \frac{\text{\# true positive words}}{\text{\# words in this predicted span}}$$

For example, for a predicted span of 10 words, if 6 words are truly positive, the Precision is 60%. We evaluate this 'local' precision for each predicted span and then take the average as the 'global' precision. Similarly, for each gold span, we calculate:

$$\text{Recall} = \frac{\text{\# words in a predicted span}}{\text{\# words in this gold span}}$$

[5] https://github.com/ipeirotis/Get-Another-Label
[6] http://www.isi.edu/publications/licensed-sw/mace/
[7] https://academiccommons.columbia.edu/catalog/ac:199939

Method	Precision	Recall	F1
Majority Vote	78.35	56.57	65.71
MACE	65.10	69.81	67.37
Dawid-Skene (DS)	78.05	65.78	71.39
CRF-MA	**80.29**	51.20	62.53
DS then HMM	76.81	71.41	74.01
HMM-Crowd	77.40	**72.29**	**74.76**

Table 2: **NER** results for Task 1 (crowd label aggregation). Rows 1-3 show non-sequential methods while Rows 4-6 show sequential methods.

The recall scores for all the gold spans are again averaged to get a global recall score.

For the biomedical IE task, because we have gold labels for only a small set of 200 abstracts, we create 100 bootstrap re-samples of the (predicted and gold) spans and perform the evaluation for each re-sample. We then report the mean and standard deviation over these 100 re-samples.

5 Evaluation Results

5.1 Named-Entity Recognition (NER)

Table 2 presents Task 1 results for aggregating crowd labels. For the non-sequential aggregation baselines, we see that Dawid and Skene (1979) outperforms both majority voting and MACE (Hovy et al., 2013). For sequential methods, our heuristic 'Dawid-Skene then HMM' method performs surprisingly well, nearly as well as HMM-Crowd. However, we will see that this heuristic does not work as well for biomedical IR.

Rodrigues et al. (2014)'s CRF-MA achieves the highest Precision of all methods, but surprisingly the lowest F1. We use their public implementation but observe different results from what they report (we observed similar results when using all the crowd data without validation/test split as they do). We suspect their released source code may be optimized for Task 2, though we could not reach the authors to verify this.

Table 3 reports NER results for Task 2: predicting sequences on unannotated text when trained on crowd labels. Results for Rodrigues et al. (2014)'s CRF-MA are reproduced using their public implementation and match their reported results. While CRF-MA outperforms 'Majority Vote then CRF' as the authors reported, and achieves the highest Recall of all methods, its F1 results are generally not competitive with other methods.

Methods based on Lample et al. (2016)'s LSTM generally outperform the CRF methods. Adding a crowd component to the LSTM yields marked improvement of 2.5-3 points F1 vs. the LSTM trained on individual crowd annotations or consensus MV annotations. LSTM-Crowd (trained on individual labels) and 'HMM-Crowd then LSTM' (LSTM trained on HMM consensus labels) offer different paths to achieving comparable, best results.

5.2 Biomedical Information Extraction (IE)

Tables 4 and **5** present Biomedical IE results for Tasks 1 and 2, respectively. We were unable to run Rodrigues et al. (2014)'s CRF-MA public implementation on the Biomedical IE dataset (due to an 'Out of Memory Error' with 8gb max heapsize).

For Task 1, Majority Vote achieves nearly 92% Precision but suffers from very low Recall. As with NER, HMM-Crowd achieves the highest Recall and F1, showing 2 points F1 improvement here over non-sequential Dawid and Skene (1979). In contrast with the NER results, our heuristic 'Dawid-Skene then HMM' performs much worse for Biomedical IE. In general, we expect heuristics to be less robust than principled methods.

For Task 2, as with NER, we again see that LSTM-Crowd (trained on individual labels) and 'HMM-Crowd then LSTM' (LSTM trained on HMM consensus labels) offer different paths to achieving fairly comparable results. While LSTM-Crowd-cat again achieves slightly lower F1, simply training Lample et al. (2016)'s LSTM directly on all crowd labels performs much better than seen earlier with NER, likely due to the relatively larger size of this dataset (see Table 1). To further investigate, we study the performances of these LSTM models as a function of training data available. In **Figure 4**, we see that as the amount of training data decreases, our crowd-augmented LSTM models produce greater relative improvement compared to the original LSTM architecture.

Table 6 presents an example from Task 1 of a sentence with its gold span, annotations and the outputs from Dawid-Skene and HMM-Crowd. Dawid-Skene aggregates labels based only on the crowd labels while our HMM-Crowd combines that with a sequence model. HMM-Crowd is able to return the longer part of the correct span.

Method	Precision	Recall	F1
CRF-MA (Rodrigues et al., 2014)	49.40	**85.60**	62.60
LSTM (Lample et al., 2016)	**83.19**	57.12	67.73
LSTM-Crowd	82.38	62.10	70.82
LSTM-Crowd-cat	79.61	62.87	70.26
Majority Vote then CRF	45.50	80.90	58.20
Dawid-Skene then LSTM	72.30	61.17	66.27
HMM-Crowd then CRF	77.40	61.40	68.50
HMM-Crowd then LSTM	76.19	66.24	**70.87**
LSTM on Gold Labels (upper-bound)	85.27	83.19	84.22

Table 3: **NER** results on Task 2: predicting sequences on unannotated text when trained on crowd labels. Rows 1-4 train the predictive model using individual crowd labels, while Rows 5-8 first aggregate crowd labels then train the model on the induced consensus labels. The last row indicates an upper-bound from training on gold labels. LSTM-Crowd and LSTM-Crowd-cat are described in Section 3.

Method	Precision	Recall	F1	std
Majority Vote	**91.89**	48.03	63.03	2.6
MACE	45.01	88.49	59.63	1.7
Dawid-Skene	77.85	66.77	71.84	1.7
Dawid-Skene then HMM	72.49	58.77	64.86	2.0
ID HMM (**?**)	78.99	68.10	73.11	1.9
HMM-Crowd	72.81	**75.14**	**73.93**	1.8

Table 4: **Biomedical IE** results for Task 1: aggregating sequential crowd labels to induce consensus labels. Rows 1-3 indicate non-sequential baselines. Results are averaged over 100 bootstrap re-samples. We report the standard deviation of F1, *std*, due to this dataset having fewer gold labels for evaluation.

Method	Precision	Recall	F1	std
LSTM (Lample et al., 2016)	77.43	61.13	68.27	1.9
LSTM-Crowd	73.83	63.93	68.47	1.6
LSTM-Crowd-cat	68.08	**68.41**	68.20	1.8
Majority Vote then CRF	**93.71**	33.16	48.92	2.8
Dawid-Skene then LSTM	70.21	65.26	67.59	1.7
HMM-Crowd then CRF	79.54	54.76	64.81	2.0
HMM-Crowd then LSTM	73.65	64.64	**68.81**	1.9

Table 5: **Biomedical IE** results for Task 2. Rows 1-3 correspond to training on all labels, while Rows 4-7 first aggregate crowd labels then train the sequence labeling model on consensus annotations.

Gold	... was as safe and effective as ... for the empiric treatment of acute invasive diarrhea in ambulatory pediatric patients requiring an emergency room visit
Annotations (2 out of 5)	... was as safe and effective as ... for the empiric treatment of acute invasive diarrhea in ambulatory pediatric patients requiring an emergency room visit
Dawid-Skene	... was as safe and effective as ... for the empiric treatment of acute invasive diarrhea in ambulatory pediatric patients requiring an emergency room visit
HMM-Crowd	... was as safe and effective as ... for the empiric treatment of acute invasive diarrhea in ambulatory pediatric patients requiring an emergency room visit

Table 6: An example from the medical abstract dataset for task 1: inferring true labels. Out of 5 annotations, only 2 have identified a positive span (the other 3 are empty). Dawid-Skene is able to assign higher weights to the minority of 2 annotations to return a part of the correct span. HMM-Crowd returns a longer part of the span, which we believe is due to useful signal from its sequence model.

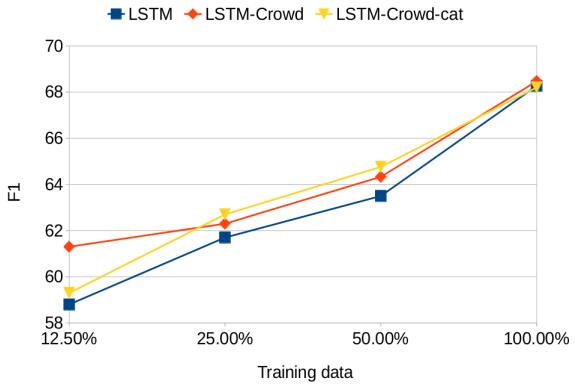


Figure 4: F1 scores in Task 2 for biomedical IE with varying percentages of training data.

6 Conclusions and Future Work

Given a dataset of crowdsourced sequence labels, we presented novel methods to: (1) aggregate sequential crowd labels to infer a best single set of consensus annotations; and (2) use crowd annotations as training data for a model that can predict sequences in unannotated text. We evaluated our approaches on two datasets representing different domains and tasks: general NER and biomedical IE. Results showed that our methods show improvement over strong baselines.

We expect our methods to be applicable to and similarly benefit other sequence labeling tasks, such as POS tagging and chunking (Hovy et al., 2014). Our methods also provide an estimate of each worker's label quality, which can be transfered between tasks and is useful for error analysis and intelligent task routing (Bragg et al., 2014). We also plan to investigate extension of the crowd component in our HMM method with hierarchical models, as well as a fully-Bayesian approach.

Acknowledgements

We thank the reviewers for their valuable comments. This work is supported in part by by National Science Foundation grant No. 1253413 and the National Cancer Institute (NCI) of the National Institutes of Health (NIH), award number UH2CA203711. Any opinions, findings, and conclusions or recommendations expressed by the authors are entirely their own and do not represent those of the sponsoring agencies.

References

Matthew James Beal. 2003. *Variational algorithms for approximate Bayesian inference*. University of London United Kingdom.

Wei Bi, Liwei Wang, James T. Kwok, and Zhuowen Tu. 2014. Learning to predict from crowdsourced data. In *Uncertainty in Artificial Intelligence*.

Daniel M Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of the fifth conference on Applied natural language processing*. Association for Computational Linguistics, pages 194–201. https://doi.org/10.3115/974557.974586.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, Springer, pages 177–186.

Jonathan Bragg, Andrey Kolobov, Mausam Mausam, and Daniel S Weld. 2014. Parallel task routing for crowdsourcing. In *Second AAAI Conference on Human Computation and Crowdsourcing*.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguis-

tics, pages 1–7. http://aclweb.org/anthology/C02-1025.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics* pages 20–28.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)* pages 1–38.

Mark Dredze, Partha Pratim Talukdar, and Koby Crammer. 2009. Sequence learning from data with multiple labels. In *ECML-PKDD 2009 workshop on Learning from Multi- Label Data*.

Paul Felt, Eric Ringger, Kevin Seppi, and Robbie Haertel. 2015. Early gains matter: A case for preferring generative over discriminative crowdsourcing models. In *Conference of the North American Chapter of the Association for Computational Linguistics*. https://doi.org/10.3115/v1/N15-1089.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, pages 80–88.

Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Annual meeting-association for computational linguistics*. Citeseer, volume 45, page 744. http://aclweb.org/anthology/P07-1094.

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

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1120–1130. http://aclweb.org/anthology/N13-1132.

Dirk Hovy, Barbara Plank, and Anders Søgaard. 2014. Experiments with crowdsourced re-annotation of a pos tagging data set. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 377–382. https://doi.org/10.3115/v1/P14-2062.

Xiaoli Huang, Jimmy Lin, and Dina Demner-Fushman. 2006. PICO as a Knowledge Representation for Clinical Questions. In *AMIA 2006 Symposium Proceedings*. pages 359–363.

Ziheng Huang, Jialu Zhong, and Rebecca J. Passonneau. 2015. Estimation of discourse segmentation labels from crowd data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2190–2200. http://aclweb.org/anthology/D15-1261.

Mark Johnson. 2007. Why doesn't em find good hmm pos-taggers? In *EMNLP-CoNLL*. pages 296–305. http://aclweb.org/anthology/D07-1031.

Hiroshi Kajino, Yuta Tsuboi, and Hisashi Kashima. 2012. A convex formulation for learning from crowds. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *International conference on artificial intelligence and statistics*. pages 619–627.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751. http://www.aclweb.org/anthology/D14-1181.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*. volume 1, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 260–270. https://doi.org/10.18653/v1/N16-1030.

Chao Liu and Yi-min Wang. 2012. Truelabel+ confusions: A spectrum of probabilistic models in analyzing multiple ratings. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. pages 225–232.

Qiang Liu, Jian Peng, and Alex T Ihler. 2012. Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems*. pages 692–700.

Andrew McCallum and Wei Li. 2003. *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*,

chapter Early results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. http://aclweb.org/anthology/W03-0430.

An T Nguyen, Byron C Wallace, and Matthew Lease. 2016. A correlated worker model for grouped, imbalanced and multitask data. In *Uncertainty in Artificial Intelligence*.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs). http://www.chokkan.org/software/crfsuite/.

Lawrence Rabiner and B Juang. 1986. An introduction to hidden markov models. *ieee assp magazine* 3(1):4–16.

Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research* 11(Apr):1297–1322.

Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. 2014. Sequence labeling with multiple annotators. *Machine learning* 95(2):165–181.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, DTIC Document.

Connie Schardt, Martha B Adams, Thomas Owens, Sheri Keitz, and Paul Fontelo. 2007. Utilization of the PICO framework to improve searching PubMed for clinical questions. *BMC medical informatics and decision making* 7(1):16.

Aashish Sheshadri and Matthew Lease. 2013. Square: A benchmark for research on computing crowd consensus. In *First AAAI Conference on Human Computation and Crowdsourcing*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 254–263. http://aclweb.org/anthology/D08-1027.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. *Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition*, pages 142–147. http://aclweb.org/anthology/W03-0419.

Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. 2014. Community-based bayesian aggregation models for crowdsourcing. In *Proceedings of the 23rd international conference on World wide web*. ACM, pages 155–164.

Byron C Wallace, Joël Kuiper, Aakash Sharma, Mingxi Brian Zhu, and Iain J Marshall. 2016. Extracting pico sentences from clinical trial reports using supervised distant supervision. *Journal of Machine Learning Research* 17(132):1–25.

Hui Yang, Anton Mityagin, Krysta M Svore, and Sergey Markov. 2010. Collecting high quality overlapping labels at low cost. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 459–466.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820* .

Multi-space Variational Encoder-Decoders for Semi-supervised Labeled Sequence Transduction

Chunting Zhou, Graham Neubig
Language Technologies Institute
Carnegie Mellon University
ctzhou,gneubig@cs.cmu.edu

Abstract

Labeled sequence transduction is a task of transforming one sequence into another sequence that satisfies desiderata specified by a set of labels. In this paper we propose *multi-space variational encoder-decoders*, a new model for labeled sequence transduction with semi-supervised learning. The generative model can use neural networks to handle both discrete and continuous latent variables to exploit various features of data. Experiments show that our model provides not only a powerful supervised framework but also can effectively take advantage of the unlabeled data. On the SIGMORPHON morphological inflection benchmark, our model outperforms single-model state-of-art results by a large margin for the majority of languages.[1]

1 Introduction

This paper proposes a model for labeled sequence transduction tasks, tasks where we are given an input sequence and a set of labels, from which we are expected to generate an output sequence that reflects the content of the input sequence and desiderata specified by the labels. Several examples of these tasks exist in prior work: using labels to moderate politeness in machine translation results (Sennrich et al., 2016), modifying the output language of a machine translation system (Johnson et al., 2016), or controlling the length of a summary in summarization (Kikuchi et al., 2016). In particular, however, we are motivated by the task of morphological reinflection (Cotterell et al., 2016), which we will use as an example in our description and test bed for our models.

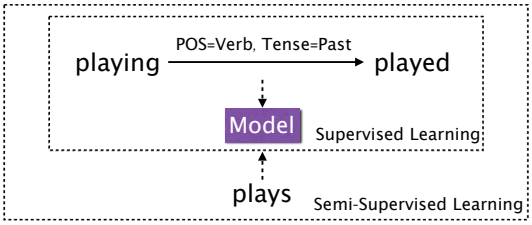


Figure 1: Standard supervised labeled sequence transduction, and our proposed semi-supervised method.

In morphologically rich languages, different affixes (i.e. prefixes, infixes, suffixes) can be combined with the lemma to reflect various syntactic and semantic features of a word. The ability to accurately analyze and generate morphological forms is crucial to creating applications such as machine translation (Chahuneau et al., 2013; Toutanova et al., 2008) or information retrieval (Darwish and Oard, 2007) in these languages. As shown in 1, *re*-inflection of an inflected form given the target linguistic labels is a challenging subtask of handling morphology as a whole, in which we take as input an inflected form (in the example, "playing") and labels representing the desired form ("`pos=Verb, tense=Past`") and must generate the desired form ("played").

Approaches to this task include those utilizing hand-crafted linguistic rules and heuristics (Taji et al., 2016), as well as learning-based approaches using alignment and extracted transduction rules (Durrett and DeNero, 2013; Alegria and Etxeberria, 2016; Nicolai et al., 2016). There have also been methods proposed using neural sequence-to-sequence models (Faruqui et al., 2016; Kann et al., 2016; Ostling, 2016), and currently ensembles of attentional encoder-decoder models (Kann and Schütze, 2016a,b) have achieved state-of-art results on this task. One feature of these neural models however, is that they are trained in a

[1] An implementation of our model are available at https://github.com/violet-zct/MSVED-morph-reinflection.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 310–320
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1029

largely *supervised* fashion (top of Fig. 1), using data explicitly labeled with the input sequence and labels, along with the output representation. Needless to say, the ability to obtain this annotated data for many languages is limited. However, we can expect that for most languages we can obtain large amounts of unlabeled surface forms that may allow for *semi-supervised* learning over this unlabeled data (entirety of Fig. 1).[2]

In this work, we propose a new framework for labeled sequence transduction problems: multi-space variational encoder-decoders (**MSVED**, §3.3). MSVEDs employ continuous or discrete latent variables belonging to multiple separate probability distributions[3] to explain the observed data. In the example of morphological reinflection, we introduce a vector of continuous random variables that represent the lemma of the source and target words, and also one discrete random variable for each of the labels, which are on the source or the target side.

This model has the advantage of both providing a powerful modeling framework for supervised learning, and allowing for learning in an unsupervised setting. For labeled data, we maximize the variational lower bound on the marginal log likelihood of the data and annotated labels. For unlabeled data, we train an auto-encoder to reconstruct a word conditioned on its lemma and morphological labels. While these labels are unavailable, a set of discrete latent variables are associated with each unlabeled word. Afterwards we can perform posterior inference on these latent variables and maximize the variational lower bound on the marginal log likelihood of data.

Experiments on the SIGMORPHON morphological reinflection task (Cotterell et al., 2016) find that our model beats the state-of-the-art for a single model in the majority of languages, and is particularly effective in languages with more complicated inflectional phenomena. Further, we find that semi-supervised learning allows for significant further gains. Finally, qualitative evaluation of lemma representations finds that our model is able to learn lemma embeddings that match with human intuition.

[2]Faruqui et al. (2016) have attempted a limited form of semi-supervised learning by re-ranking with a standard n-gram language model, but this is not integrated with the learning process for the neural model and gains are limited.

[3]Analogous to multi-space hidden Markov models (Tokuda et al., 2002)

2 Labeled Sequence Transduction

In this section, we first present some notations regarding labeled sequence transduction problems in general, then describe a particular instantiation for morphological reinflection.

Notation: Labeled sequence transduction problems involve transforming a source sequence $\mathbf{x}^{(s)}$ into a target sequence $\mathbf{x}^{(t)}$, with some labels describing the particular variety of transformation to be performed. We use discrete variables $y_1^{(t)}, y_2^{(t)}, \cdots, y_K^{(t)}$ to denote the labels associated with each target sequence, where K is the total number of labels. Let $\mathbf{y}^{(t)} = [y_1^{(t)}, y_2^{(t)}, \cdots, y_K^{(t)}]$ denote a vector of these discrete variables. Each discrete variable $y_k^{(t)}$ represents a categorical feature pertaining to the target sequence, and has a set of possible labels. In the later sections, we also use $\mathbf{y}^{(t)}$ and $y_k^{(t)}$ to denote discrete latent variables corresponding to these labels.

Given a source sequence $\mathbf{x}^{(s)}$ and a set of associated target labels $\mathbf{y}^{(t)}$, our goal is to generate a target sequence $\mathbf{x}^{(t)}$ that exhibits the features specified by $\mathbf{y}^{(t)}$ using a probabilistic model $p(\mathbf{x}^{(t)}|\mathbf{x}^{(s)}, \mathbf{y}^{(t)})$. The best target sequence $\hat{\mathbf{x}}^{(t)}$ is then given by:

$$\hat{\mathbf{x}}^{(t)} = \arg\max_{\mathbf{x}^{(t)}} p(\mathbf{x}^{(t)}|\mathbf{x}^{(s)}, \mathbf{y}^{(t)}). \quad (1)$$

Morphological Reinflection Problem: In morphological reinflection, the source sequence $\mathbf{x}^{(s)}$ consists of the characters in an inflected word (e.g., "`played`"), while the associated labels $\mathbf{y}^{(t)}$ describe some linguistic features (e.g., $y_{\text{pos}}^{(t)} =$ `Verb`, $y_{\text{tense}}^{(t)} =$ `Past`) that we hope to realize in the target. The target sequence $\mathbf{x}^{(t)}$ is therefore the characters of the re-inflected form of the source word (e.g., "`played`") that satisfy the linguistic features specified by $\mathbf{y}^{(t)}$. For this task, each discrete variable $y_k^{(t)}$ has a set of possible labels (e.g. `pos=V`, `pos=ADJ`, etc) and follows a multinomial distribution.

3 Proposed Method

3.1 Preliminaries: Variational Autoencoder

As mentioned above, our proposed model uses probabilistic latent variables in a model based on neural networks. The variational autoencoder (Kingma and Welling, 2014) is an efficient way to handle (continuous) latent variables in neural

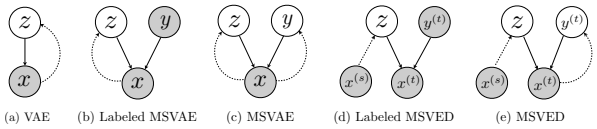


Figure 2: Graphical models of (a) VAE, (b) labeled MSVAE, (c) MSVAE, (d) labeled MSVED, and (e) MSVED. White circles are latent variables and shaded circles are observed variables. Dashed lines indicate the inference process while the solid lines indicate the generative process.

models. We describe it briefly here, and interested readers can refer to Doersch (2016) for details. The VAE learns a generative model of the probability $p(\mathbf{x}|\mathbf{z})$ of observed data $\mathbf{x}$ given a latent variable $\mathbf{z}$, and simultaneously uses a recognition model $q(\mathbf{z}|\mathbf{x})$ at learning time to estimate $\mathbf{z}$ for a particular observation $\mathbf{x}$ (Fig. 2(a)). $q(\cdot)$ and $p(\cdot)$ are modeled using neural networks parameterized by ϕ and θ respectively, and these parameters are learned by maximizing the variational lower bound on the marginal log likelihood of data:

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \quad (2)$$

The KL-divergence term (a standard feature of variational methods) ensures that the distributions estimated by the recognition model $q_\phi(\mathbf{z}|\mathbf{x})$ do not deviate far from our prior probability $p(\mathbf{z})$ of the values of the latent variables. To optimize the parameters with gradient descent, Kingma and Welling (2014) introduce a reparameterization trick that allows for training using simple backpropagation w.r.t. the Gaussian latent variables $\mathbf{z}$. Specifically, we can express $\mathbf{z}$ as a deterministic variable $\mathbf{z} = g_\phi(\epsilon, \mathbf{x})$ where ϵ is an independent Gaussian noise variable $\epsilon \sim \mathcal{N}(0, 1)$. The mean μ and the variance σ^2 of $\mathbf{z}$ are reparameterized by the differentiable functions w.r.t. ϕ. Thus, instead of generating $\mathbf{z}$ from $q_\phi(\mathbf{z}|\mathbf{x})$, we sample the auxiliary variable ϵ and obtain $\mathbf{z} = \mu_\phi(x) + \sigma_\phi(x) \circ \epsilon$, which enables gradients to backpropagate through ϕ.

3.2 Multi-space Variational Autoencoders

As an intermediate step to our full model, we next describe a generative model for a single sequence with both continuous and discrete latent variables, the multi-space variational auto-encoder (MSVAE). MSVAEs are a combination of two threads of previous work: deep generative models with both continuous/discrete latent variables for classification problems (Kingma et al., 2014; Maaløe et al., 2016) and VAEs with only continuous variables for sequential data (Bowman et al., 2016; Chung et al., 2015; Zhang et al., 2016; Fabius and van Amersfoort, 2014; Bayer and Osendorfer, 2014). In MSVAEs, we have an observed sequence $\mathbf{x}$, continuous latent variables $\mathbf{z}$ like the VAE, as well as discrete variables $\mathbf{y}$.

In the case of the morphology example, $\mathbf{x}$ can be interpreted as an inflected word to be generated. $\mathbf{y}$ is a vector representing its linguistic labels, either annotated by an annotator in the observed case, or unannotated in the unobserved case. $\mathbf{z}$ is a vector of latent continuous variables, e.g. a latent embedding of the lemma that captures all the information about $\mathbf{x}$ that is not already represented in labels $\mathbf{y}$.

MSVAE: Because inflected words can be naturally thought of as "lemma+morphological labels", to interpret a word, we resort to discrete and continuous latent variables that represent the linguistic labels and the lemma respectively. In this case when the labels of the sequence $\mathbf{y}$ is not observed, we perform inference over possible linguistic labels and these inferred labels are referenced in generating $\mathbf{x}$.

The generative model $p_\theta(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{z})p_\pi(\mathbf{y})p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})$ is defined as:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \quad (3)$$

$$p_\pi(\mathbf{y}) = \prod_k \mathrm{Cat}(y_k|\pi_k) \quad (4)$$

$$p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z}) = f(\mathbf{x}; \mathbf{y}, \mathbf{z}, \theta). \quad (5)$$

Like the standard VAE, we assume the prior of the latent variable $\mathbf{z}$ is a diagonal Gaussian distribution with zero mean and unit variance. We assume that each variable in $\mathbf{y}$ is independent, resulting in a factorized distribution in Eq. 4, where $\mathrm{Cat}(y_k|\pi_k)$ is a multinomial distribution with parameters π_k. For the purposes of this study, we set these to a uniform distribution $\pi_{k,j} = \frac{1}{|\pi_k|}$. $f(\mathbf{x}; \mathbf{y}, \mathbf{z}, \theta)$ calculates the likelihood of $\mathbf{x}$, a function parametrized by deep neural networks. Specifically, we employ an RNN decoder to generate the target word conditioned on the lemma variable $\mathbf{z}$ and linguistic labels $\mathbf{y}$, detailed in §5.

When inferring the latent variables from the given data $\mathbf{x}$, we assume the joint distribution of latent variables $\mathbf{z}$ and $\mathbf{y}$ has a factorized form, i.e. $q(\mathbf{z}, \mathbf{y}|\mathbf{x}) = q(\mathbf{z}|\mathbf{x})q(\mathbf{y}|\mathbf{x})$ as shown in Fig. 2(c).

The inference model is defined as follows:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}), \text{diag}(\sigma^2_\phi(\mathbf{x}))) \quad (6)$$

$$q_\phi(\mathbf{y}|\mathbf{x}) = \prod_k q_\phi(y_k|\mathbf{x}) = \prod_k \text{Cat}(y_k|\pi_\phi(\mathbf{x})) \quad (7)$$

where the inference distribution over $\mathbf{z}$ is a diagonal Gaussian distribution with mean and variance parameterized by neural networks. The inference model $q(\mathbf{y}|\mathbf{x})$ on labels $\mathbf{y}$ has the form of a discriminative classifier that generates a set of multinomial probability vectors $\pi_\phi(\mathbf{x})$ over all labels for each tag y_k. We represent each multinomial distribution $q(y_k|\mathbf{x})$ with an MLP.

The MSVAE is trained by maximizing the following variational lower bound $\mathcal{U}(\mathbf{x})$ on the objective for unlabeled data:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &\geq \mathbb{E}_{(\mathbf{y},\mathbf{z})\sim q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(\mathbf{x},\mathbf{y},\mathbf{z})}{q_\phi(\mathbf{y},\mathbf{z}|\mathbf{x})} \\ &= \mathbb{E}_{\mathbf{y}\sim q_\phi(\mathbf{y}|\mathbf{x})}[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{y})] \\ &- \text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) + \log p_\pi(\mathbf{y}) \\ &- \log q_\phi(\mathbf{y}|\mathbf{x})] = \mathcal{U}(\mathbf{x}) \quad (8)\end{aligned}$$

Note that this introduction of discrete variables requires more sophisticated optimization algorithms, which we will discuss in §4.1.

Labeled MSVAE: When $\mathbf{y}$ is observed as shown in Fig. 2(b), we maximize the following variational lower bound on the marginal log likelihood of the data and the labels:

$$\begin{aligned}\log p_\theta(\mathbf{x},\mathbf{y}) &\geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{p_\theta(\mathbf{x},\mathbf{y},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} = \\ &\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) + \log p_\pi(\mathbf{y})] \\ &- \text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \quad (9)\end{aligned}$$

which is a simple extension to Eq. 2.

Note that when labels are not observed, the inference model $q_\phi(\mathbf{y}|\mathbf{x})$ has the form of a discriminative classifier, thus we can use observed labels as the supervision signal to learn a better classifier. In this case we also minimize the following cross entropy as the classification loss:

$$\mathcal{D}(\mathbf{x},\mathbf{y}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_l(\mathbf{x},\mathbf{y})}[-\log q_\phi(\mathbf{y}|\mathbf{x})] \quad (10)$$

where $p_l(\mathbf{x},\mathbf{y})$ is the distribution of labeled data. This is a form of multi-task learning, as this additional loss also informs the learning of our representations.

3.3 Multi-space Variational Encoder-Decoders

Finally, we discuss the full proposed method: the multi-space variational encoder-decoder (MSVED), which generates the target $\mathbf{x}^{(t)}$ from the source $\mathbf{x}^{(s)}$ and labels $\mathbf{y}^{(t)}$. Again, we discuss two cases of this model: labels of the target sequence are observed and not observed.

MSVED: The graphical model for the MSVED is given in Fig. 2 (e). Because the labels of target sequence are not observed, once again we treat them as discrete latent variables and make inference on the these labels conditioned on the target sequence. The generative process for the MSVED is very similar to that of the MSVAE with one important exception: while the standard MSVAE conditions the recognition model $q(\mathbf{z}|\mathbf{x})$ on $\mathbf{x}$, then generates $\mathbf{x}$ itself, the MSVED conditions the recognition model $q(\mathbf{z}|\mathbf{x}^{(s)})$ on the source $\mathbf{x}^{(s)}$, then generates the target $\mathbf{x}^{(t)}$. Because only the recognition model is changed, the generative equations for $p_\theta(\mathbf{x}^{(t)},\mathbf{y}^{(t)},\mathbf{z})$ are exactly the same as Eqs. 3–5 with $\mathbf{x}^{(t)}$ swapped for $\mathbf{x}$ and $\mathbf{y}^{(t)}$ swapped for $\mathbf{y}$. The variational lower bound on the conditional log likelihood, however, is affected by the recognition model, and thus is computed as:

$$\begin{aligned}&\log p_\theta(\mathbf{x}^{(t)}|\mathbf{x}^{(s)}) \\ &\geq \mathbb{E}_{(\mathbf{y}^{(t)},\mathbf{z})\sim q_\phi(\mathbf{y}^{(t)},\mathbf{z}|\mathbf{x}^{(s)},\mathbf{x}^{(t)})} \log \frac{p_\theta(\mathbf{x}^{(t)},\mathbf{y}^{(t)},\mathbf{z}|\mathbf{x}^{(s)})}{q_\phi(\mathbf{y}^{(t)},\mathbf{z}|\mathbf{x}^{(s)},\mathbf{x}^{(t)})} \\ &= \mathbb{E}_{\mathbf{y}^{(t)}\sim q_\phi(\mathbf{y}^{(t)}|\mathbf{x}^{(t)})}[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(s)})}[\log p_\theta(\mathbf{x}^{(t)}|\mathbf{y}^{(t)},\mathbf{z})] \\ &- \text{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(s)})||p(\mathbf{z})) + \log p_\pi(\mathbf{y}^{(t)}) \\ &- \log q_\phi(\mathbf{y}^{(t)}|\mathbf{x}^{(t)})] = \mathcal{L}_u(\mathbf{x}^{(t)}|\mathbf{x}^{(s)}) \quad (11)\end{aligned}$$

Labeled MSVED: When the complete form of $\mathbf{x}^{(s)}$, $\mathbf{y}^{(t)}$, and $\mathbf{x}^{(t)}$ is observed in our training data, the graphical model of the labeled MSVED model is illustrated in Fig. 2 (d). We maximize the variational lower bound on the conditional log likelihood of observing $\mathbf{x}^{(t)}$ and $\mathbf{y}^{(t)}$ as follows:

$$\begin{aligned}&\log p_\theta(\mathbf{x}^{(t)},\mathbf{y}^{(t)}|\mathbf{x}^{(s)}) \\ &\geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(s)})} \log \frac{p_\theta(\mathbf{x}^{(t)},\mathbf{y}^{(t)},\mathbf{z}|\mathbf{x}^{(s)})}{q_\phi(\mathbf{z}|\mathbf{x}^{(s)})} \\ &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(s)})}[\log p_\theta(\mathbf{x}^{(t)}|\mathbf{y}^{(t)},\mathbf{z}) + \log p_\pi(\mathbf{y}^{(t)})] - \\ &\text{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(s)})||p(\mathbf{z})) = \mathcal{L}_l(\mathbf{x}^{(t)},\mathbf{y}^{(t)}|\mathbf{x}^{(s)}) \quad (12)\end{aligned}$$

4 Learning MSVED

Now that we have described our overall model, we discuss details of the learning process that prove

useful to its success.

4.1 Learning Discrete Latent Variables

One challenge in training our model is that it is not trivial to perform back-propagation through discrete random variables, and thus it is difficult to learn in the models containing discrete tags such as MSVAE or MSVED.[4] To alleviate this problem, we use the recently proposed Gumbel-Softmax trick (Maddison et al., 2014; Gumbel and Lieblein, 1954) to create a differentiable estimator for categorical variables.

The Gumbel-Max trick (Gumbel and Lieblein, 1954) offers a simple way to draw samples from a categorical distribution with class probabilities $\pi_1, \pi_2, \cdots$ by using the `argmax` operation as follows: $\text{one\_hot}(\arg\max_i[g_i + \log \pi_i])$, where $g_1, g_2, \cdots$ are i.i.d. samples drawn from the Gumbel(0,1) distribution.[5] When making inferences on the morphological labels $y_1, y_2, \cdots$, the Gumbel-Max trick can be approximated by the continuous softmax function with temperature τ to generate a sample vector $\hat{\mathbf{y}}_i$ for each label i:

$$\hat{y}_{ij} = \frac{\exp((\log(\pi_{ij}) + g_{ij})/\tau)}{\sum_{k=1}^{N_i} \exp((\log(\pi_{ik}) + g_{ik})/\tau} \quad (13)$$

where N_i is the number of classes of label i. When τ approaches zero, the generated sample $\hat{\mathbf{y}}_i$ becomes a one-hot vector. When $\tau > 0$, $\hat{\mathbf{y}}_i$ is smooth w.r.t π_i. In experiments, we start with a relatively large temperature and decrease it gradually.

4.2 Learning Continuous Latent Variables

MSVED aims at generating the target sequence conditioned on the latent variable $\mathbf{z}$ and the target labels $\mathbf{y}^{(t)}$. This requires the encoder to generate an informative representation $\mathbf{z}$ encoding the content of the $\mathbf{x}^{(s)}$. However, the variational lower bound in our loss function contains the KL-divergence between the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior $p(\mathbf{z})$, which is relatively easy to learn compared with learning to generate output from a latent representation. We observe that with the vanilla implementation the KL cost quickly decreases to near zero, setting $q_\phi(\mathbf{z}|\mathbf{x})$ equal to standard normal distribution. In this case, the RNN decoder can easily rely on the true output of last time step during training to decode the next token, which degenerates into an RNN language model. Hence, the latent variables are ignored by the decoder and cannot encode any useful information. The latent variable $\mathbf{z}$ learns an undesirable distribution that coincides with the imposed prior distribution but has no contribution to the decoder. To force the decoder to use the latent variables, we take the following two approaches which are similar to Bowman et al. (2016).

KL-Divergence Annealing: We add a coefficient λ to the KL cost and gradually anneal it from zero to a predefined threshold $\lambda_{\mathbf{m}}$. At the early stage of training, we set λ to be zero and let the model first figure out how to project the representation of the source sequence to a roughly right point in the space and then regularize it with the KL cost. Although we are not optimizing the tight variational lower bound, the model balances well between generation and regularization. This technique can also be seen in (Kočiskỳ et al., 2016; Miao and Blunsom, 2016).

Input Dropout in the Decoder: Besides annealing the KL cost, we also randomly drop out the input token with a probability of β at each time step of the decoder during learning. The previous ground-truth token embedding is replaced with a zero vector when dropped. In this way, the RNN decoder could not fully rely on the ground-truth previous token, which ensures that the decoder uses information encoded in the latent variables.

5 Architecture for Morphological Reinflection

Training details: For the morphological reinflection task, our supervised training data consists of source $\mathbf{x}^{(s)}$, target $\mathbf{x}^{(t)}$, and target tags $\mathbf{y}^{(t)}$. We test three variants of our model trained using different types of data and different loss functions. First, the single-directional supervised model (**SD-Sup**) is purely supervised: it only decodes the target word from the given source word with the loss function $\mathcal{L}_l(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}|\mathbf{x}^{(s)})$ from Eq. 12. Second, the bi-directional supervised model (**BD-Sup**) is trained in both directions: decoding the target word from the source word and decoding the source word from the target word, which corresponds to the loss function $\mathcal{L}_l(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}|\mathbf{x}^{(s)}) + \mathcal{L}_u(\mathbf{x}^{(s)}|\mathbf{x}^{(t)})$ using Eqs. 11–12. Finally, the semi-supervised model (**Semi-sup**) is trained to maxi-

[4] Kingma et al. (2014) solve this problem by marginalizing over all labels, but this is infeasible in our case where we have an exponential number of label combinations.

[5] The Gumbel (0,1) distribution can be sampled by first drawing $u \sim$ Uniform(0,1) and computing $g = -\log(-\log(u))$.

Language	#LD	#ULD	Proposed MSVED SD-Sup	Proposed MSVED BD-Sup	Proposed MSVED Semi-sup	Baseline MED Single	Baseline MED Ensemble
Turkish	12,798	29,608	93.25	95.66†	**97.25**‡	89.56	95.00
Hungarian	19,200	34,025	97.00	98.54†	**99.16**‡	96.46	98.37
Spanish	12,799	72,151	88.32	91.50	93.74	94.74†‡	**96.69**
Russian	12,798	67,691	75.77	83.07	86.80‡	83.55†	**87.13**
Navajo	12,635	6,839	85.00	95.37†	**98.25**‡	63.62	83.00
Maltese	19,200	46,918	84.83	88.21†	**88.46**‡	79.59	84.25
Arabic	12,797	53,791	79.13	92.62†	**93.37**‡	72.58	82.80
Georgian	12,795	46,562	89.31	93.63†	95.97‡	91.06	**96.21**
German	12,777	56,246	75.55	84.08	90.28‡	89.11†	**92.41**
Finnish	12,800	74,687	75.59	85.11	91.20‡	85.63†	**93.18**
Avg. Acc	–	–	84.38	90.78†	**93.45**‡	84.59	90.90

Table 1: Results for Task 3 of SIGMORPHON 2016 on Morphology Reinflection. † represents the best single supervised model score, ‡ represents the best model including semi-supervised models, and bold represents the best score overall. **#LD** and **#ULD** are the number of supervised data and unlabeled words respectively.

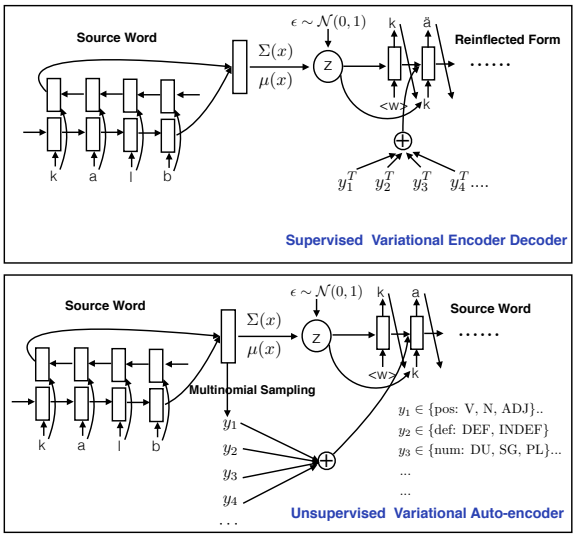


Figure 3: Model architecture for labeled and unlabeled data. For the encoder-decoder model, only one direction from the source to target is given. The classification model is not illustrated in the diagram.

mize the variational lower bounds and minimize the classification cross-entropy error of 10.

$$\mathcal{L}(\mathbf{x^{(s)}}, \mathbf{x^{(t)}}, \mathbf{y^{(t)}}, \mathbf{x}) = \alpha \cdot \mathcal{U}(\mathbf{x}) + \mathcal{L}_u(\mathbf{x}^{(s)}|\mathbf{x}^{(t)}) + \mathcal{L}_l(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}|\mathbf{x}^{(s)}) - \mathcal{D}(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}) \quad (14)$$

The weight α controls the relative weight between the loss from unlabeled data and labeled data.

We use Monte Carlo methods to estimate the expectation over the posterior distribution $q(\mathbf{z}|\mathbf{x})$ and $q(\mathbf{y}|\mathbf{x})$ inside the objective function 14. Specifically, we draw Gumbel noise and Gaussian noise one at a time to compute the latent variables $\mathbf{y}$ and $\mathbf{z}$.

The overall model architecture is shown in Fig. 3. Each character and each label is associated with a continuous vector. We employ Gated Recurrent Units (GRUs) for the encoder and decoder. Let $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ denote the hidden state of the forward and backward encoder RNN at time step t. $\mathbf{u}$ is the hidden representation of $\mathbf{x}^{(s)}$ concatenating the last hidden state from both directions i.e. $[\overrightarrow{h_T}; \overleftarrow{h_T}]$ where T is the word length. $\mathbf{u}$ is used as the input for the inference model on $\mathbf{z}$. We represent $\mu(\mathbf{u})$ and $\sigma^2(\mathbf{u})$ as MLPs and sample $\mathbf{z}$ from $\mathcal{N}(\mu(\mathbf{u}), \text{diag}(\sigma^2(\mathbf{u})))$, using $\mathbf{z} = \mu + \sigma \circ \epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Similarly, we can obtain the hidden representation of $\mathbf{x}^{(t)}$ and use this as input to the inference model on each label $\mathbf{y}_i^{(t)}$ which is also an MLP following a softmax layer to generate the categorical probabilities of target labels.

In decoding, we use 3 types of information in calculating the probability of the next character : (1) the current decoder state, (2) a tag context vector using attention (Bahdanau et al., 2015) over the tag embeddings, and (3) the latent variable $\mathbf{z}$. The intuition behind this design is that we would like the model to constantly consider the lemma represented by $\mathbf{z}$, and also reference the tag corresponding to the current morpheme being generated at this point. We do not marginalize over the latent variable $\mathbf{z}$ however, instead we use the mode μ of $\mathbf{z}$ as the latent representation for $\mathbf{z}$. We use beam search with a beam size of 8 to perform search over the character vocabulary at each decoding time step.

Other experimental setups: All hyperparameters are tuned on the validation set, and include the following: For KL cost annealing, λ_m is set to be 0.2 for all language settings. For character drop-out at the decoder, we empirically set β to be 0.4 for all languages. We set the dimension of character embeddings to be 300, tag label embeddings to be 200, RNN hidden state to be 256, and

latent variable $\mathbf{z}$ to be 150. We set α the weight for the unsupervised loss to be 0.8. We train the model with Adadelta (Zeiler, 2012) and use early-stop with a patience of 10.

6 Experiments

6.1 Background: SIGMORPHON 2016

SIGMORPHON 2016 is a shared task on morphological inflection over 10 different morphologically rich languages. There are a total of three tasks, the most difficult of which is task 3, which requires the system to output the reinflection of an inflected word.[6] The training data format in task 3 is in triples: (source word, target labels, target word). In the test phase, the system is asked to generate the target word given a source word and the target labels. There are a total of three tracks for each task, divided based the amount of supervised data that can be used to solve the problem, among which track 2 has the strictest limitation of only using data for the corresponding task. As this is an ideal testbed for our method, which can learn from unlabeled data, we choose track 2 and task 3 to test our our model's ability to exploit this data.

As a baseline, we compare our results with the MED system (Kann and Schütze, 2016a) which achieved state-of-the-art results in the shared task. This system used an encoder-decoder model with attention on the concatenated source word and target labels. Its best result is obtained from an ensemble of five RNN encoder-decoders (**Ensemble**). To make a fair comparison with our models, which don't use ensembling, we also calculated single model results (**Single**).

All models are trained using the labeled training data provided for task 3. For our semi-supervised model (**Semi-sup**), we also leverage unlabeled data from the training and validation data for tasks 1 and 2 to train variational auto-encoders.

6.2 Results and Analysis

From the results in Tab. 1, we can glean a number of observations. First, comparing the results of our full **Semi-sup** model, we can see that for all languages except Spanish, it achieves accuracies better than the single MED system, often by a large margin. Even compared to the MED ensembled model, our single-model system is quite competitive, achieving higher accuracies for Hungarian, Navajo, Maltese, and Arabic, as well as achieving average accuracies that are state-of-the-art.

[6]Task 1 is inflection of a lemma word and task 2 is reinflection but also provides the source word labels.

Language	Prefix	Stem	Suffix
Turkish	0.21	1.12	98.75
Hungarian	0.00	0.08	99.79
Spanish	0.09	3.25	90.74
Russian	0.66	7.70	85.00
Navajo	77.64	18.38	26.40
Maltese	48.81	11.05	98.74
Arabic	68.52	37.04	88.24
Georgian	4.46	0.41	92.47
German	0.84	3.32	89.19
Finnish	0.02	12.33	96.16

Table 2: Percentage of inflected word forms that have modified each part of the lemma (Cotterell et al., 2016) (some words can be inflected zero or multiple times, thus sums may not add to 100%).

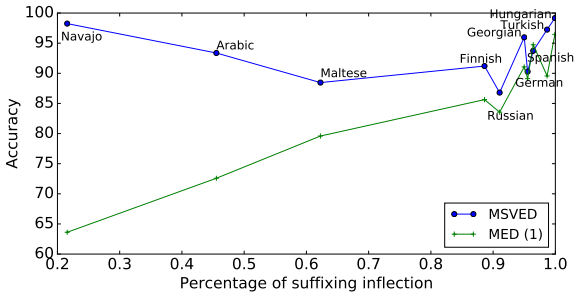


Figure 4: Performance on test data w.r.t. the percentage of suffixing inflection. Points with the same x-axis value correspond to the same language results.

Next, comparing the different varieties of our proposed models, we can see that the semi-supervised model consistently outperforms the bidirectional model for all languages. And similarly, the bidirectional model consistently outperforms the single direction model. From these results, we can conclude that the unlabeled data is beneficial to learn useful latent variables that can be used to decode the corresponding word.

Examining the linguistic characteristics of the models in which our model performs well provides even more interesting insights. Cotterell et al. (2016) estimate how often the inflection process involves prefix changes, stem-internal changes or suffix changes, the results of which are shown in Tab. 2. Among the many languages, the inflection processes of Arabic, Maltese and Navajo are relatively diverse, and contain a large amount of all three forms of inflection. By examining the experimental results together with the morphological inflection process of different languages, we found that among all the languages, Navajo, Maltese and Arabic obtain the largest gains in performance compared with the ensem-

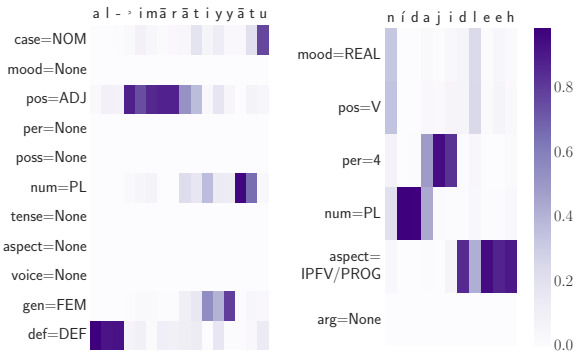


Figure 5: Two examples of attention weights on target linguistic labels: Arabic (**Left**) and Navajo (**Right**). When a tag equals `None`, it means the word does not have this tag.

bled MED system. To demonstrate this visually, in Fig. 4, we compare the semi-supervised MSVED with the MED single model w.r.t. the percentage of suffixing inflection of each language, showing this clear trend.

This strongly demonstrates that our model is agnostic to different morphological inflection forms whereas the conventional encoder-decoder with attention on the source input tends to perform better on suffixing-oriented morphological inflection. We hypothesize that for languages that the inflection mostly comes from suffixing, transduction is relatively easy because the source and target words share the same prefix and the decoder can copy the prefix of the source word via attention. However, for languages in which different inflections of a lemma go through different morphological processes, the inflected word and the target word may differ greatly and thus it is crucial to first analyze the lemma of the inflected word before generating the corresponding the reinflection form based on the target labels. This is precisely what our model does by extracting the lemma representation $\mathbf{z}$ learned by the variational inference model.

6.3 Analysis on Tag Attention

To analyze how the decoder attends to the linguistic labels associated with the target word, we randomly pick two words from the Arabic and Navajo test set and plot the attention weight in Fig. 5. The Arabic word "al-'imārātiyyātu" is an adjective which means "Emirati", and its source word in the test data is "'imārātiyyin" [7]. Both of these are declensions of "'imārātiyy". The source word is

[7] https://en.wiktionary.org/wiki/%D8%A5%D9%85%D8%A7%D8%B1%D8%A7%D8%AA%D9%8A

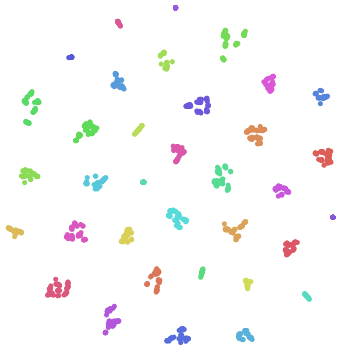


Figure 6: Visualization of latent variables $\mathbf{z}$ for Maltese with 35 pseudo-lemma groups in the figure.

singular, masculine, genitive and indefinite, while the required inflection is plural, feminine, nominative and definite. We can see from the left heat map that the attention weights are turned on at several positions of the word when generating corresponding inflections. For example, "al-" in Arabic is the definite article that marks definite nouns. The same phenomenon can also be observed in the Navajo example, as well as other languages, but due to space limitation, we don't provide detailed analysis here.

6.4 Visualization of Latent Lemmas

To investigate the learned latent representations, in this section we visualize the $\mathbf{z}$ vectors, examining whether the latent space groups together words with the same lemma. Each sample in SIGMORPHON 2016 contains source word and target words which share the same lemma. We run a heuristic process to assign pairs of words to groups that likely share a lemma by grouping together word pairs for which at least one of the words in each pair shares a surface form. This process is not error free – errors may occur in the case where multiple lemmas share the same surface form – but in general the groupings will generally reflect lemmas except in these rare erroneous cases, so we dub each of these groups a *pseudo-lemma*.

In Fig. 6, we randomly pick 1500 words from Maltese and visualize the continuous latent vectors of these words. We compute the latent vectors as $\mu_\phi(\mathbf{x})$ in the variational posterior inference (Eq. 6) without adding the variance. As expected, words that belong to the same pseudo-lemma (in the same color) are projected into adjacent points in the two-dimensional space. This demonstrates that the continuous latent variable captures the canonical form of a set of words and demonstrates the effectiveness of the proposed representation.

Language	Src Word	Tgt Labels	Gold Tgt	MED	Ours
Turkish	kocama	pos=N,poss=PSS1S,case=ESS,num=SG	kocamda	kocama	kocamda
	yaratmamdan	pos=N,case=NOM,num=SG	yaratma	yaratma	yaratman
	bitimizde	pos=N,tense=PST,per=1,num=SG	bittik	bitiydik	bittim
Maltese	ndammhomli	pos=V,polar=NEG,tense=PST,num=SG	tindammhiex	ndammejthiex	tindammhiex
	tqożżhieli	pos=V,polar=NEG,tense=PST,num=SG	tqożżx	tqożżx	qażżejtx
	tissikkmuhomli	pos=V,polar=POS,tense=PST,num=PL	ssikkmulna	tissikkmulna	tissikkmulna
Finnish	verovapaatta	pos=ADJ,case=PRT,num=PL	verovapaita	verovappaita	verovapaita
	turrumme	pos=V,mood=POT,tense=PRS,num=PL	turtunemme	turtunemme	turrunemme
	sukunimin	pos=N,case=PRIV,num=PL	sukunimitt	sukunimeitta	sukunimeitta

Table 3: Randomly picked output examples on the test data. Within each block, the first, second and third lines are outputs that ours is correct and MED's is wrong, ours is wrong and MED's is correct, both are wrong respectively.

6.5 Analyzing Effects of Size of Unlabeled Data

From Tab. 1, we can see that semi-supervised learning always performs better than supervised learning without unlabeled data. In this section, we investigate to what extent the size of unlabeled data can help with performance. We process a German corpus from a 2017 Wikipedia dump and obtain more than 100,000 German words. These words are ranked in order of occurrence frequency in Wikipedia. The data contains a certain amount of noise since we did not apply any special processing. We shuffle all unlabeled data from both the Wikipedia and the data provided in the shared task used in previous experiments, and increase the number of unlabeled words used in learning by 10,000 each time, and finally use all the unlabeled data (more than 150,000 words) to train the model. Fig. 7 shows that the performance on the test data improves as the amount of unlabeled data increases, which implies that the unsupervised learning continues to help improve the model's ability to model the latent lemma representation even as we scale to a noisy, real, and relatively large-scale dataset. Note that the growth rate of the performance grows slower as more data is added, because although the number of unlabeled data is increasing, the model has seen most word patterns in a relatively small vocabulary.

6.6 Case Study on Reinflected Words

In Tab. 3, we examine some model outputs on the test data from the MED system and our model respectively. It can be seen that most errors of MED and our models can be ascribed to either over-copy or under-copy of characters. In particular, from the complete outputs we observe that our model tends to be more aggressive in its changes, resulting in it performing more complicated transformations, both successfully (such as Maltese "ndammhomli" to "tindammhiex") and unsuccessfully ("tqożżx" to "qażżejtx"). In contrast, the attentional encoder-decoder model is more conservative in its changes, likely because it is less effective in learning an abstracted representation for the lemma, and instead copies characters directly from the input.

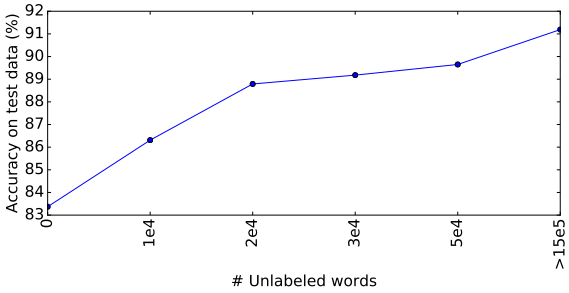


Figure 7: Performance on the German test data w.r.t. the amount of unlabeled Wikipedia data.

7 Conclusion and Future Work

In this work, we propose a multi-space variational encoder-decoder framework for labeled sequence transduction problem. The MSVED performs well in the task of morphological reinflection, outperforming the state of the art, and further improving with the addition of external unlabeled data. Future work will adapt this framework to other sequence transduction scenarios such as machine translation, dialogue generation, question answering, where continuous and discrete latent variables can be abstracted to guide sequence generation.

Acknowledgments

The authors thank Jiatao Gu, Xuezhe Ma, Zihang Dai and Pengcheng Yin for their helpful discussions. This work has been supported in part by an Amazon Academic Research Award.

References

Iñaki Alegria and Izaskun Etxeberria. 2016. Ehu at the sigmorphon 2016 shared task. a simple proposal: Grapheme-to-phoneme for inflection. *In Proceedings of the 2016 Meeting of SIGMORPHON* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *The International Conference on Learning Representations* .

Justin Bayer and Christian Osendorfer. 2014. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610* .

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *Proceedings of CoNLL* .

Victor Chahuneau, Eva Schlinger, Noah A Smith, and Chris Dyer. 2013. Translating into morphologically rich languages with synthetic phrases. Association for Computational Linguistics.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*. pages 2980–2988.

R. Cotterell, C. Kirov, J. Sylak-Glassman, D. Yarowsky, J. Eisner, and M. Hulden. 2016. The sigmorphon 2016 shared taskmorphological reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Kareem Darwish and Douglas W Oard. 2007. Adapting morphology for arabic information retrieval. In *Arabic Computational Morphology*, Springer, pages 245–262.

Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* .

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 1185–1195.

Otto Fabius and Joost R van Amersfoort. 2014. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581* .

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 634–643.

Emil Julius Gumbel and Julius Lieblein. 1954. Statistical theory of extreme values and some practical applications: a series of lectures. *US Government Printing Office Washington* .

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558* .

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural multi-source morphological reinflection. *arXiv preprint arXiv:1612.06027* .

Katharina Kann and Hinrich Schütze. 2016a. Med: The lmu system for the sigmorphon 2016 shared task on morphological reinflection. In *In Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Berlin, Germany.

Katharina Kann and Hinrich Schütze. 2016b. Single-model encoder-decoder with explicit morphological representation for reinflection. In *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1328–1338.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*. Montréal, Canada, pages 3581–3589.

D.P. Kingma and M. Welling. 2014. Auto-encoding variational bayes. In *The International Conference on Learning Representations*.

Tomáš Kočiskỳ, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. *the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)* .

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. 2016. Auxiliary deep generative models. *Proceedings of the 33rd International Conference on Machine Learning* .

Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Advances in Neural Information Processing Systems*. pages 3086–3094.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. *the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)* .

Garrett Nicolai, Bradley Hauer, Adam St. Arnaud, and Grzegorz Kondrak. 2016. Morphological reinflection via discriminative string transduction. *In Proceedings of the 2016 Meeting of SIGMORPHON* .

Robert Ostling. 2016. Morphological reinflection with convolutional neural networks. *In Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology* page 23.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of The North American Chapter of the Association for Computational Linguistics (NAACL)*. pages 35–40.

Dima Taji, Ramy Eskander, Nizar Habash, and Owen Rambow. 2016. The columbia university - new york university abu dhabi sigmorphon 2016 morphological reinflection shared task submission. *In Proceedings of the 2016 Meeting of SIGMORPHON* .

Keiichi Tokuda, Takashi Masuko, Noboru Miyazaki, and Takao Kobayashi. 2002. Multi-space probability distribution hmm. *IEICE TRANSACTIONS on Information and Systems* 85(3):455–464.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. pages 514–522.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* .

Scalable Bayesian Learning of Recurrent Neural Networks for Language Modeling

Zhe Gan*, **Chunyuan Li***†, **Changyou Chen, Yunchen Pu, Qinliang Su, Lawrence Carin**
Department of Electrical and Computer Engineering, Duke University
{zg27, cl319, cc448, yp42, qs15, lcarin}@duke.edu

Abstract

Recurrent neural networks (RNNs) have shown promising performance for language modeling. However, traditional training of RNNs using back-propagation through time often suffers from overfitting. One reason for this is that stochastic optimization (used for large training sets) does not provide good estimates of model uncertainty. This paper leverages recent advances in stochastic gradient Markov Chain Monte Carlo (also appropriate for large training sets) to learn weight uncertainty in RNNs. It yields a principled Bayesian learning algorithm, adding gradient noise during training (enhancing exploration of the model-parameter space) and model averaging when testing. Extensive experiments on various RNN models and across a broad range of applications demonstrate the superiority of the proposed approach relative to stochastic optimization.

1 Introduction

Language modeling is a fundamental task, used for example to predict the next word or character in a text sequence given the context. Recently, recurrent neural networks (RNNs) have shown promising performance on this task (Mikolov et al., 2010; Sutskever et al., 2011). RNNs with Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997) have emerged as a popular architecture, due to their representational power and effectiveness at capturing long-term dependencies.

RNNs are usually trained via back-propagation through time (Werbos, 1990), using stochastic optimization methods such as stochastic gradient descent (SGD) (Robbins and Monro, 1951); stochastic methods of this type are particularly important for training with large data sets. However, this approach often provides a *maximum a posteriori* (MAP) estimate of model parameters. The MAP solution is a single point estimate, ignoring weight uncertainty (Blundell et al., 2015; Hernández-Lobato and Adams, 2015). Natural language often exhibits significant variability, and hence such a point estimate may make over-confident predictions on test data.

To alleviate overfitting RNNs, good regularization is known as a key factor to successful applications. In the neural network literature, Bayesian learning has been proposed as a principled method to impose regularization and incorporate model uncertainty (MacKay, 1992; Neal, 1995), by imposing prior distributions on model parameters. Due to the intractability of posterior distributions in neural networks, Hamiltonian Monte Carlo (HMC) (Neal, 1995) has been used to provide sample-based approximations to the true posterior. Despite the elegant theoretical property of asymptotic convergence to the true posterior, HMC and other conventional Markov Chain Monte Carlo methods are not scalable to large training sets.

This paper seeks to scale up Bayesian learning of RNNs to meet the challenge of the increasing amount of "big" sequential data in natural language processing, leveraging recent advances in *stochastic* gradient Markov Chain Monte Carlo (SG-MCMC) algorithms (Welling and Teh, 2011; Chen et al., 2014; Ding et al., 2014; Li et al., 2016a,b). Specifically, instead of training a single network, SG-MCMC is employed to train an *ensemble* of networks, where each network has its parameters drawn from a shared posterior distribution. This is implemented by adding additional

*Equal contribution. †Corresponding author.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 321–331
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1030

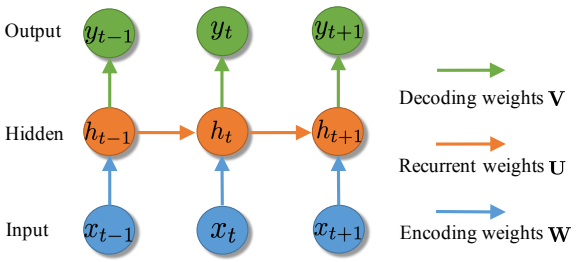


Figure 1: Illustration of different weight learning strategies in a single-hidden-layer RNN. Stochastic optimization used for MAP estimation puts fixed values on all weights. Naive dropout is allowed to put weight uncertainty only on encoding and decoding weights, and fixed values on recurrent weights. The proposed SG-MCMC scheme imposes distributions on all weights.

gradient noise during training and utilizing model averaging when testing.

This simple procedure has the following salutary properties for training neural networks: (i) When training, the injected noise encourages model-parameter trajectories to better explore the parameter space. This procedure was also empirically found effective in Neelakantan et al. (2016). (ii) Model averaging when testing alleviates overfitting and hence improves generalization, transferring uncertainty in the learned model parameters to subsequent prediction. (iii) In theory, both asymptotic and non-asymptotic consistency properties of SG-MCMC methods in posterior estimation have been recently established to guarantee convergence (Chen et al., 2015a; Teh et al., 2016). (iv) SG-MCMC is scalable; it shares the same level of computational cost as SGD in training, by only requiring the evaluation of gradients on a small mini-batch. To the authors' knowledge, RNN training using SG-MCMC has not been investigated previously, and is a contribution of this paper. We also perform extensive experiments on several natural language processing tasks, demonstrating the effectiveness of SG-MCMC for RNNs, including character/word-level language modeling, image captioning and sentence classification.

2 Related Work

Several scalable Bayesian learning methods have been proposed recently for neural networks. These come in two broad categories: stochastic variational inference (Graves, 2011; Blundell et al., 2015; Hernández-Lobato and Adams, 2015) and SG-MCMC methods (Korattikara et al., 2015; Li et al., 2016a). While prior work focuses on feed-forward neural networks, there has been little if any research reported for RNNs using SG-MCMC.

Dropout (Hinton et al., 2012; Srivastava et al., 2014) is a commonly used regularization method for training neural networks. Recently, several works have studied how to apply dropout to RNNs (Pachitariu and Sahani, 2013; Bayer et al., 2013; Pham et al., 2014; Zaremba et al., 2014; Bluche et al., 2015; Moon et al., 2015; Semeniuta et al., 2016; Gal and Ghahramani, 2016b). Among them, naive dropout (Zaremba et al., 2014) can impose weight uncertainty only on *encoding weights* (those that connect input to hidden units) and *decoding weights* (those that connect hidden units to output), but not the *recurrent weights* (those that connect consecutive hidden states). It has been concluded that noise added in the recurrent connections leads to model instabilities, hence disrupting the RNN's ability to model sequences.

Dropout has been recently shown to be a variational approximation technique in Bayesian learning (Gal and Ghahramani, 2016a; Kingma et al., 2015). Based on this, (Gal and Ghahramani, 2016b) proposed a new variant of dropout that can be successfully applied to recurrent layers, where the same dropout masks are shared along time for encoding, decoding and recurrent weights, respectively. Alternatively, we focus on SG-MCMC, which can be viewed as the Bayesian interpretation of dropout from the perspective of posterior sampling (Li et al., 2016c); this also allows imposition of model uncertainty on recurrent layers, enhancing performance. A comparison of naive dropout and SG-MCMC is illustrated in Fig. 1.

3 Recurrent Neural Networks

3.1 RNN as Bayesian Predictive Models

Consider data $\mathcal{D} = \{\mathbf{D}_1, \cdots, \mathbf{D}_N\}$, where $\mathbf{D}_n \triangleq (\mathbf{X}_n, \mathbf{Y}_n)$, with input $\mathbf{X}_n$ and output $\mathbf{Y}_n$. Our goal is to learn model parameters $\boldsymbol{\theta}$ to best characterize the relationship from $\mathbf{X}_n$ to $\mathbf{Y}_n$, with corresponding data likelihood $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{D}_n|\boldsymbol{\theta})$. In Bayesian statistics, one sets a prior on $\boldsymbol{\theta}$ via distribution $p(\boldsymbol{\theta})$. The posterior $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})$ reflects the belief concerning the model parameter distribution after observing the data. Given a test input $\tilde{\mathbf{X}}$ (with missing output $\tilde{\mathbf{Y}}$), the uncertainty learned in training

is transferred to prediction, yielding the posterior predictive distribution:

$$p(\tilde{\mathbf{Y}}|\tilde{\mathbf{X}},\mathcal{D}) = \int_{\boldsymbol{\theta}} p(\tilde{\mathbf{Y}}|\tilde{\mathbf{X}},\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\mathrm{d}\boldsymbol{\theta}\,. \quad (1)$$

When the input is a sequence, RNNs may be used to parameterize the input-output relationship. Specifically, consider input sequence $\mathbf{X} = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T\}$, where $\boldsymbol{x}_t$ is the input data vector at time t. There is a corresponding hidden state vector $\boldsymbol{h}_t$ at each time t, obtained by recursively applying the *transition function* $\boldsymbol{h}_t = \mathcal{H}(\boldsymbol{h}_{t-1},\boldsymbol{x}_t)$ (specified in Section 3.2; see Fig. 1). The output $\mathbf{Y}$ differs depending on the application: a sequence $\{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T\}$ in language modeling or a discrete label in sentence classification. In RNNs the corresponding *decoding function* is $p(\boldsymbol{y}|\boldsymbol{h})$, described in Section 3.3.

3.2 RNN Architectures

The transition function $\mathcal{H}(\cdot)$ can be implemented with a *gated* activation function, such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) or a Gated Recurrent Unit (GRU) (Cho et al., 2014). Both the LSTM and GRU have been proposed to address the issue of learning long-term sequential dependencies.

Long Short-Term Memory The LSTM architecture addresses the problem of learning long-term dependencies by introducing a *memory cell*, that is able to preserve the state over long periods of time. Specifically, each LSTM unit has a cell containing a state $\boldsymbol{c}_t$ at time t. This cell can be viewed as a memory unit. Reading or writing the cell is controlled through sigmoid gates: input gate $\boldsymbol{i}_t$, forget gate $\boldsymbol{f}_t$, and output gate $\boldsymbol{o}_t$. The hidden units $\boldsymbol{h}_t$ are updated as

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\mathbf{W}_i\boldsymbol{x}_t + \mathbf{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i)\,,\\
\boldsymbol{f}_t &= \sigma(\mathbf{W}_f\boldsymbol{x}_t + \mathbf{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f)\,,\\
\boldsymbol{o}_t &= \sigma(\mathbf{W}_o\boldsymbol{x}_t + \mathbf{U}_o\boldsymbol{h}_{t-1} + \boldsymbol{b}_o)\,,\\
\tilde{\boldsymbol{c}}_t &= \tanh(\mathbf{W}_c\boldsymbol{x}_t + \mathbf{U}_c\boldsymbol{h}_{t-1} + \boldsymbol{b}_c)\,,\\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t\,,\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)\,,
\end{aligned}$$

where $\sigma(\cdot)$ denotes the logistic sigmoid function, and $\odot$ represents the element-wise matrix multiplication operator. $\mathbf{W}_{\{i,f,o,c\}}$ are *encoding weights*, and $\mathbf{U}_{\{i,f,o,c\}}$ are *recurrent weights*, as shown in Fig. 1. $\boldsymbol{b}_{\{i,f,o,c\}}$ are bias terms.

Variants Similar to the LSTM unit, the GRU also has gating units that modulate the flow of information inside the hidden unit. It has been shown that a GRU can achieve similar performance to an LSTM in sequence modeling (Chung et al., 2014). We specify the GRU in the Supplementary Material.

The LSTM can be extended to the bidirectional LSTM and multilayer LSTM. A bidirectional LSTM consists of two LSTMs that are run in parallel: one on the input sequence and the other on the reverse of the input sequence. At each time step, the hidden state of the bidirectional LSTM is the concatenation of the forward and backward hidden states. In multilayer LSTMs, the hidden state of an LSTM unit in layer ℓ is used as input to the LSTM unit in layer $\ell+1$ at the same time step (Graves, 2013).

3.3 Applications

The proposed Bayesian framework can be applied to any RNN model; we focus on the following tasks to demonstrate the ideas.

Language Modeling In word-level language modeling, the input to the network is a sequence of words, and the network is trained to predict the next word in the sequence with a softmax classifier. Specifically, for a length-T sequence, denote $\boldsymbol{y}_t = \boldsymbol{x}_{t+1}$ for $t = 1,\ldots,T-1$. $\boldsymbol{x}_1$ and $\boldsymbol{y}_T$ are always set to a special START and END token, respectively. At each time t, there is a decoding function $p(\boldsymbol{y}_t|\boldsymbol{h}_t) = \mathrm{softmax}(\mathbf{V}\boldsymbol{h}_t)$ to compute the distribution over words, where $\mathbf{V}$ are the *decoding weights* (the number of rows of $\mathbf{V}$ corresponds to the number of words/characters). We also extend this basic language model to consider other applications: (i) a *character-level language model* can be specified in a similar manner by replacing words with characters (Karpathy et al., 2016). (ii) *Image captioning* can be considered as a conditional language modeling problem, in which we learn a generative language model of the caption conditioned on an image (Vinyals et al., 2015; Gan et al., 2017).

Sentence Classification Sentence classification aims to assign a semantic category label $\boldsymbol{y}$ to a whole sentence $\mathbf{X}$. This is usually implemented through applying the decoding function once at the end of sequence: $p(\boldsymbol{y}|\boldsymbol{h}_T) = \mathrm{softmax}(\mathbf{V}\boldsymbol{h}_T)$, where the final hidden state of a RNN $\boldsymbol{h}_T$ is often considered as the summary of the sentence (here

the number of rows of $\mathbf{V}$ corresponds to the number of classes).

4 Scalable Learning with SG-MCMC

4.1 The Pitfall of Stochastic Optimization

Typically there is no closed-form solution for the posterior $p(\boldsymbol{\theta}|\mathcal{D})$, and traditional Markov Chain Monte Carlo (MCMC) methods (Neal, 1995) scale poorly for large N. To ease the computational burden, stochastic optimization is often employed to find the MAP solution. This is equivalent to minimizing an objective of regularized loss function $U(\boldsymbol{\theta})$ that corresponds to a (non-convex) model of interest: $\boldsymbol{\theta}_{\text{MAP}} = \arg\min U(\boldsymbol{\theta})$, $U(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta}|\mathcal{D})$. The expectation in (1) is approximated as:

$$p(\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}, \mathcal{D}) = p(\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}, \boldsymbol{\theta}_{\text{MAP}}) \,. \tag{2}$$

Though simple and effective, this procedure largely loses the benefit of the Bayesian approach, because the uncertainty on weights is ignored. To more accurately approximate (1), we employ stochastic gradient (SG) MCMC (Welling and Teh, 2011).

4.2 Large-scale Bayesian Learning

The negative log-posterior is

$$U(\boldsymbol{\theta}) \triangleq -\log p(\boldsymbol{\theta}) - \sum_{n=1}^{N} \log p(\mathbf{D}_n|\boldsymbol{\theta}). \tag{3}$$

In optimization, $E = -\sum_{n=1}^{N} \log p(\mathbf{D}_n|\boldsymbol{\theta})$ is typically referred to as the loss function, and $R \propto -\log p(\boldsymbol{\theta})$ as a regularizer.

For large N, stochastic approximations are often employed:

$$\tilde{U}_t(\boldsymbol{\theta}) \triangleq -\log p(\boldsymbol{\theta}) - \frac{N}{M} \sum_{m=1}^{M} \log p(\mathbf{D}_{i_m}|\boldsymbol{\theta}), \tag{4}$$

where $\mathcal{S}_m = \{i_1, \cdots, i_M\}$ is a *random* subset of the set $\{1, 2, \cdots, N\}$, with $M \ll N$. The gradient on this mini-batch is denoted as $\tilde{\boldsymbol{f}}_t = \nabla \tilde{U}_t(\boldsymbol{\theta})$, which is an unbiased estimate of the true gradient. The evaluation of (4) is cheap even when N is large, allowing one to efficiently collect a sufficient number of samples in large-scale Bayesian learning, $\{\boldsymbol{\theta}_s\}_{s=1}^{S}$, where S is the number of samples (this will be specified later). These samples are used to construct a sample-based estimation to the expectation in (1):

Table 1: SG-MCMC algorithms and their optimization counterparts. Algorithms in the same row share similar characteristics.

Algorithms	SG-MCMC	Optimization
Basic	SGLD	SGD
Precondition	pSGLD	RMSprop/Adagrad
Momentum	SGHMC	momentum SGD
Thermostat	SGNHT	Santa

$$p(\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p(\tilde{\mathbf{Y}}|\tilde{\mathbf{X}}, \boldsymbol{\theta}_s) \,. \tag{5}$$

The finite-time estimation errors of SG-MCMC methods are bounded (Chen et al., 2015a), which guarantees (5) is an unbiased estimate of (1) asymptotically under appropriate decreasing step-sizes.

4.3 SG-MCMC Algorithms

SG-MCMC and stochastic optimization are parallel lines of work, designed for different purposes; their relationship has recently been revealed in the context of deep learning. The most basic SG-MCMC algorithm has been applied to Langevin dynamics, and is termed SGLD (Welling and Teh, 2011). To help convergence, a momentum term has been introduced in SGHMC (Chen et al., 2014), a "thermostat" has been devised in SGNHT (Ding et al., 2014; Gan et al., 2015) and preconditioners have been employed in pSGLD (Li et al., 2016a). These SG-MCMC algorithms often share similar characteristics with their counterpart approaches from the optimization literature such as the momentum SGD, Santa (Chen et al., 2016) and RMSprop/Adagrad (Tieleman and Hinton, 2012; Duchi et al., 2011). The interrelationships between SG-MCMC and optimization-based approaches are summarized in Table 1.

SGLD Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh, 2011) draws posterior samples, with updates

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t \tilde{\boldsymbol{f}}_{t-1} + \sqrt{2\eta_t}\boldsymbol{\xi}_t \,, \tag{6}$$

where η_t is the learning rate, and $\boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_p)$ is a standard Gaussian random vector. SGLD is the SG-MCMC analog to stochastic gradient descent (SGD), whose parameter updates are given by:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t \tilde{\boldsymbol{f}}_{t-1} \,. \tag{7}$$

Algorithm 1: pSGLD

Input: Default hyperparameter settings:
$\eta_t = 1 \times 10^{-3}, \lambda = 10^{-8}, \beta_1 = 0.99.$
Initialize: $\boldsymbol{v}_0 \leftarrow \mathbf{0}, \boldsymbol{\theta}_1 \sim \mathcal{N}(0, \mathbf{I})$;
for $t = 1, 2, \ldots, T$ **do**
  % Estimate gradient from minibatch $\mathcal{S}_t$
  $\tilde{\boldsymbol{f}}_t = \nabla \tilde{U}_t(\boldsymbol{\theta})$;
  % Preconditioning
  $\boldsymbol{v}_t \leftarrow \beta_1 \boldsymbol{v}_{t-1} + (1-\beta_1)\tilde{\boldsymbol{f}}_t \odot \tilde{\boldsymbol{f}}_t$;
  $\mathbf{G}_t^{-1} \leftarrow \text{diag}\left(\mathbf{1} \oslash \left(\lambda \mathbf{1} + \boldsymbol{v}_t^{\frac{1}{2}}\right)\right)$;
  % Parameter update
  $\boldsymbol{\xi}_t \sim \mathcal{N}(0, \eta_t \mathbf{G}_t^{-1})$;
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \frac{\eta_t}{2}\mathbf{G}_t^{-1}\tilde{\boldsymbol{f}}_t + \boldsymbol{\xi}_t$;
end

SGD is guaranteed to converge to a local minimum under mild conditions (Bottou, 2010). The additional Gaussian term in SGLD helps the learning trajectory to explore the parameter space to approximate posterior samples, instead of obtaining a local minimum.

pSGLD Preconditioned SGLD (pSGLD) (Li et al., 2016a) was proposed recently to improve the mixing of SGLD. It utilizes magnitudes of recent gradients to construct a diagonal preconditioner to approximate the Fisher information matrix, and thus adjusts to the local geometry of parameter space by equalizing the gradients so that a constant stepsize is adequate for all dimensions. This is important for RNNs, whose parameter space often exhibits *pathological curvature* and *saddle points* (Pascanu et al., 2013), resulting in slow mixing. There are multiple choices of preconditioners; similar ideas in optimization include Adagrad (Duchi et al., 2011), Adam (Kingma and Ba, 2015) and RMSprop (Tieleman and Hinton, 2012). An efficient version of pSGLD, adopting RMSprop as the preconditioner $\mathbf{G}$, is summarized in Algorithm 1, where $\oslash$ denotes element-wise matrix division. When the preconditioner is fixed as the identity matrix, the method reduces to SGLD.

4.4 Understanding SG-MCMC

To further understand SG-MCMC, we show its close connection to dropout/dropConnect (Srivastava et al., 2014; Wan et al., 2013). These methods improve the generalization ability of deep models, by randomly adding binary/Gaussian noise to the local units or global weights. For neural networks with the nonlinear function $q(\cdot)$ and consecutive layers $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$, dropout and dropConnect are denoted as:

$$\begin{aligned} \text{Dropout:} \quad & \boldsymbol{h}_2 = \boldsymbol{\xi}_0 \odot q(\boldsymbol{\theta}\boldsymbol{h}_1), \\ \text{DropConnect:} \quad & \boldsymbol{h}_2 = q((\boldsymbol{\xi}_0 \odot \boldsymbol{\theta})\boldsymbol{h}_1), \end{aligned}$$

where the injected noise $\boldsymbol{\xi}_0$ can be binary-valued with dropping rate p or its equivalent Gaussian form (Wang and Manning, 2013):

$$\begin{aligned} \text{Binary noise:} \quad & \boldsymbol{\xi}_0 \sim \text{Ber}(p), \\ \text{Gaussian noise:} \quad & \boldsymbol{\xi}_0 \sim \mathcal{N}(1, \frac{p}{1-p}). \end{aligned}$$

Note that $\boldsymbol{\xi}_0$ is defined as a vector for dropout, and a matrix for dropConnect. By combining dropConnect and Gaussian noise from the above, we have the update rule (Li et al., 2016c):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\xi}_0 \odot \boldsymbol{\theta}_t - \frac{\eta}{2}\tilde{\boldsymbol{f}}_t = \boldsymbol{\theta}_t - \frac{\eta}{2}\tilde{\boldsymbol{f}}_t + \boldsymbol{\xi}_0' \,, \quad (8)$$

where $\boldsymbol{\xi}_0' \sim \mathcal{N}\left(0, \frac{p}{(1-p)}\text{diag}(\boldsymbol{\theta}_t^2)\right)$; (8) shows that dropout/ dropConnect and SGLD in (6) share the same form of update rule, with the distinction being that the level of injected noise is different. In practice, the noise injected by SGLD may not be enough. A better way that we find to improve the performance is to jointly apply SGLD and dropout. This method can be interpreted as using SGLD to sample the posterior distribution of a mixture of RNNs, with mixture probability controlled by the dropout rate.

5 Experiments

We present results on several tasks, including character/word-level language modeling, image captioning, and sentence classification. We do not perform any dataset-specific tuning other than early stopping on validation sets. When dropout is utilized, the dropout rate is set to 0.5. All experiments are implemented in Theano (Theano Development Team, 2016), using a NVIDIA GeForce GTX TITAN X GPU with 12GB memory.

The hyper-parameters for the proposed algorithm include step size, minibatch size, thinning interval, number of burn-in epochs and variance of the Gaussian priors. We list the specific values used in our experiments in Table 2. The explanation of these hyperparameters, the initialization of model parameters and model specifications on each dataset are provided in the Supplementary Material.

Table 2: Hyper-parameter settings of pSGLD for different datasets. For PTB, SGLD is used.

Datasets	WP	PTB	Flickr8k	Flickr30k	MR	CR	SUBJ	MPQA	TREC
Minibatch Size	100	32	64	64	50	50	50	50	50
Step Size	2×10^{-3}	1	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
# Total Epoch	20	40	20	20	20	20	20	20	20
Burn-in (#Epoch)	4	4	3	3	1	1	1	1	1
Thinning Interval (#Epoch)	1/2	1/2	1	1/2	1	1	1	1	1
# Samples Collected	32	72	17	34	19	19	19	19	19

5.1 Language Modeling

We first test character-level and word-level language modeling. The setup is as follows.

- Following Karpathy et al. (2016), we test character-level language modeling on the *War and Peace* (WP) novel. The training/validation/test sets contain 260/32/33 batches, in which there are 100 characters. The vocabulary size is 87, and we consider a 2-hidden-layer RNN of dimension 128.
- The *Penn Treebank* (PTB) corpus (Marcus et al., 1993) is used for word-level language modeling. The dataset adopts the standard split (929K training words, 73K validation words, and 82K test words) and has a vocabulary of size 10K. We train LSTMs of three sizes; these are denoted the small/medium/large LSTM. All LSTMs have two layers and are unrolled for 20 steps. The small, medium and large LSTM has 200, 650 and 1500 units per layer, respectively.

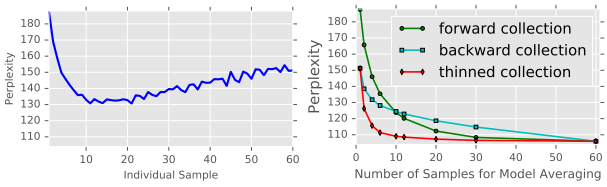
  We consider two types of training schemes on PTB corpus: (i) *Successive minibatches*: Following Zaremba et al. (2014), the final hidden states of the current minibatch are used as the initial hidden states of the subsequent minibatch (successive minibatches sequentially traverse the training set). (ii) *Random minibatches*: The initial hidden states of each minibatch are set to zero vectors, hence we can randomly sample minibatches in each update.

We study the effects of different types of architecture (LSTM/GRU/Vanilla RNN (Karpathy et al., 2016)) on the WP dataset, and effects of different learning algorithms on the PTB dataset. The comparison of test cross-entropy loss on WP is shown in Table 3. We observe that pSGLD consistently outperforms RMSprop. Table 4 summarizes the test set performance on PTB[1]. It is clear

[1]The results reported here do not match Zaremba et al. (2014) due to the implementation details. However, we provide a fair comparison to all methods.

Table 3: Test cross-entropy loss on WP dataset.

Methods	LSTM	GRU	RNN
RMSprop	1.3607	1.2759	1.4239
pSGLD	**1.3375**	**1.2561**	**1.4093**



(a) Single sample (b) Different collections

Figure 2: Effects of collected samples.

that our sampling-based method consistently outperforms the optimization counterpart, where the performance gain mainly comes from adding gradient noise and model averaging. When compared with dropout, SGLD performs better on the small LSTM model, but worse on the medium and large LSTM model. This may imply that dropout is suitable to regularizing large networks, while SGLD exhibits better regularization ability on small networks, partially due to the fact that dropout may inject a higher level of noise during training than SGLD. In order to inject a higher level of noise into SGLD, we empirically apply SGLD and dropout jointly, and found that this provided the best performace on the medium and large LSTM model.

We study three strategies to do model averaging, *i.e.*, *forward collection*, *backward collection* and *thinned collection*. Given samples $(\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_K)$ and the number of samples S used for averaging, *forward collection* refers to using $(\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_S)$ for the evaluation of a test function, *backward collection* refers to using $(\boldsymbol{\theta}_{K-S+1}, \cdots, \boldsymbol{\theta}_K)$, while *thinned collection* chooses samples from $\boldsymbol{\theta}_1$ to $\boldsymbol{\theta}_K$ with interval K/S. Fig. 2 plots the effects of these strategies, where Fig. 2(a) plots the perplexity of every single sample, Fig. 2(b) plots the perplexities using the three schemes. Only after 20

Table 4: Test perplexity on Penn Treebank.

Methods		Small	Medium	Large
Random minibatches	SGD	123.85	126.31	130.25
	SGD+Dropout	136.39	100.12	97.65
	SGLD	**117.36**	109.14	105.86
	SGLD+Dropout	139.54	**99.58**	**94.03**
Successive minibatches	SGD	113.45	123.14	127.68
	SGD+Dropout	117.85	84.60	80.85
	SGLD	**108.61**	121.16	131.40
	SGLD+Dropout	125.44	**82.71**	**78.91**
Literature	Moon et al. (2015)	–	97.0	118.7
	Moon et al. (2015)+ emb. dropout	–	86.5	86.0
	Zaremba et al. (2014)	–	82.7	78.4
	Gal and Ghahramani (2016b)	–	78.6	73.4

samples is a converged perplexity achieved in the thinned collection, while it requires 30 samples for forward collection or 60 samples for backward collection. This is unsurprising, because thinned collection provides a better way to select samples. Nevertheless, averaging of samples provides significantly lower perplexity than using single samples. Note that the overfitting problem in Fig. 2(a) is also alleviated by model averaging.

To better illustrate the benefit of model averaging, we visualize in Fig. 3 the probabilities of each word in a randomly chosen test sentence. The first 3 rows are the results predicted by 3 distinctive model samples, respectively; the bottom row is the result after averaging. Their corresponding perplexities for the test sentence are also shown on the right of each row. The 3 individual samples provide reasonable probabilities. For example, the consecutive words "New York", "stock exchange" and "did not" are assigned with a higher probability. After averaging, we can see a much lower perplexity, as the samples can complement each other. For example, though the second sample can yield the lowest single-model perplexity, its prediction on word "York" is still benefited from the other two via averaging.

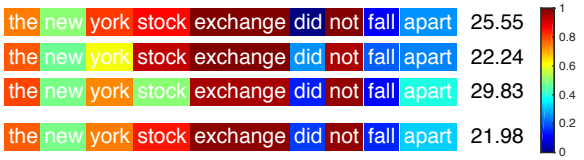


Figure 3: Predictive probabilities obtained by 3 samples and their average. Colors indicate normalized probability of each word. Best viewed in color.

Figure 4: Image captioning with different samples. Left are the given images, right are the corresponding captions. The captions in each box are from the same model sample.

5.2 Image Caption Generation

We next consider the problem of image caption generation, which is a conditional RNN model, where image features are extracted by residual network (He et al., 2016), and then fed into the RNN to generate the caption. We present results on two benchmark datasets, Flickr8k (Hodosh et al., 2013) and Flickr30k (Young et al., 2014). These datasets contain 8,000 and 31,000 images, respectively. Each image is annotated with 5 sentences. A single-layer LSTM is employed with the number of hidden units set to 512.

The widely used BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE-L (Lin, 2004), and CIDEr-D (Vedantam et al., 2015) metrics are used to evaluate the performance. All the metrics are computed by using the code released by the COCO evaluation server (Chen et al., 2015b).

Table 5 presents results for pSGLD/RMSprop

Table 5: Performance on Flickr8k & Flickr30k: BLEU's, METEOR, CIDEr, ROUGE-L and perplexity.

Methods	B-1	B-2	B-3	B-4	METEOR	CIDEr	ROUGE-L	Perp.
Results on Flickr8k								
RMSprop	0.640	0.427	0.288	0.197	0.205	0.476	0.500	16.64
RMSprop + Dropout	0.647	0.444	0.305	0.209	0.208	0.514	0.510	15.72
RMSprop + Gal's Dropout	0.651	0.443	0.305	0.209	0.206	0.501	0.509	14.70
pSGLD	**0.669**	**0.463**	**0.321**	**0.224**	**0.214**	**0.535**	**0.522**	14.29
pSGLD + Dropout	0.656	0.450	0.309	0.211	0.209	0.512	0.512	**14.26**
Results on Flickr30k								
RMSprop	0.644	0.422	0.279	0.184	0.180	0.372	0.476	17.80
RMSprop + Dropout	0.656	0.435	0.295	0.200	0.185	0.396	0.481	18.05
RMSprop + Gal's Dropout	0.636	0.429	0.290	0.197	0.190	0.408	0.480	17.27
pSGLD	0.657	0.438	0.300	0.206	**0.192**	**0.421**	**0.490**	**15.61**
pSGLD + Dropout	**0.666**	**0.448**	**0.308**	**0.209**	0.189	0.419	0.487	17.05

with or without dropout. In addition to (naive) dropout, we further compare pSGLD with the *Gal's dropout*, recently proposed in Gal and Ghahramani (2016b), which is shown to be applicable to recurrent layers. Consistent with the results in the basic language modeling, pSGLD yields improved performance compared to RMSprop. For example, pSGLD provides 2.7 BLEU-4 score improvement over RMSprop on the Flickr8k dataset. By comparing pSGLD with RMSprop with dropout, we conclude that pSGLD exhibits better regularization ability than dropout on these two datasets.

Apart from modeling weight uncertainty, different samples from our algorithm may capture different aspects of the input image. An example with two images is shown in Fig. 4, where 2 randomly chosen model samples are considered for each image. For each model sample, the top 3 generated captions are presented. We use the beam search approach (Vinyals et al., 2015) to generate captions, with a beam of size 5. In Fig. 4, the two samples for the first image mainly differ in the color and activity of the dog, *e.g.*, "tan" or "yellow", "playing" or "running", whereas for the second image, the two samples reflect different understanding of the image content.

5.3 Sentence Classification

We study the task of sentence classification on 5 datasets as in Kiros et al. (2015): *MR* (Pang and Lee, 2005), *CR* (Hu and Liu, 2004), *SUBJ* (Pang and Lee, 2004), *MPQA* (Wiebe et al., 2005) and *TREC* (Li and Roth, 2002). A single-layer bidirectional LSTM is employed with the number of hidden units set to 400. Table 6 shows the testing classification errors. 10-fold cross-validation is used for evaluation on the first 4 datasets, while TREC has a pre-defined training/test split, and we run each algorithm 10 times on TREC. The combination of pSGLD and dropout consistently provides the lowest errors.

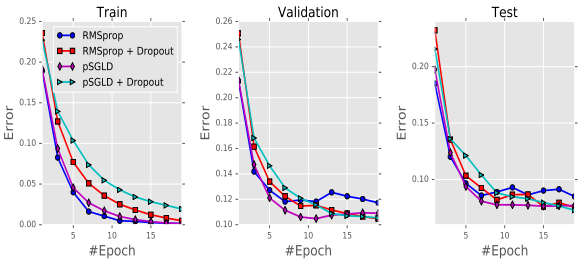


Figure 5: Learning curves on TREC dataset.

In the following, we focus on the analysis of TREC. Each sentence of TREC is a question, and the goal is to decide which topic type the question is most related to: *location*, *human*, *numeric*, *abbreviation*, *entity* or *description*. Fig. 5 plots the learning curves of different algorithms on the training, validation and testing sets of the TREC dataset. pSGLD and dropout have similar behavior: they explore the parameter space during learning, and thus coverge slower than RMSprop on the training dataset. However, the learned uncertainty alleviates overfitting and results in lower errors on the validation and testing datasets.

To further study the Bayesian nature of the proposed approach, in Fig. 6 we choose two testing sentences with high uncertainty (*i.e.*, standard derivation in prediction) from the TREC dataset. Interestingly, after embedding to 2d-space with tSNE (Van der Maaten and Hinton, 2008), the two

Table 6: Sentence classification errors on five benchmark datasets.

Methods	MR	CR	SUBJ	MPQA	TREC
RMSprop	21.86±1.19	20.20±1.35	8.13±1.19	10.60±1.28	8.14±0.63
RMSprop + Dropout	20.52±0.99	19.57±1.79	7.24±0.86	10.66±0.74	7.48±0.47
RMSprop + Gal's Dropout	20.22±1.12	19.29±1.93	7.52±1.17	10.59±1.12	7.34±0.66
pSGLD	20.36±0.85	18.72±1.28	7.00±0.89	10.54±0.99	7.48±0.82
pSGLD + Dropout	**19.33**±1.10	**18.18**±1.32	**6.61**±1.06	**10.22**±0.89	**6.88**±0.65

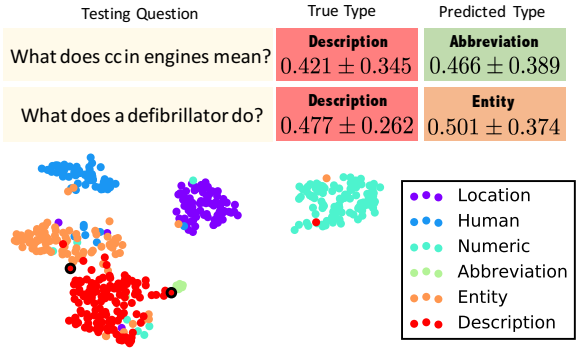


Figure 6: Visualization. Top two rows show selected ambiguous sentences, which correspond to the points with black circles in tSNE visualization of the testing dataset.

sentences correspond to points lying on the boundary of different classes. We use 20 model samples to estimate the prediction mean and standard derivation on the true type and predicted type. The classifier yields higher probability on the wrong types, associated with higher standard derivations. One can leverage the uncertainty information to make decisions: either manually make a human judgement when uncertainty is high, or automatically choose the one with lower standard derivations when both types exhibits similar prediction means. A more rigorous usage of the uncertainty information is left as future work.

5.4 Discussion

Ablation Study We investigate the effectivenss of each module in the proposed algorithm in Table 7 on two datasets: TREC and PTB. The small network size is used on PTB. Let M_1 denote only gradient noise, and M_2 denote only model averaging. As can be seen, The last sample in pSGLD (M_1) does not necessarily bring better results than RMSprop, but the model averaging over the samples of pSGLD indeed provide better results than model averaging of RMSprop (M_2). This indicates that both gradient noise and model averaging are crucial for good performance in pSGLD.

Table 7: Ablation study on TREC and PTB.

Datasets	RMSprop	M_1	M_2	pSGLD
TREC	8.14	8.34	7.54	7.48
PTB	120.45	122.14	114.86	109.44

Table 8: Running time on Flickr30k in seconds.

Stages	pSGLD	RMSprop+Dropout
Training	20324	12578
Testing	7047	1311

Running Time We report the training and testing time for image captioning on the Flickr30k dataset in Table 8. For pSGLD, the extra cost in training comes from adding gradient noise, and the extra cost in testing comes from model averaging. However, the cost in model averaging can be alleviated via the distillation methods: learning a single neural network that approximates the results of either a large model or an ensemble of models (Korattikara et al., 2015; Kim and Rush, 2016; Kuncoro et al., 2016). The idea can be incorporated with our SG-MCMC technique to achieve the same goal, which we leave for our future work.

6 Conclusion

We propose a scalable Bayesian learning framework using SG-MCMC, to model weight uncertainty in recurrent neural networks. The learning framework is tested on several tasks, including language models, image caption generation and sentence classification. Our algorithm outperforms stochastic optimization algorithms, indicating the importance of learning weight uncertainty in recurrent neural networks. Our algorithm requires little additional computational overhead in training, and multiple times of forward-passing for model averaging in testing.

Acknowledgments This research was supported by ARO, DARPA, DOE, NGA, ONR and NSF. We acknowledge Wenlin Wang for the code on language modeling experiment.

References

S. Banerjee and A. Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL workshop*.

J. Bayer, C. Osendorfer, D. Korhammer, N. Chen, S. Urban, and P. van der Smagt. 2013. On fast dropout and its applicability to recurrent networks. *arXiv:1311.0701* .

T. Bluche, C. Kermorvant, and J. Louradour. 2015. Where to apply dropout in recurrent neural networks for handwriting recognition? In *ICDAR*.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. 2015. Weight uncertainty in neural networks. In *ICML*.

L Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*.

C. Chen, D. Carlson, Z. Gan, C. Li, and L. Carin. 2016. Bridging the gap between stochastic gradient MCMC and stochastic optimization. In *AISTATS*.

C. Chen, N. Ding, and L. Carin. 2015a. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *NIPS*.

T. Chen, E. B. Fox, and C. Guestrin. 2014. Stochastic gradient Hamiltonian Monte Carlo. In *ICML*.

X. Chen, H. Fang, T. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. 2015b. Microsoft coco captions: Data collection and evaluation server. *arXiv:1504.00325* .

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555* .

N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven. 2014. Bayesian sampling using stochastic gradient thermostats. In *NIPS*.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* .

Y. Gal and Z. Ghahramani. 2016a. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *ICML*.

Y. Gal and Z. Ghahramani. 2016b. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*.

Z. Gan, C. Chen, R. Henao, D. Carlson, and L. Carin. 2015. Scalable deep poisson factor analysis for topic modeling. In *ICML*.

Z. Gan, C. Gan, X. He, Y. Pu, K. Tran, J. Gao, L. Carin, and L. Deng. 2017. Semantic compositional networks for visual captioning. In *CVPR*.

A. Graves. 2011. Practical variational inference for neural networks. In *NIPS*.

A. Graves. 2013. Generating sequences with recurrent neural networks. *arXiv:1308.0850* .

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

J. M. Hernández-Lobato and R. P. Adams. 2015. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*.

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

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. In *Neural computation*.

M. Hodosh, P. Young, and J. Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *JAIR* .

M. Hu and B. Liu. 2004. Mining and summarizing customer reviews. *SIGKDD* .

A. Karpathy, J. Johnson, and L. Fei-Fei. 2016. Visualizing and understanding recurrent networks. In *ICLR Workshop*.

Y. Kim and A. M. Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

D. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

D. Kingma, T. Salimans, and M. Welling. 2015. Variational dropout and the local reparameterization trick. In *NIPS*.

R. Kiros, Y. Zhu, R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. 2015. Skip-thought vectors. In *NIPS*.

A. Korattikara, V. Rathod, K. Murphy, and M. Welling. 2015. Bayesian dark knowledge. In *NIPS*.

A. Kuncoro, M. Ballesteros, L. Kong, C. Dyer, and N. A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. In *EMNLP*.

C. Li, C. Chen, D. Carlson, and L. Carin. 2016a. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *AAAI*.

C. Li, C. Chen, K. Fan, and L. Carin. 2016b. High-order stochastic gradient thermostats for Bayesian learning of deep models. In *AAAI*.

C. Li, A. Stevens, C. Chen, Y. Pu, Z. Gan, and L. Carin. 2016c. Learning weight uncertainty with stochastic gradient mcmc for shape classification. In *CVPR*.

X. Li and D. Roth. 2002. Learning question classifiers. *ACL* .

C. Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL workshop*.

D. J. C. MacKay. 1992. A practical Bayesian framework for backpropagation networks. In *Neural computation*.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* .

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

T. Moon, H. Choi, H. Lee, and I. Song. 2015. Rnndrop: A novel dropout for rnns in asr. *ASRU* .

R. M. Neal. 1995. *Bayesian learning for neural networks*. PhD thesis, University of Toronto.

A. Neelakantan, L. Vilnis, Q. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens. 2016. Adding gradient noise improves learning for very deep networks. In *ICLR workshop*.

M. Pachitariu and M. Sahani. 2013. Regularization and nonlinearities for neural language models: when are they needed? *arXiv:1301.5650* .

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *ACL* .

B. Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *ACL* .

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

R. Pascanu, T. Mikolov, and Y. Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*.

V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *ICFHR*.

H. Robbins and S. Monro. 1951. A stochastic approximation method. In *The annals of mathematical statistics*.

S. Semeniuta, A. Severyn, and E. Barth. 2016. Recurrent dropout without memory loss. *arXiv:1603.05118* .

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR* .

I. Sutskever, J. Martens, and G. E. Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.

Y. W. Teh, A. H. Thiéry, and S. J. Vollmer. 2016. Consistency and fluctuations for stochastic gradient Langevin dynamics. *JMLR* .

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv:1605.02688* .

T. Tieleman and G. Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning* .

L. Van der Maaten and G. E. Hinton. 2008. Visualizing data using t-SNE. *JMLR* .

R. Vedantam, C. L. Zitnick, and D. Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*.

L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. 2013. Regularization of neural networks using DropConnect. In *ICML*.

S. Wang and C. Manning. 2013. Fast Dropout training. In *ICML*.

M. Welling and Y. W. Teh. 2011. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*.

P. Werbos. 1990. Backpropagation through time: what it does and how to do it. In *Proceedings of the IEEE*.

J. Wiebe, T. Wilson, and C. Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation* .

P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL* .

W. Zaremba, I. Sutskever, and O. Vinyals. 2014. Recurrent neural network regularization. *arXiv:1409.2329* .

Learning attention for historical text normalization by learning to pronounce

Marcel Bollmann
Department of Linguistics
Ruhr-Universität Bochum
Germany
bollmann@linguistics.rub.de

Joachim Bingel
Dept. of Computer Science
University of Copenhagen
Denmark
bingel@di.ku.dk

Anders Søgaard
Dept. of Computer Science
University of Copenhagen
Denmark
soegaard@di.ku.dk

Abstract

Automated processing of historical texts often relies on pre-normalization to modern word forms. Training encoder-decoder architectures to solve such problems typically requires a lot of training data, which is not available for the named task. We address this problem by using several novel encoder-decoder architectures, including a multi-task learning (MTL) architecture using a grapheme-to-phoneme dictionary as auxiliary data, pushing the state-of-the-art by an absolute 2% increase in performance. We analyze the induced models across 44 different texts from Early New High German. Interestingly, we observe that, as previously conjectured, multi-task learning can learn to focus attention during decoding, in ways remarkably similar to recently proposed attention mechanisms. This, we believe, is an important step toward understanding how MTL works.

1 Introduction

There is a growing interest in automated processing of historical documents, as evidenced by the growing field of digital humanities and the increasing number of digitally available collections of historical documents. A common approach to deal with the high amount of variance often found in this type of data is to perform spelling normalization (Piotrowski, 2012), which is the mapping of historical spelling variants to standardized/modernized forms (e.g. *vnd* → *und* 'and').

Training data for supervised learning of historical text normalization is typically scarce, making it a challenging task for neural architectures, which typically require large amounts of labeled data. Nevertheless, we explore framing the spelling normalization task as a character-based sequence-to-sequence transduction problem, and use encoder–decoder recurrent neural networks (RNNs) to induce our transduction models. This is similar to models that have been proposed for neural machine translation (e.g., Cho et al. (2014)), so essentially, our approach could also be considered a specific case of character-based neural machine translation.

By basing our model on individual characters as input, we keep the vocabulary size small, which in turn reduces the model's complexity and the amount of data required to train it effectively. Using an encoder–decoder architecture removes the need for an explicit character alignment between historical and modern wordforms. Furthermore, we explore using an auxiliary task for which data is more readily available, namely grapheme-to-phoneme mapping (word pronunciation), to regularize the induction of the normalization models.

We propose several architectures, including multi-task learning architectures taking advantage of the auxiliary data, and evaluate them across 44 small datasets from Early New High German.

Contributions Our contributions are as follows:

- We are, to the best of our knowledge, the first to propose and evaluate encoder-decoder architectures for historical text normalization.
- We evaluate several such architectures across 44 datasets of Early New High German.
- We show that such architectures benefit from bidirectional encoding, beam search, and attention.
- We also show that MTL with pronunciation as an auxiliary task improves the performance of architectures without attention.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 332–344
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1031

- We analyze the above architectures and show that the MTL architecture *learns* attention from the auxiliary task, making the attention mechanism largely redundant.

- We make our implementation publicly available at https://bitbucket.org/mbollmann/acl2017.

In sum, we both push the state-of-the-art in historical text normalization and present an analysis that, we believe, brings us a step further in understanding the benefits of multi-task learning.

2 Datasets

Normalization For the normalization task, we use a total of 44 texts from the Anselm corpus (Dipper and Schultz-Balluff, 2013) of Early New High German.[1] The corpus is a collection of manuscripts and prints of the same core text, a religious treatise. Although the texts are semi-parallel and share some vocabulary, they were written in different time periods (between the 14th and 16th century) as well as different dialectal regions, and show quite diverse spelling characteristics. For example, the modern German word *Frau* 'woman' can be spelled as *fraw/vraw* (Me), *frawe* (N2), *frauwe* (St), *fraüwe* (B2), *frow* (Stu), *vrowe* (Ka), *vorwe* (Sa), or *vrouwe* (B), among others.[2]

All texts in the Anselm corpus are manually annotated with gold-standard normalizations following guidelines described in Krasselt et al. (2015). For our experiments, we excluded texts from the corpus that are shorter than 4,000 tokens, as well as a few for which annotations were not yet available at the time of writing (mostly Low German and Dutch versions). Nonetheless, the remaining 44 texts are still quite short for machine-learning standards, ranging from about 4,200 to 13,200 tokens, with an average length of 7,350 tokens.

For all texts, we removed tokens that consisted solely of punctuation characters. We also lowercase all characters, since it helps keep the size of the vocabulary low, and uppercasing of words is usually not very consistent in historical texts. Tokenization was not an issue for pre-processing these texts, since modern token boundaries have already been marked by the transcribers.

[1] https://www.linguistics.rub.de/anselm/

[2] We refer to individual texts using the same internal IDs that are found in the Anselm corpus (cf. the website).

Grapheme-to-phoneme mappings We use learning to pronounce as our auxiliary task. This task consists of learning mappings from sequences of graphemes to the corresponding sequences of phonemes. We use the German part of the CELEX lexical database (Baayen et al., 1995), particularly the database of phonetic transcriptions of German wordforms. The database contains a total of 365,530 wordforms with transcriptions in DISC format, which assigns one character to each distinct phonological segment (including affricates and diphthongs). For example, the word *Jungfrau* 'virgin' is represented as `'jUN-frB`.

3 Model

3.1 Base model

We propose several architectures that are extensions of a base neural network architecture, closely following the sequence-to-sequence model proposed by Sutskever et al. (2014). It consists of the following:

- an embedding layer that maps one-hot input vectors to dense vectors;

- an encoder RNN that transforms the input sequence to an intermediate vector of fixed dimensionality;

- a decoder RNN whose hidden state is initialized with the intermediate vector, and which is fed the output prediction of one timestep as the input for the next one; and

- a final dense layer with a softmax activation which takes the decoder's output and generates a probability distribution over the output classes at each timestep.

For the encoder/decoder RNNs, we use long short-term memory units (LSTM) (Hochreiter and Schmidhuber, 1997). LSTMs are designed to allow recurrent networks to better learn long-term dependencies, and have proven advantageous to standard RNNs on many tasks. We found no significant advantage from stacking multiple LSTM layers for our task, so we use the simplest competitive model with only a single LSTM unit for both encoder and decoder.

By using this encoder–decoder model, we avoid the need to generate explicit alignments between the input and output sequences, which would bring up the question of how to deal with input/output

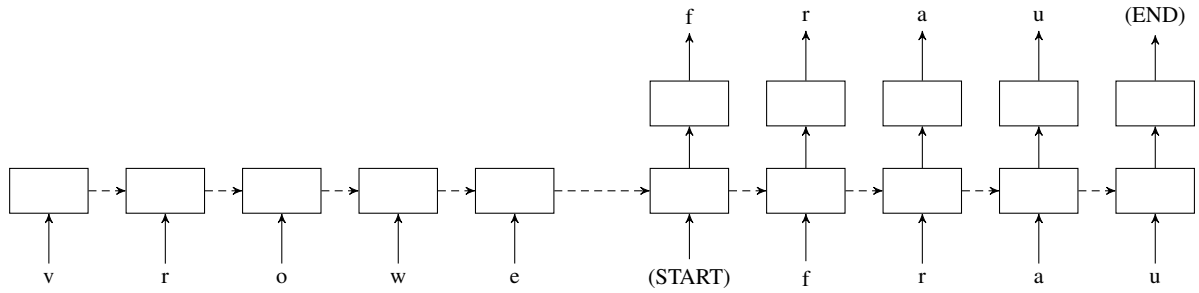


Figure 1: Flow diagram of the base model; left side is the encoder, right side the decoder, the latter of which has an additional prediction layer on top. Multi-task learning variants use two separate prediction layers for main/auxiliary tasks, while sharing the rest of the model. Embedding layers for the inputs are not explicitly shown.

pairs of different lengths. Another important property is that the model does not start to generate any output until it has seen the full input sequence, which in theory allows it to learn from any part of the input, without being restricted to fixed context windows. An example illustration of the unrolled network is shown in Fig. 1.

3.2 Training

During training, the encoder inputs are the historical wordforms, while the decoder inputs correspond to the correct modern target wordforms. We then train each model by minimizing the cross-entropy loss across all output characters; i.e., if $y = (y_1, ..., y_n)$ is the correct output word (as a list of one-hot vectors of output characters) and $\hat{y} = (\hat{y}_1, ..., \hat{y}_n)$ is the model's output, we minimize the mean loss $-\sum_{i=1}^{n} y_i \log \hat{y}_i$ over all training samples. For the optimization, we use the Adam algorithm (Kingma and Ba, 2015) with a learning rate of 0.003.

To reduce computational complexity, we also set a maximum word length of 14, and filter all training samples where either the input or output word is longer than 14 characters. This only affects 172 samples across the whole dataset, and is only done during training. In other words, we evaluate our models across all the test examples.

3.3 Decoding

For prediction, our base model generates output character sequences in a greedy fashion, selecting the character with the highest probability at each timestep. This works fairly well, but the greedy approach can yield suboptimal global picks, in which each individual character is sensibly derived from the input, but the overall word is nonsensical. We therefore also experiment with beam search decoding, setting the beam size to 5.

Finally, we also experiment with using a lexical filter during the decoding step. Here, before picking the next 5 most likely characters during beam search, we remove all characters that would lead to a string not covered by the lexicon. This is again intended to reduce the occurrence of nonsensical outputs. For the lexicon, we use all word forms from CELEX (cf. Sec. 2) plus the target word forms from the training set.[3]

3.4 Attention

In our base architecture, we assume that we can decode from a single vector encoding of the input sequence. This is a strong assumption, especially with long input sequences. Attention mechanisms give us more flexibility. The idea is that instead of encoding the entire input sequence into a fixed-length vector, we allow the decoder to "attend" to different parts of the input character sequence at each time step of the output generation. Importantly, we let the model learn what to attend to based on the input sequence and what it has produced so far.

Our implementation is identical to the decoder with soft attention described by Xu et al. (2015). If $a = (a_1, ..., a_n)$ is the encoder's output and h_t is the decoder's hidden state at timestep t, we first calculate a context vector $\hat{z}_t$ as a weighted combination of the output vectors a_i:

$$\hat{z}_t = \sum_{i=1}^{n} \alpha_i a_i \quad (1)$$

[3] We observe that due to this filtering, we cannot reach 2.25% of the targets in our test set, most of which are Latin word forms.

The weights α_i are derived by feeding the encoder's output and the decoder's hidden state from the previous timestep into a multilayer perceptron, called the attention model (f_{att}):

$$\alpha = softmax(f_{\text{att}}(a, h_{t-1})) \tag{2}$$

We then modify the decoder by conditioning its internal states not only on the previous hidden state h_{t-1} and the previously predicted output character y_{t-1}, but also on the context vector $\hat{z}_t$:

$$\begin{aligned} i_t &= \sigma(W_i[h_{t-1}, y_{t-1}, \hat{z}_t] + b_i) \\ f_t &= \sigma(W_f[h_{t-1}, y_{t-1}, \hat{z}_t] + b_f) \\ o_t &= \sigma(W_o[h_{t-1}, y_{t-1}, \hat{z}_t] + b_o) \\ g_t &= \tanh(W_g[h_{t-1}, y_{t-1}, \hat{z}_t] + b_g) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{3}$$

In Eq. 3, we follow the traditional LSTM description consisting of input gate i_t, forget gate f_t, output gate o_t, cell state c_t and hidden state h_t, where W and b are trainable parameters.

For all experiments including an attentional decoder, we use a *bi-directional encoder*, comprised of one LSTM layer that reads the input sequence normally and another LSTM layer that reads it backwards, and attend over the concatenated outputs of these two layers.

While a precise alignment of input and output sequences is sometimes difficult, most of the time the sequences align in a sequential order, which can be exploited by an attentional component.

3.5 Multi-task learning

Finally, we introduce a variant of the base architecture, with or without beam search, that does multi-task learning (Caruana, 1993). The multi-task architecture only differs from the base architecture in having two classifier functions at the outer layer, one for each of our two tasks. Our auxiliary task is to predict a sequence of phonemes as the correct pronunciation of an input sequence of graphemes. This choice is motivated by the relationship between phonology and orthography, in particular the observation that spelling variation often stems from phonological variation.

We train our multi-task learning architecture by alternating between the two tasks, sampling one instance of the auxiliary task for each training sample of the main task. We use the encoder-decoder to generate a corresponding output sequence, whether a modern word form or a pronunciation. Doing so, we suffer a loss with respect to the true output sequence and update the model parameters. The update for a sample from a specific task affects the parameters of corresponding classifier function, as well as all the parameters of the shared hidden layers.

3.6 Hyperparameters

We used a single manuscript (B) for manually evaluating and setting the hyperparameters. This manuscript is left out of the averages reported below. We believe that using a single manuscript for development, and using the same hyperparameters across *all* manuscripts, is more realistic, as we often do not have enough data in historical text normalization to reliably tune hyperparameters.

For the final evaluation, we set the size of the embedding and the recurrent LSTM layers to 128, applied a dropout of 0.3 to the input of each recurrent layer, and trained the model on mini-batches with 50 samples each for a total of 50 epochs (in the multi-task learning setup, mini-batches contain 50 samples of each task, and epochs are counted by the size of the training set for the main task only). All these parameters were set on the B manuscript alone.

3.7 Implementation

We implemented all of the models in Keras (Chollet, 2015). Any parameters not explicitly described here were left at their default values in Keras v1.0.8.

4 Evaluation

We split up each text into three parts, using 1,000 tokens each for a test set and a development set (that is not currently used), and the remainder of the text (between 2,000 and 11,000 tokens) for training. We then train and evaluate on each of the 43 texts (excluding the B text that was used for hyper-parameter tuning) individually.

Baselines We compare our architectures to several competitive baselines. Our first baseline is an averaged perceptron model trained to predict output character n-grams for each input character, after using Levenshtein alignment with generated segment distances (Wieling et al., 2009, Sec. 3.3) to align input and output characters. Our second baseline uses the same alignment, but trains a

		Avg. Accuracy
Norma		77.89%
Averaged perceptron		75.72%
Bi-LSTM tagger		79.91%
MTL bi-LSTM tagger		79.56%
Base model	GREEDY	78.91%
	BEAM	79.27%
	BEAM+FILTER	80.46%
	BEAM+FILTER+ATTENTION	**82.72%**
MTL model	GREEDY	80.64%
	BEAM	81.13%
	BEAM+FILTER	**82.76%**
	BEAM+FILTER+ATTENTION	82.02%

Table 1: Average word accuracy across 43 texts from the Anselm dataset, evaluated on the first 1,000 tokens of each text. Evaluation on the base encoder-decoder model (Sec. 3.1) with greedy search, beam search ($k = 5$) and/or lexical filtering (Sec. 3.3), with attentional decoder (Sec. 3.4), and the multi-task learning (MTL) model using grapheme-to-phoneme mappings (Sec. 3.5).

deep bi-LSTM sequential tagger, following Bollmann and Søgaard (2016). We evaluate this tagger using both standard and multi-task learning. Finally, we compare our model to the rule-based and Levenshtein-based algorithms provided by the Norma tool (Bollmann, 2012).[4]

4.1 Word accuracy

We use word-level accuracy as our evaluation metric. While we also measure character-level metrics, minor differences on character level can cause large differences in downstream applications, so we believe that perfectly matching the output sequences is more useful. Average scores across all 43 texts are presented in Table 1 (see Appendix A for individual scores).

We first see that almost all our encoder-decoder architectures perform significantly better than the four state-of-the-art baselines. All our architectures perform better than Norma and the averaged perceptron, and all the MTL architectures outperform Bollmann and Søgaard (2016).

We also see that beam search, filtering, and attention lead to cumulative gains in the context of the single-task architecture – with the best architecture outperforming the state-of-the-art by almost 3% in absolute terms.

For our multi-task architecture, we also observe gains when we add beam search and filtering, but importantly, adding attention does not help. In fact, attention hurts the performance of our multi-task architecture quite significantly. Also note that the multi-task architecture *without* attention performs on-par with the single-task architecture *with* attention.

We hypothesize that the reason for this pattern, which is not only observed in the average scores in Table 1, but also quite consistent across the individual results in Appendix A, is that our multi-task learning already learns how to focus attention.

This is the hypothesis that we will try to validate in Sec. 5: *That multi-task learning can induce strategies for focusing attention comparable to attention strategies for recurrent neural networks.*

Sample predictions A small selection of predictions from our models is shown in Table 2. They serve to illustrate the effects of the various settings; e.g., the base model with greedy search tends to produce more nonsense words (*ters*, *ünsget*) than the others. Using a lexical filter helps the most in this regard: the base model with filtering correctly normalizes *ergieng* to *erging* '(he) fared', while decoding without a filter produces the non-word *erbiggen*. Even for *herczenlichen* (modern *herzlichen* 'heartfelt'), where no model finds the correct target form, only the model with filtering produces a somewhat reasonable alternative (*herzgeliebtes* 'heartily loved').

In some cases (such as *gewarnet* 'warned'),

[4] https://github.com/comphist/norma

Input	Target	Base model				MTL model
		GREEDY	BEAM	B+F	B+F+A	B+F
ergieng	erging	erbiggen	erbiggen	erging	erging	erging
herczenlichen	herzlichen	herrgelichen	herzgelichen	herzgeliebtes	herzel	herzel
tewr	teuer	ters	terter	terme	teurer	der
iüngst	jüngst	ünsget	pingst	fingst	fingst	jüngst
gewarnet	gewarnt	prandet	prandert	pranget	gewarnt	gewarnt
dick	oft	oft	oft	oft	dicke	dicke

Table 2: Selected predictions from some of our models on the M4 text; B = BEAM, F = FILTER, A = ATTENTION.

only the models with attention or multi-task learning produce the correct normalization, but even when they are wrong, they often agree on the prediction (e.g. *dicke*, *herzel*). We will investigate this property further in Sec. 5.

4.2 Learned vector representations

To gain further insights into our model, we created t-SNE projections (Maaten and Hinton, 2008) of vector representations learned on the M4 text.

Fig. 2 shows the learned character embeddings. In the representations from the base model (Fig. 2a), characters that are often normalized to the same target character are indeed grouped closely together: e.g., historical <*v*> and <*u*> (and, to a smaller extent, <*f*>) are often used interchangeably in the M4 text. Note the wide separation of <*n*> and <*m*>, which is a feature of M4 that does not hold true for all of the texts, as these do not always display a clear distinction between nasals. On the other hand, the MTL model shows a better generalization of the training data (Fig. 2b): here, <*u*> is grouped closer to other vowel characters and far away from <*v*>/<*f*>. Also, <*n*> and <*m*> are now in close proximity.

We can also visualize the internal word representations that are produced by the encoder (Fig. 3). Here, we chose words that demonstrate the interchangeable use of <*u*> and <*v*>. Historical *vnd, vns, vmb* become modern *und, uns, um*, changing the <*v*> to <*u*>. However, the representation of *vmb* learned by the base model is closer to forms like *von, vor, uor*, all starting with <*v*> in the target normalization. In the MTL model, however, these examples are indeed clustered together.

5 Analysis: Multi-task learning helps focus attention

Table 1 shows that models which employ *either* an attention mechanism *or* multi-task learning obtain similar improvements in word accuracy. However, we observe a decline in word accuracy for models that *combine* multi-task learning with attention.

A possible interpretation of this counter-intuitive pattern might be that attention and MTL, to some degree, learn similar functions of the input data, a conjecture by Caruana (1998). We put this hypothesis to the test by closely investigating properties of the individual models below.

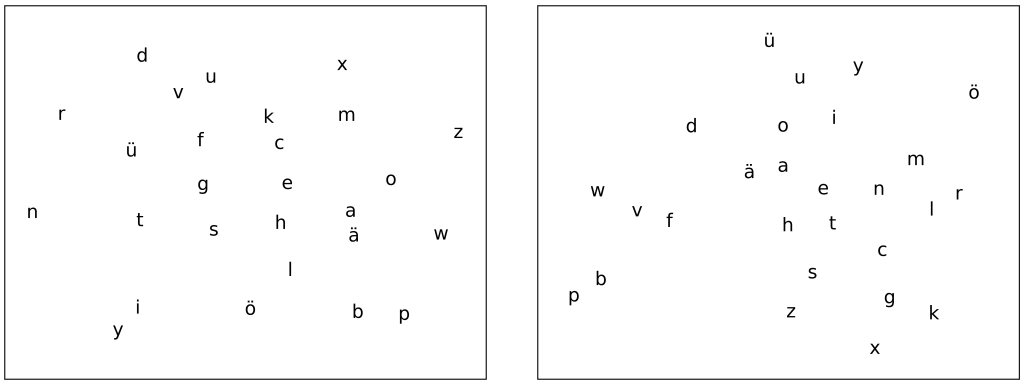
5.1 Model parameters

First, we are interested in the weight parameters of the final layer that transforms the decoder output to class probabilities. We consider these parameters for our standard encoder-decoder model and compare them to the weights that are learned by the attention and multi-task models, respectively.[5]

Note that hidden layer parameters are not necessarily comparable across models, but with a fixed seed, differences in parameters over a reference model *may* be (and are, in our case). With a fixed seed, and iterating over data points in the same order, it is conceivable the two non-baselines end up in roughly the same alternative local optimum (or at least take comparable routes).

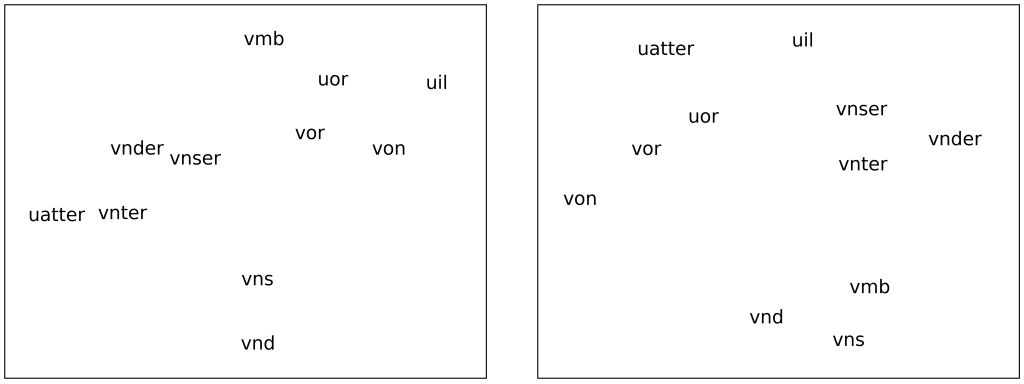
We observe that the weight differences between the standard and the attention model correlate with the differences between the standard and multi-task model by a Pearson's r of 0.346, averaged across datasets, with a standard deviation of 0.315; on individual datasets, correlation coefficient is as

[5]For the multi-task models, this analysis disregards those dimensions that do not correspond to classes in the main task.



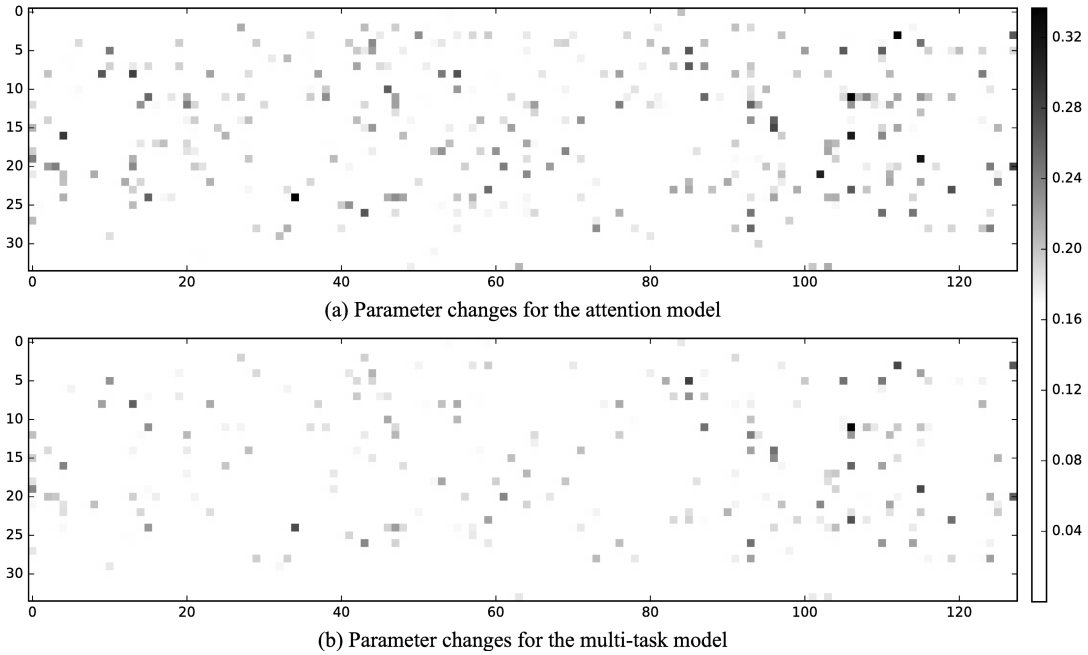
(a) Base model (b) Multi-task learning model

Figure 2: t-SNE projections (with perplexity 7) of character embeddings from models trained on M4



(a) Base model (b) Multi-task learning model

Figure 3: t-SNE projections (with perplexity 5) of the intermediate vectors produced by the encoder ("historical word embeddings"), from models trained on M4



(a) Parameter changes for the attention model

(b) Parameter changes for the multi-task model

Figure 4: Heat map of parameter differences in the final dense layer between (a) the plain and the attention model as well as (b) the plain and the multi-task model, when trained on the N4 manuscript. The changes correlate by $\rho = 0.959$.

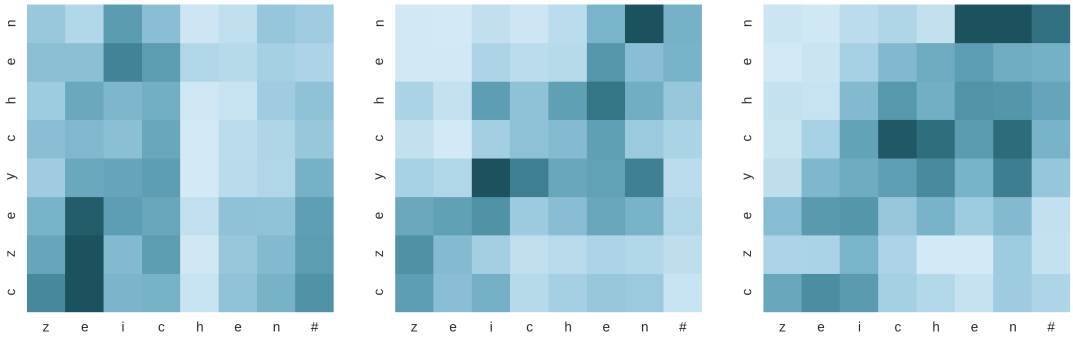


Figure 5: First-derivative saliency w.r.t. the input sequence, as calculated from the base model (left), the attentional model (center), and the MTL model (right). The scores for the attentional and the multi-task model correlate by $\rho = 0.615$, while the correlation of either one with the base model is $|\rho| < 0.12$.

high as 96. Figure 4 illustrates these highly parallel weight changes for the different models when trained on the N4 dataset.

5.2 Final output

Next, we compare the effect that employing either an attention mechanism or multi-task learning has on the actual output of our system. We find that out of the 210.9 word errors that the base model produces on average across all test sets (comprising 1,000 tokens each), attention resolves 47.7, while multi-task learning resolves an average of 45.4 errors. Crucially, the overlap of errors that are resolved by both the attention and the MTL model amounts to 27.7 on average.

Attention and multi-task also introduce new errors compared to the base model (26.6 and 29.5 per test set, respectively), and again we can observe a relatively high agreement of the models (11.8 word errors are introduced by both models).

Finally, the attention and multi-task models display a word-level agreement of κ=0.834 (Cohen's kappa), while either of these models is less strongly correlated with the base model (κ=0.817 for attention and κ=0.814 for multi-task learning).

5.3 Saliency analysis

Our last analysis regards the saliency of the input timesteps with respect to the predictions of our models. We follow Li et al. (2016) in calculating first-derivative saliency for given input/output pairs and compare the scores from the different models. The higher the saliency of an input timestep, the more important it is in determining the model's prediction at a given output timestep. Therefore, if two models produce similar saliency matrices for a given input/output pair, they have learned to focus on similar parts of the input during the prediction. Our hypothesis is that the attentional and the multi-task learning model should be more similar in terms of saliency scores than either of them compared to the base model.

Figure 5 shows a plot of the saliency matrices generated from the word pair *czeychen – zeichen* 'sign'. Here, the scores for the attentional and the MTL model indeed correlate by $\rho = 0.615$, while those for the base model do not correlate with either of them. A systematic analysis across 19,000 word pairs (where all models agree on the output) shows that this effect only holds for longer input sequences ($\geq$ 7 characters), with a mean $\rho = 0.303$ (± 0.177) for attentional vs. MTL model, while the base model correlates with either of them by $\rho < 0.21$.

6 Related Work

Many traditional approaches to spelling normalization of historical texts use edit distances or some form of character-level rewrite rules, handcrafted (Baron and Rayson, 2008) or learned automatically (Bollmann, 2013; Porta et al., 2013).

A more recent approach is based on character-based statistical machine translation applied to historical text (Pettersson et al., 2013; Sánchez-Martínez et al., 2013; Scherrer and Erjavec, 2013; Ljubešić et al., 2016) or dialectal data (Scherrer and Ljubešić, 2016). This is conceptually very similar to our approach, except that we substitute the classical SMT algorithms for neural networks. Indeed, our models can be seen as a form of character-based neural MT (Cho et al., 2014).

Neural networks have rarely been applied to

historical spelling normalization so far. Azawi et al. (2013) normalize old Bible text using bi-directional LSTMs with a layer that performs alignment between input and output wordforms. Bollmann and Søgaard (2016) also use bi-LSTMs to frame spelling normalization as a character-based sequence labelling task, performing character alignment as a preprocessing step.

Multi-task learning was shown to be effective for a variety of NLP tasks, such as POS tagging, chunking, named entity recognition (Collobert et al., 2011) or sentence compression (Klerke et al., 2016). It has also been used in encoder-decoder architectures, typically for machine translation (Dong et al., 2015; Luong et al., 2016), though so far not with attentional decoders.

7 Conclusion and Future Work

We presented an approach to historical spelling normalization using neural networks with an encoder-decoder architecture, and showed that it consistently outperforms several existing baselines. Encouragingly, our work proves to be fully competitive with the sequence-labeling approach by Bollmann and Søgaard (2016), without requiring a prior character alignment.

Specifically, we demonstrated the aptitude of multi-task learning to mitigate the shortage of training data for the named task. We included a multifaceted analysis of the effects that MTL introduces to our models and the resemblance that it bears to attention mechanisms. We believe that this analysis is a valuable contribution to the understanding of MTL approaches also beyond spelling normalization, and we are confident that our observations will stimulate further research into the relationship between MTL and attention.

Finally, many improvements to the presented approach are conceivable, most notably introducing some form of token context to the model. Currently, we only consider word forms in isolation, which is problematic for ambiguous cases (such as *jn*, which can normalize to *in* 'in' or *ihn* 'him') and conceivably makes the task harder for others. Reranking the predictions with a language model could be one possible way to improve on this. Ljubešić et al. (2016), for example, experiment with segment-based normalization, using a character-based SMT model with character input derived from segments (essentially, token n-grams) instead of single tokens, which also introduces context. Such an approach could also deal with the issue of tokenization differences between the historical and the modern text, which is another challenge often found in datasets of historical text.

Acknowledgments

Marcel Bollmann was supported by Deutsche Forschungsgemeinschaft (DFG), Grant DI 1558/4. This research is further supported by ERC Starting Grant LOWLANDS No. 313695, as well as by Trygfonden.

References

Mayce Al Azawi, Muhammad Zeshan Afzal, and Thomas M. Breuel. 2013. Normalizing historical orthography for OCR historical documents using LSTM. In *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*. ACM, pages 80–85. https://doi.org/10.1145/2501115.2501131.

R. Harald Baayen, Richard Piepenbrock, and Léon Gulikers. 1995. The CELEX lexical database (Release 2) (CD-ROM). Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA. https://catalog.ldc.upenn.edu/ldc96l14.

Alistair Baron and Paul Rayson. 2008. VARD 2: A tool for dealing with spelling variation in historical corpora. In *Proceedings of the Postgraduate Conference in Corpus Linguistics*. http://eprints.lancs.ac.uk/41666/.

Marcel Bollmann. 2012. (Semi-)automatic normalization of historical texts using distance measures and the Norma tool. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2)*. Lisbon, Portugal. https://www.linguistics.ruhr-uni-bochum.de/comphist/pub/acrh12.pdf.

Marcel Bollmann. 2013. Automatic normalization for linguistic annotation of historical language data. *Bochumer Linguistische Arbeitsberichte* 13. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:hebis:30:3-310764.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bi-directional lstms and multi-task learning. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*. Osaka, Japan. http://aclweb.org/anthology/C16-1013.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*. pages 41–48.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer, pages 95–133. http://dl.acm.org/citation.cfm?id=296635.296645.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*. Doha, Qatar, pages 103–111. http://dx.doi.org/10.3115/v1/W14-4012.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537. http://dl.acm.org/citation.cfm?id=1953048.2078186.

Stefanie Dipper and Simone Schultz-Balluff. 2013. The Anselm corpus: Methods and perspectives of a parallel aligned corpus. In *Proceedings of the NODALIDA Workshop on Computational Historical Linguistics*. http://www.ep.liu.se/ecp/087/003/ecp1387003.pdf.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1723–1732. https://doi.org/10.3115/v1/P15-1166.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)* ArXiv:1412.6980. http://arxiv.org/abs/1412.6980.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *Proceedings of NAACL-HLT 2016*. San Diego, CA, pages 1528–1533. http://dx.doi.org/10.18653/v1/N16-1179.

Julia Krasselt, Marcel Bollmann, Stefanie Dipper, and Florian Petran. 2015. Guidelines for normalizing historical German texts. *Bochumer Linguistische Arbeitsberichte* 15. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:hebis:30:3-419680.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 681–691. https://doi.org/10.18653/v1/N16-1082.

Nikola Ljubešić, Katja Zupan, Darja Fišer, and Tomaž Erjavec. 2016. Normalising Slovene data: historical texts vs. user-generated content. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS)*. Bochum, Germany, pages 146–155.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. *4th International Conference on Learning Representations (ICLR 2016)* https://arxiv.org/abs/1511.06114v4.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9:2579–2605. http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. 2013. An SMT approach to automatic annotation of historical text. In *Proceedings of the NODALIDA Workshop on Computational Historical Linguistics*. Oslo, Norway. http://www.ep.liu.se/ecp/087/005/ecp1387005.pdf.

Michael Piotrowski. 2012. *Natural Language Processing for Historical Texts*. Number 17 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool, San Rafael, CA. http://dx.doi.org/10.2200/s00436ed1v01y201207hlt017.

Jordi Porta, José-Luis Sancho, and Javier Gómez. 2013. Edit transducers for spelling variation in Old Spanish. In *Proceedings of the NODALIDA Workshop on Computational Historical Linguistics*. Oslo, Norway. http://www.ep.liu.se/ecp/087/006/ecp1387006.pdf.

Yves Scherrer and Tomaž Erjavec. 2013. Modernizing historical Slovene words with character-based SMT. In *Proceedings of the 4th Biennial Workshop on Balto-Slavic Natural Language Processing*. Sofia, Bulgaria. https://hal.inria.fr/hal-00838575.

Yves Scherrer and Nikola Ljubešić. 2016. Automatic normalisation of the Swiss German ArchiMob corpus using character-level machine translation. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS)*. Bochum, Germany, pages 248–255. http://archive-ouverte.unige.ch/unige:90846.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS 2014)*. 27, pages 3104–3112.

Felipe Sánchez-Martínez, Isabel Martínez-Sempere, Xavier Ivars-Ribes, and Rafael C. Carrasco. 2013. An open diachronic corpus of historical Spanish:

annotation criteria and automatic modernisation of spelling. http://arxiv.org/abs/1306.3692v1.

Martijn Wieling, Jelena Prokić, and John Nerbonne. 2009. Evaluating the pairwise string alignment of pronunciations. In *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education (LaTeCH – SHELT&R 2009)*. Athens, Greece, pages 26–34. http://dl.acm.org/citation.cfm?id=1642049.1642053.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *JMLR Workshop and Conference Proceedings: Proceedings of the 32nd International Conference on Machine Learning*. Lille, France, volume 37, pages 2048–2057. http://proceedings.mlr.press/v37/xuc15.pdf.

A Supplementary Material

For interested parties, we provide our full evaluation results for each single text in our dataset. Table 3 shows token counts, a rough classification of each text's dialectal region, and the results for the baseline methods. Table 4 presents the full results for our encoder-decoder models.

ID	Region	Tokens	Norma	Avg. Perc.	Bi-LSTM Tagger	
					BASE	MTL
B	East Central	4,718	79.60%	76.30%	79.20%	78.82%
D3	East Central	5,704	79.70%	77.20%	80.10%	81.62%
H	East Central	8,427	83.00%	78.60%	85.00%	84.32%
B2	West Central	9,145	76.20%	74.60%	82.00%	80.12%
KÄ1492	West Central	7,332	78.40%	74.80%	81.60%	80.82%
KJ1499	West Central	7,330	77.00%	73.50%	84.50%	80.22%
N1500	West Central	7,272	77.60%	72.70%	79.00%	78.52%
N1509	West Central	7,418	78.40%	74.30%	80.80%	80.02%
N1514	West Central	7,412	78.50%	72.20%	79.00%	79.62%
St	West Central	7,407	73.30%	70.30%	75.50%	73.03%
D4	Upper/Central	5,806	76.10%	72.40%	76.50%	76.62%
N4	Upper	8,593	79.30%	80.00%	81.80%	82.52%
s1496/97	Upper	5,840	81.20%	77.70%	83.00%	82.62%
B3	East Upper	6,222	82.30%	79.50%	81.50%	83.02%
Hk	East Upper	8,690	79.10%	78.20%	80.90%	79.52%
M	East Upper	8,700	75.20%	72.80%	83.90%	82.72%
M2	East Upper	8,729	76.30%	75.10%	76.70%	79.32%
M3	East Upper	7,929	79.20%	77.30%	80.40%	81.52%
M5	East Upper	4,705	81.60%	76.40%	77.70%	76.92%
M6	East Upper	4,632	74.90%	73.70%	75.20%	75.72%
M9	East Upper	4,739	81.00%	79.00%	80.40%	79.32%
M10	East Upper	4,379	77.20%	76.00%	75.10%	75.92%
Me	East Upper	4,560	80.20%	76.90%	80.30%	79.12%
Sb	East Upper	7,218	79.60%	75.70%	80.00%	80.12%
T	East Upper	8,678	76.00%	73.40%	75.80%	73.43%
W	East Upper	8,217	77.60%	78.20%	81.40%	80.72%
We	East Upper	6,661	82.70%	78.60%	81.50%	82.22%
Ba	North Upper	5,934	79.10%	80.20%	80.70%	80.02%
Ba2	North Upper	5,953	80.70%	78.10%	82.50%	82.12%
M4	North Upper	8,574	76.70%	75.70%	79.40%	79.32%
M7	North Upper	4,638	78.60%	75.60%	78.20%	77.42%
M8	North Upper	8,275	79.30%	78.20%	81.10%	80.02%
n	North Upper	9,191	79.80%	81.90%	84.40%	84.62%
N	North Upper	13,285	74.00%	71.70%	79.00%	79.42%
N2	North Upper	7,058	82.80%	80.30%	84.30%	81.72%
N3	North Upper	4,192	78.10%	76.40%	77.60%	77.12%
Be	West Upper	8,203	74.90%	75.30%	78.80%	77.52%
Ka	West Upper	12,641	72.80%	75.40%	80.10%	81.62%
SG	West Upper	7,838	79.70%	78.00%	81.70%	81.12%
Sa	West Upper	8,668	71.50%	71.90%	76.10%	74.93%
Sa2	West Upper	8,834	77.60%	73.50%	79.50%	79.72%
St2	West Upper	8,686	72.80%	73.20%	78.20%	79.92%
Stu	West Upper	8,011	78.00%	76.50%	79.40%	79.62%
Le	Dutch	7,087	71.30%	65.00%	75.60%	75.12%
Average (-B)		7,353	77.89%	76.30%	79.91%	79.56%

Table 3: Word accuracy on the Anselm dataset, evaluated on the first 1,000 tokens, using the baseline models (cf. Sec. 4): the Norma tool (Bollmann, 2012), an averaged perceptron model, and a deep bi-LSTM sequential tagger (Bollmann and Søgaard, 2016).

ID	Base model				Multi-task learning model			
	G	B	B+F	B+F+A	G	B	B+F	B+F+A
B	76.90%	77.30%	78.40%	**82.70%**	77.70%	79.50%	81.70%	80.10%
D3	81.50%	81.60%	82.70%	**83.20%**	81.10%	81.70%	82.90%	**83.20%**
H	82.60%	82.90%	84.50%	**87.40%**	85.00%	85.80%	86.60%	85.20%
B2	81.00%	81.20%	82.40%	**83.40%**	80.00%	80.40%	82.70%	83.00%
KÄ1492	83.00%	83.40%	83.60%	84.00%	83.40%	83.70%	**85.10%**	84.90%
KJ1499	81.30%	81.30%	82.00%	**84.60%**	84.00%	84.00%	83.80%	82.50%
N1500	79.50%	80.30%	81.30%	**84.00%**	82.20%	82.50%	83.60%	82.30%
N1509	82.10%	82.40%	83.10%	**85.00%**	82.80%	83.50%	84.50%	82.80%
N1514	80.40%	80.50%	81.10%	83.40%	82.30%	82.80%	**84.20%**	83.10%
St	74.60%	74.60%	76.40%	79.70%	77.60%	77.80%	**80.20%**	77.70%
D4	77.90%	77.20%	79.00%	81.40%	77.00%	77.90%	**81.50%**	79.90%
N4	82.10%	82.30%	82.90%	**84.80%**	83.10%	83.00%	84.40%	84.00%
s1496/97	80.40%	80.10%	81.10%	82.10%	82.30%	82.50%	**85.20%**	83.90%
B3	80.80%	81.20%	82.20%	**85.20%**	82.70%	83.30%	84.80%	84.50%
Hk	77.30%	79.00%	79.40%	82.90%	80.30%	80.40%	81.20%	**83.70%**
M	81.40%	81.50%	82.60%	**85.00%**	82.90%	82.90%	82.70%	84.00%
M2	79.90%	80.50%	81.30%	81.80%	78.80%	77.80%	79.60%	**83.20%**
M3	81.00%	81.10%	82.00%	**83.70%**	82.80%	82.50%	83.50%	81.70%
M5	76.60%	77.10%	79.00%	**82.00%**	78.20%	78.20%	80.90%	81.50%
M6	72.70%	73.80%	75.20%	80.20%	77.30%	79.00%	**80.30%**	76.60%
M9	78.20%	78.50%	79.70%	**83.20%**	80.70%	79.70%	**83.20%**	79.60%
M10	72.00%	72.40%	73.20%	77.40%	75.70%	76.30%	**77.90%**	77.80%
Me	76.90%	76.50%	78.50%	**81.30%**	77.30%	79.20%	81.00%	77.40%
Sb	78.80%	79.10%	81.30%	81.40%	80.60%	81.00%	**84.00%**	82.90%
T	75.60%	75.10%	77.40%	**80.30%**	76.90%	78.00%	80.10%	79.50%
W	80.80%	81.20%	82.40%	81.90%	80.40%	81.60%	**84.40%**	**84.40%**
We	77.70%	80.00%	81.80%	**84.40%**	83.00%	82.70%	83.80%	83.30%
Ba	81.00%	80.60%	80.90%	**84.00%**	80.40%	81.00%	82.60%	81.60%
Ba2	79.70%	80.90%	82.00%	84.00%	82.60%	83.30%	**85.40%**	85.10%
M4	78.40%	78.60%	79.90%	81.00%	82.10%	82.20%	**82.60%**	80.50%
M7	74.70%	76.30%	78.60%	82.00%	79.60%	79.90%	**82.30%**	81.10%
M8	80.80%	81.30%	82.50%	**85.70%**	82.00%	82.50%	84.00%	85.40%
n	83.40%	83.40%	84.30%	86.00%	84.90%	86.30%	**88.00%**	85.50%
N	77.40%	77.40%	79.40%	79.80%	80.00%	80.30%	**81.50%**	80.30%
N2	82.00%	82.30%	83.80%	86.40%	82.40%	83.50%	**86.60%**	85.80%
N3	73.60%	74.00%	75.10%	**81.20%**	76.00%	76.30%	80.30%	78.70%
Be	75.50%	75.40%	77.60%	78.10%	78.10%	78.40%	79.70%	**80.20%**
Ka	81.20%	81.20%	81.80%	**83.90%**	81.20%	83.10%	83.40%	82.30%
SG	81.10%	81.90%	83.40%	**85.50%**	82.60%	84.30%	84.90%	83.00%
Sa	76.80%	77.20%	78.10%	**80.60%**	77.50%	78.00%	79.70%	79.90%
Sa2	78.90%	79.70%	80.70%	81.30%	79.70%	81.00%	**82.30%**	**82.30%**
St2	77.70%	78.10%	79.00%	**81.60%**	79.60%	79.70%	80.50%	80.60%
Stu	77.40%	77.30%	78.30%	82.50%	82.00%	81.80%	**83.10%**	82.90%
Le	77.40%	78.10%	78.20%	79.60%	78.30%	78.60%	**79.80%**	78.90%
Average (-B)	78.91%	79.27%	80.46%	**82.72%**	80.64%	81.13%	**82.76%**	82.02%

Table 4: Word accuracy on the Anselm dataset, evaluated on the first 1,000 tokens, using our base encoder-decoder model (Sec. 3) and the multi-task model. G = greedy decoding, B = beam-search decoding (with beam size 5), F = lexical filter, A = attentional model. Best results (also taking into account the baseline results from Table 3) shown in bold.

Deep Learning in Semantic Kernel Spaces

Danilo Croce **Simone Filice** **Giuseppe Castellucci** **Roberto Basili**
Department of Enterprise Engineering
University of Roma Tor Vergata,
Via del Politecnico 1, 00133, Rome, Italy
{croce,filice,basili}@info.uniroma2.it
castellucci@ing.uniroma2.it

Abstract

Kernel methods enable the direct usage of structured representations of textual data during language learning and inference tasks. Expressive kernels, such as Tree Kernels, achieve excellent performance in NLP. On the other side, deep neural networks have been demonstrated effective in automatically learning feature representations during training. However, their input is tensor data, i.e., they cannot manage rich structured information. In this paper, we show that expressive kernels and deep neural networks can be combined in a common framework in order to (*i*) explicitly model structured information and (*ii*) learn non-linear decision functions. We show that the input layer of a deep architecture can be pre-trained through the application of the Nyström low-rank approximation of kernel spaces. The resulting "kernelized" neural network achieves state-of-the-art accuracy in three different tasks.

1 Introduction

Learning for Natural Language Processing (NLP) requires to more or less explicitly account for trees or graphs to express syntactic and semantic information. A straightforward modeling of such information has been obtained in statistical language learning with Tree Kernels (TKs) (Collins and Duffy, 2001), or by means of structured neural models (Hochreiter and Schmidhuber, 1997; Socher et al., 2013). In particular, kernel-based methods (Shawe-Taylor and Cristianini, 2004) have been largely applied in language processing for alleviating the need of complex activities of manual feature engineering (e.g., (Moschitti et al., 2008)). Although *ad-hoc* features are adopted by many successful approaches to language learning (e.g., (Gildea and Jurafsky, 2002)), kernels provide a natural way to capture textual generalizations directly operating over (possibly complex) linguistic structures. Sequence (Cancedda et al., 2003) or tree kernels (Collins and Duffy, 2001) are of particular interest as the feature space they implicitly generate reflects linguistic patterns. On the other hand, Recursive Neural Networks (Socher et al., 2013) have been shown to learn dense feature representations of the nodes in a structure, thus exploiting similarities between nodes and sub-trees. Also, Long-Short Term Memory (Hochreiter and Schmidhuber, 1997) networks build intermediate representations of sequences, resulting in similarity estimates over sequences and their inner sub-sequences.

While such methods are highly effective and reach state-of-the-art results in many tasks, their adoption can be problematic. In kernel-based Support Vector Machine (SVM) the classification model corresponds to the set of support vectors (SVs) and weights justifying the maximal margin hyperplane: the classification cost crucially depends on their number, as classifying a new instance requires a kernel computation against all SVs, making their adoption in large data settings prohibitive. This scalability issue is evident in many NLP and Information Retrieval applications, such as in answer re-ranking in question answering (Severyn et al., 2013; Filice et al., 2016), where the number of SVs is typically very large. Improving the efficiency of kernel-based methods is a largely studied topic. The reduction of computational costs has been early designed by imposing a budget (Dekel and Singer, 2006; Wang and Vucetic, 2010), that is limiting the maximum number of SVs in a model. However, in complex tasks, such methods still require large budgets to reach

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 345–354
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1032

adequate accuracies. On the other hand, training complex neural networks is also difficult as no common design practice is established against complex data structures. In Levy et al. (2015), a careful analysis of neural word embedding models is carried out and the role of the hyper-parameter estimation is outlined. Different neural architectures result in the same performances, whenever optimal hyper-parameter tuning is applied. In this latter case, no significant difference is observed across different architectures, making the choice between different neural architectures a complex and empirical task.

A general approach to the large scale modeling of complex structures is a critical and open problem. A viable and general solution to this scalability issue is provided by the Nyström method (Williams and Seeger, 2001); it allows to approximate the Gram matrix of a kernel function and support the embedding of future input examples into a low-dimensional space. For example, if used over TKs, the Nyström projection corresponds to the embedding of any tree into a low-dimensional vector.

In this paper, we show that the Nyström based low-rank embedding of input examples can be used as the early layer of a deep feed-forward neural network. A standard NN back-propagation training can thus be applied to induce non-linear functions in the kernel space. The resulting deep architecture, called *Kernel-based Deep Architecture* (KDA), is a mathematically justified integration of expressive kernel functions and deep neural architectures, with several advantages: it (*i*) directly operates over complex non-tensor structures, e.g., trees, without any manual feature or architectural engineering, (*ii*) achieves a drastic reduction of the computational cost w.r.t. pure kernel methods, and (*iii*) exploits the non-linearity of NNs to produce accurate models. The experimental evaluation shows that the proposed approach achieves state-of-the-art results in three semantic inference tasks: Semantic Parsing, Question Classification and Community Question Answering.

In the rest of the paper, Section 2 surveys some of the investigated kernels. In Section 3 the Nyström methodology and KDA are presented. Experimental evaluations are described in Section 4. Finally, Section 5 derives the conclusions.

2 Kernel-based Semantic Inference

In almost all NLP tasks, explicit models of complex syntactic and semantic structures are required, such as in Paraphrase Detection: deciding whether two sentences are valid paraphrases involves learning grammatical rewriting rules, such as semantics preserving mappings among subtrees. Also in Question Answering, the syntactic information about input questions is crucial. While manual feature engineering is always possible, kernel methods on structured representations of data objects, e.g., sentences, have been largely applied. Since Collins and Duffy (2001), sentences can be modeled through their corresponding parse tree, and Tree Kernels (TKs) result in similarity metrics directly operating over tree fragments. Such kernels corresponds to dot products in the (implicit) feature space made of all possible tree fragments (Haussler, 1999). Notice that the number of tree fragments in a tree bank is combinatorial with the number of tree nodes and gives rise to billions of features, i.e., dimensions. In this high-dimensional space, kernel-based algorithms, such as SVMs, can implicitly learn robust prediction models (Shawe-Taylor and Cristianini, 2004), resulting in state-of-the-art approaches in several NLP tasks, e.g., Semantic Role Labeling (Moschitti et al., 2008), Question Classification (Croce et al., 2011) or Paraphrase Identification (Filice et al., 2015). As the feature space generated by the structural kernels depends on the input structures, different tree representations can be adopted to reflect more or less expressive syntactic/semantic feature spaces. While constituency parse trees have been early used (e.g., (Collins and Duffy, 2001)), dependency parse trees correspond to graph structures. TKs usually rely on their tree conversions, where grammatical edge labels corresponds to nodes. An expressive tree representation of dependency graphs is the Grammatical Relation Centered Tree (GRCT). As illustrated in Figure 1, PoS-Tags and grammatical functions correspond to nodes, dominating their associated lexicals.

Types of tree kernels. While a variety of TK functions have been studied, e.g., the *Partial Tree Kernel* (PTK) (Moschitti, 2006), the kernels used in this work model grammatical and semantic information, as triggered respectively by the dependency edge labels and lexical nodes. The latter is exploited through recent results in distributional models of lexical semantics, as proposed in

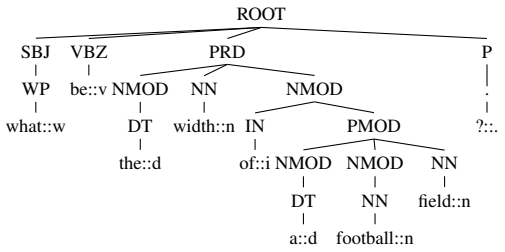


Figure 1: Grammatical Relation Centered Tree (GRCT) of *"What is the width of a football field?"*

word embedding methods (e.g., (Mikolov et al., 2013; Sahlgren, 2006). In particular, we adopt the *Smoothed Partial Tree Kernel* (SPTK) described in Croce et al. (2011): it extends the PTK formulation with a similarity function between lexical nodes in a GRCT, i.e., the cosine similarity between word vector representations based on word embeddings. We also use a further extension of the SPTK, called *Compositionally Smoothed Partial Tree Kernel* (CSPTK) (as in Annesi et al. (2014)). In CSPTK, the lexical information provided by the sentence words is propagated along the non-terminal nodes representing head-modifier dependencies. Figure 2 shows a compositionally-labeled tree, where the similarity function at the nodes can model lexical composition, i.e., capturing contextual information. For example, in the sentence, "*What instrument does Hendrix play?*", the role of the word *instrument* can be fully captured only if its composition with the verb *play* is considered. The CSPTK applies a composition function between nodes: while several algebraic functions can be adopted to compose two word vectors representing a head/modifier pair, here we refer to a simple additive function that assigns to each (h, m) pair the linear combination of the involved vectors, i.e., $(\mathbf{h}, \mathbf{m}) = A\mathbf{h} + B\mathbf{m}$: although simple and efficient, it actually produces very effective CSPTK functions.

Complexity. The training phase of an optimal maximum margin algorithm (such as SVM) requires a number of kernel operations that is more than linear (almost $\mathcal{O}(n^2)$) with respect to the number of training examples n, as discussed in Chang and Lin (2011). Also the classification phase depends on the size of the input dataset and the intrinsic complexity of the targeted task: classifying a new instance requires to evaluate the kernel function with respect to each support vector. For complex tasks, the number of selected support vectors tends to be very large, and using the resulting model can be impractical. This cost is also problematic as single kernel operations can be very expensive: the cost of evaluating the PTK on a single tree pair is almost linear in the number of nodes in the input trees, as shown in Moschitti (2006). When lexical semantics is considered, as in SPTKs and CSPTKs, it is more than linear in the number of nodes (Croce et al., 2011).

3 Deep Learning in Kernel Spaces

3.1 The Nyström method

Given an input training dataset D, a kernel $K(o_i, o_j)$ is a similarity function over $\mathcal{D}^2$ that corresponds to a dot product in the implicit kernel space, i.e., $K(o_i, o_j) = \Phi(o_i) \cdot \Phi(o_j)$. The advantage of kernels is that the projection function $\Phi(o) = \mathbf{x} \in \mathbb{R}^n$ is never explicitly computed (Shawe-Taylor and Cristianini, 2004). In fact, this operation may be prohibitive when the dimensionality n of the underlying kernel space is extremely large, as for Tree Kernels (Collins and Duffy, 2001). Kernel functions are used by learning algorithms, such as SVM, to operate only implicitly on instances in the kernel space, by never accessing their explicit definition. Let us apply the projection function Φ over all examples from $\mathcal{D}$ to derive representations, $\mathbf{x}$ denoting the rows of the matrix X. The Gram matrix can always be computed as $G = XX^\top$, with each single element corresponding to $G_{ij} = \Phi(o_i)\Phi(o_j) = K(o_i, o_j)$. The aim of the Nyström method is to derive a new low-dimensional embedding $\tilde{x}$ in a l-dimensional space, with $l \ll n$ so that $\tilde{G} = \tilde{X}\tilde{X}^\top$ and $\tilde{G} \approx G$. This is obtained by generating an approximation $\tilde{G}$ of G using a subset of l columns of the matrix, i.e., a selection of a subset $L \subset \mathcal{D}$ of the available examples, called *landmarks*. Suppose we randomly sample l columns of G, and let $C \in \mathbb{R}^{|D| \times l}$ be the matrix of these sampled columns. Then, we can rearrange the columns and rows of G and define $X = [X_1 \;\; X_2]$ such that:

$$G = XX^\top = \begin{bmatrix} W & X_1^\top X_2 \\ X_2^\top X_1 & X_2^\top X_2 \end{bmatrix}$$
$$\text{and} \quad C = \begin{bmatrix} W \\ X_2^\top X_1 \end{bmatrix} \tag{1}$$

where $W = X_1^\top X_1$, i.e., the subset of G that contains only landmarks. The Nyström approximation can be defined as:

$$G \approx \tilde{G} = CW^\dagger C^\top \tag{2}$$

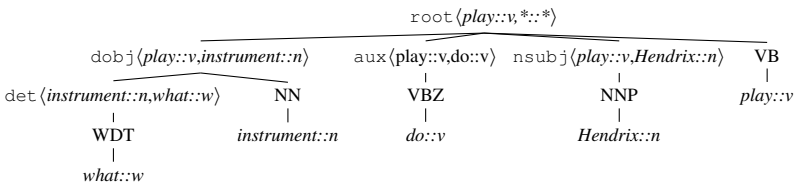


Figure 2: Compositional Grammatical Relation Centered Tree (CGRCT) of "*What instrument does Hendrix play?*"

where $W^\dagger$ denotes the Moore-Penrose inverse of W. The Singular Value Decomposition (SVD) is used to obtain $W^\dagger$ as it follows. First, W is decomposed so that $W = USV^\top$, where U and V are both orthogonal matrices, and S is a diagonal matrix containing the (non-zero) singular values of W on its diagonal. Since W is symmetric and positive definite $W = USU^\top$. Then $W^\dagger = US^{-1}U^\top = US^{-\frac{1}{2}}S^{-\frac{1}{2}}U^\top$ and the Equation 2 can be rewritten as

$$G \approx \tilde{G} = CUS^{-\frac{1}{2}}S^{-\frac{1}{2}}U^\top C^\top = (CUS^{-\frac{1}{2}})(CUS^{-\frac{1}{2}})^\top = \tilde{X}\tilde{X}^\top \quad (3)$$

Given an input example $o \in \mathcal{D}$, a new low-dimensional representation $\tilde{\mathbf{x}}$ can be thus determined by considering the corresponding item of C as

$$\tilde{\mathbf{x}} = \mathbf{c}US^{-\frac{1}{2}} \quad (4)$$

where $\mathbf{c}$ is the vector whose dimensions contain the evaluations of the kernel function between o and each landmark $o_j \in L$. Therefore, the method produces l-dimensional vectors. If k is the average number of basic operations required during a single kernel computation, the overall cost of a single projection is $\mathcal{O}(kl + l^2)$, where the first term corresponds to the cost of generating the vector $\mathbf{c}$, while the second term is needed for the matrix multiplications in Equation 4. Typically, the number of landmarks l ranges from hundreds to few thousands and, for complex kernels (such as Tree Kernels), the projection cost can be reduced to $\mathcal{O}(kl)$. Several policies have been defined to determine the best selection of landmarks to reduce the Gram Matrix approximation error. In this work the uniform sampling without replacement is adopted, as suggested by Kumar et al. (2012), where this policy has been theoretically and empirically shown to achieve results comparable with other (more complex) selection policies.

3.2 A Kernel-based Deep Architecture

The above introduced Nyström representation $\tilde{\mathbf{x}}$ of any input example o is linear and can be adopted to feed a neural network architecture. We assume a labeled dataset $\mathcal{L} = \{(o, y) \mid o \in \mathcal{D},\ y \in Y\}$ being available, where o refers to a generic instance and y is its associated class. In this Section, we define a Multi-Layer Perceptron (MLP) architecture, with a specific Nyström layer based on the Nyström embeddings of Eq. 4. We will refer to this architecture as Kernel-based Deep Architecture (KDA). KDA has an *input layer*, a *Nyström layer*, a possibly empty sequence of non-linear *hidden layers* and a final *classification layer*, which produces the output.

The *input* layer corresponds to the input vector $\mathbf{c}$, i.e., the row of the C matrix associated to an example o. Notice that, for adopting the KDA, the values of the matrix C should be all available. In the training stage, these values are in general cached. During the classification stage, the $\mathbf{c}$ vector corresponding to an example o is directly computed by l kernel computations between o and each one of the l landmarks.

The input layer is mapped to the *Nyström* layer, through the projection in Equation 4. Notice that the embedding provides also the proper weights, defined by $US^{-\frac{1}{2}}$, so that the mapping can be expressed through the Nyström matrix $H_{Ny} = US^{-\frac{1}{2}}$: it corresponds to a pre-trained stage derived through SVD, as discussed in Section 3.1. Equation 4 provides a static definition for H_{Ny} whose weights can be left invariant during the neural network training. However, the values of H_{Ny} can be made available for the standard back-propagation adjustments applied for training[1]. Formally, the low-dimensional embedding of an input example o, is $\tilde{\mathbf{x}} = \mathbf{c}\, H_{Ny} = \mathbf{c}\, US^{-\frac{1}{2}}$.

The resulting outcome $\tilde{\mathbf{x}}$ is the input to one or more non-linear *hidden* layers. Each t-th hidden layer is realized through a matrix $H_t \in \mathbb{R}^{h_{t-1} \times h_t}$ and a bias vector $\mathbf{b}_t \in \mathbb{R}^{1 \times h_t}$, whereas h_t denotes

[1] In our preliminary experiments, adjustments to the H_{Ny} matrix have been tested, but no significant effect was observed. Therefore, no adjustment has been used in any reported experiment, although more in depth exploration is needed on this aspect.

the desired hidden layer dimensionality. Clearly, given that $H_{Ny} \in \mathbb{R}^{l \times l}$, $h_0 = l$. The first hidden layer in fact receives in input $\tilde{\mathbf{x}} = \mathbf{c}H_{Ny}$, that corresponds to $t = 0$ layer input $\mathbf{x}_0 = \tilde{\mathbf{x}}$ and its computation is formally expressed by $\mathbf{x}_1 = f(\mathbf{x}_0 H_1 + \mathbf{b}_1)$, where f is a non-linear activation function. In general, the generic t-th layer is modeled as:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1} H_t + \mathbf{b}_t) \quad (5)$$

The final layer of KDA is the *classification layer*, realized through the output matrix H_O and the output bias vector $\mathbf{b}_O$. Their dimensionality depends on the dimensionality of the last hidden layer (called O_{-1}) and the number $|Y|$ of different classes, i.e., $H_O \in \mathbb{R}^{h_{O_{-1}} \times |Y|}$ and $\mathbf{b}_O \in \mathbb{R}^{1 \times |Y|}$, respectively. In particular, this layer computes a linear classification function with a softmax operator so that $\hat{y} = softmax(\mathbf{x}_{O_{-1}} H_O + \mathbf{b}_O)$.

In order to avoid over-fitting, two different regularization schemes are applied. First, the dropout is applied to the input $\mathbf{x}_t$ of each hidden layer ($t \geq 1$) and to the input $\mathbf{x}_{O_{-1}}$ of the final classifier. Second, a L_2 regularization is applied to the norm of each layer[2] H_t and H_O.

Finally, the KDA is trained by optimizing a loss function made of the sum of two factors: first, the cross-entropy function between the gold classes and the predicted ones; second the L_2 regularization, whose importance is regulated by a meta-parameter λ. The final loss function is thus

$$L(y, \hat{y}) = \sum_{(o,y) \in \mathcal{L}} y \, log(\hat{y}) + \lambda \sum_{H \in \{H_t\} \cup \{H_O\}} ||H||^2$$

where $\hat{y}$ are the softmax values computed by the network and y are the true one-hot encoding values associated with the example from the labeled training dataset $\mathcal{L}$.

4 Empirical Investigation

The proposed KDA has been applied adopting the same architecture but with different kernels to three NLP tasks, i.e., Question Classification, Community Question Answering, and Automatic Boundary Detection in Semantic Role Labeling. The Nyström projector has been implemented in the KeLP framework[3]. The neural network has been implemented in Tensorflow[4], with 2 hidden layers whose dimensionality corresponds to the number of involved Nyström landmarks. The *rectified linear unit* is the non-linear activation function in each layer. The dropout has been applied in each hidden layer and in the final classification layer. The values of the dropout parameter and the λ parameter of the L_2-regularization have been selected from a set of values via grid-search. The Adam optimizer with a learning rate of 0.001 has been applied to minimize the loss function, with a multi-epoch (500) training, each fed with batches of size 256. We adopted an early stop strategy, where the best model was selected according to the performance over the development set. Every performance measure is obtained against a specific sampling of the Nyström landmarks. Results averaged against 5 such samplings are always hereafter reported.

4.1 Question Classification

Question Classification (QC) is the task of mapping a question into a closed set of answer types in a Question Answering system. We used the UIUC dataset (Li and Roth, 2006), including a training and test set of $5,452$ and 500 questions, respectively, organized in 6 classes (like `ENTITY` or `HUMAN`). TKs resulted very effective, as shown in Croce et al. (2011); Annesi et al. (2014). In Annesi et al. (2014), QC is mapped into a One-vs-All multi-classification schema, where the CSPTK achieves state-of-the-art results of 95%: it acts directly over compositionally labeled trees without relying on any manually designed feature.

In order to proof the benefits of the KDA architecture, we generated Nyström representation of the CSPTK kernel function[5] with default parameters (i.e., $\mu = \lambda = 0.4$). The SVM formulation by Chang and Lin (2011), fed with the CSPTK (hereafter KSVM), is here adopted to determine the reachable upper bound in classification quality, i.e., a 95% of accuracy, at higher computational costs. It establishes the state-of-the-art over the UIUC dataset. The resulting model includes 3,873 support vectors: this corresponds to the number of kernel operations required to classify any input test question. The Nyström method based on a number of landmarks ranging from 100 to 1,000 is adopted for modeling input vectors in

[2] The input layer and the Nyström layer are not modified during the learning process, and they are not regularized.

[3] http://www.kelp-ml.org

[4] https://www.tensorflow.org/

[5] The lexical vectors used in the CSPTK are generated again using the Word2vec tool with a Skip-gram model.

the CSPTK kernel space. Results are reported in Table 1: computational saving refers to the percentage of avoided kernel computations with respect to the application of the KSVM to each test instance. To justify the need of the Neural Network, we compared the proposed KDA to an efficient linear SVM that is directly trained over the Nyström embeddings. This SVM implements the Dual Coordinate Descent method (Hsieh et al., 2008) and will be referred as SVM_{DCD}. We also measured the state-of-the-art Convolutional Neural Network[6] (CNN) of Kim (2014), achieving the remarkable accuracy of 93.6%. Notice that the linear classifier SVM_{DCD} operating over the approximated kernel space achieves the same classification quality of the CNN when just 1,000 landmarks are considered. KDA improves this results, achieving 94.3% accuracy even with fewer landmarks (only 600), showing the effectiveness of non-linear learning over the Nyström input. Although KSVM improves to 95%, KDA provides a saving of more than 84% kernel computations at classification time. This result is straightforward as it confirms that **linguistic information encoded in a tree is important in the analysis of questions** and can be used as a pre-training strategy. Figure 3 shows the accuracy curves according to various approximations of the kernel space, i.e., number of landmarks.

Table 1: Results in terms of Accuracy and saving in the Question Classification task

Model	#Land.	Accuracy	Saving
CNN	-	93.6%	-
KSVM	-	95.0%	0.0%
KDA (SVM_{DCD})	100	88.5% (84.1%)	97.4%
	200	92.2% (88.7%)	94.8%
	400	93.7% (91.6%)	89.7%
	600	**94.3%** (92.8%)	**84.5%**
	800	94.3% (93.0%)	79.3%
	1,000	94.2% (93.6%)	74.2%

4.2 Community Question-Answering

In the SemEval-2016 task 3, participants were asked to automatically provide good answers in a community question answering setting (Nakov et al., 2016). We focused on the subtask A: given a question and a large collection of question-comment threads created by a user community, the task consists in (re-)ranking the comments w.r.t. their utility in answering the question. Subtask A can be modeled as a binary classification problem, where instances are (question, comment) pairs. Each pair generates an example for a binary SVM, where the positive label is associated to a *good* comment and the negative label refers to *potentially useful* and *bad* comments. The classification score achieved over different (question, comment) pairs is used to sort instances and produce the final ranking over comments. The above setting results in a train and test dataset made of 20,340 and 3,270 examples, respectively. In Filice et al. (2016), a Kernel-based SVM classifier (KSVM) achieved state-of-the-art results by adopting a kernel combination that exploited (*i*) feature vectors containing linguistic similarities between the texts in a pair; (*ii*) shallow syntactic trees that encode the lexical and morpho-syntactic information shared between text pairs; (*iii*) feature vectors capturing task-specific information.

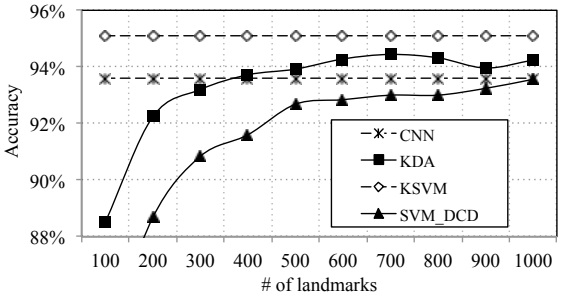


Figure 3: QC task - accuracy curves w.r.t. the number of landmarks.

Table 2: Results in terms of F_1 and savings in the Community Question Answering task

Model	#Land.	F_1	Saving
KSVM	-	0.644	0.0%
ConvKN	-	0.662	-
KDA (SVM_{DCD})	100	0.638 (0.596)	99.1%
	200	0.635 (0.627)	98.2%
	400	0.657 (0.637)	96.5%
	600	0.669 (0.645)	94.7%
	800	**0.680** (0.653)	**92.9%**
	1,000	0.674 (0.644)	91.2%

Such model includes 11,322 support vectors. We investigated the KDA architecture, trained by maximizing the F_1 measure, based on a Nyström layer initialized using the same kernel functions as KSVM. We varied the Nyström dimensions from 100 to 1,000 landmarks, i.e., a much lower number than the support vectors of KSVM.

Table 2 reports the results: very high F_1 scores

[6]The deep architecture presented in Kim (2014) outperforms several NN models, including the Recursive Neural Tensor Network or Tree-LSTM presented in (Socher et al., 2013; Tai et al., 2015) which presents a semantic compositionality model that exploits parse trees.

are observed with impressive savings in terms of kernel computations (between 91.2% and 99%). Also on the cQA task, the F_1 obtained by the SVM_{DCD} is significantly lower than the KDA one. Moreover, with 800 landmarks KDA achieves the remarkable results of 0.68 of F_1, that is the state-of-the-art against other convolutional systems, e.g., ConvKN (Barrón-Cedeño et al., 2016): this latter combines convolutional tree kernels with kernels operating on sentence embeddings generated by a convolutional neural network.

4.3 Argument Boundary Detection

Semantic Role Labeling (SRL) consists of the detection of the semantic arguments associated with the predicate of a sentence (called Lexical Unit) and their classification into their specific roles (Fillmore, 1985). For example, given the sentence "*Bootleggers then copy the film onto hundreds of tapes*" the task would be to recognize the verb *copy* as representing the DUPLICATION frame with roles, CREATOR for *Bootleggers*, ORIGINAL for *the film* and GOAL for *hundreds of tapes*.

Argument Boundary Detection (ABD) corresponds to the SRL subtask of detecting the sentence fragments spanning individual roles. In the previous example the phrase "*the film*" represents a role (i.e., ORIGINAL), while "*of tapes*" or "*film onto hundreds*" do not, as they just partially cover one or multiple roles, respectively. The ABD task has been successfully tackled using TKs since Moschitti et al. (2008). It can be modeled as a binary classification task over each parse tree node n, where the argument span reflects words covered by the sub-tree rooted at n. In our experiments, Grammatical Relation Centered Tree (GRCT) derived from dependency grammar (Fig. 4) are employed, as shown in Fig. 5. Each node is considered as a candidate in covering a possible *argument*. In particular, the structure in Fig. 5a is a positive example. On the contrary, in Fig. 5b the NMOD node only covers the phrase "*of tapes*", i.e., a subset of the correct role, and it represents a negative example[7].

We selected all the sentences whose predicate word (lexical unit) is a verb (they are about

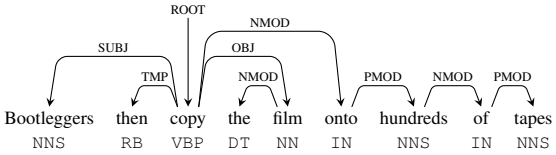


Figure 4: Example of dependency parse tree

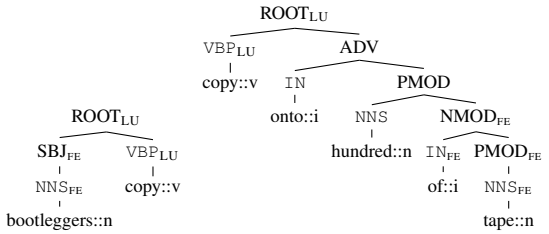


Figure 5: Conversion from dependency graph to GRCT. Tree in Fig. 5a is a positive example, while in Fig. 5b a negative one.

60,000), from the 1.3 version of the Framenet dataset (Baker et al., 1998). This gives rise to about 1,400,000 sub-trees, i.e., the positive and negative instances. The dataset is split in train and test according to the 90/10 proportion (as in (Johansson and Nugues, 2008)). This size makes the application of a traditional kernel-based method unfeasible, unless a significant instance sub-sampling is performed.

We firstly experimented standard SVM learning over a sampled training set of 10,000 examples, a typical size for annotated datasets in computational linguistics tasks. We adopted the Smoothed Partial Tree Kernel (Croce et al., 2011) with standard parameters (i.e., $\mu = \lambda = 0.4$) and lexical nodes expressed through 250-dimensional vectors obtained by applying Word2Vec (Mikolov et al., 2013) to the entire Wikipedia. When trained over this 10k instances dataset, the kernel-based SVM (KSVM) achieves an F_1 of 70.2%, over the same test set used in Croce and Basili (2016) that includes 146,399 examples. The KSVM learning produces a model including $2,994$ support vectors, i.e., the number of kernel operations required to classify each new test instance. We then apply the Nyström linearization to a larger dataset made of 100k examples, and trained a classifier using both the Dual Coordinate Descent method (Hsieh et al., 2008), SVM_{DCD}, and the KDA proposed in this work. Table 3 presents the results in terms of F_1 and saved kernel operation. Although SVM_{DCD} with 500 landmarks already achieves 0.713 F_1, a score higher than KSVM, it is signif-

[7]The nodes of the subtree covering the words to be verified as possible argument are marked with a FE tag. The word evoking the frame and its ancestor nodes are also marked with the LU tag. The other nodes are pruned out, except the ones connecting the LU nodes to the FE ones.

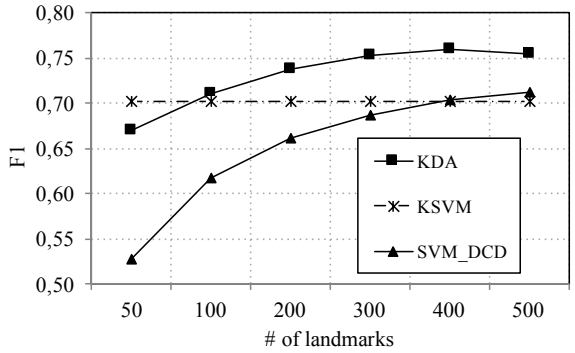


Figure 6: ABD task: F_1 measure curves w.r.t. the number of landmarks.

icantly improved by the KDA. KDA achieves up to 0.76 F_1 with only 400 landmarks, resulting in a huge step forward w.r.t. the KSVM. This result is straightforward considering (*i*) the reduction of required kernel operations, i.e., more than 86% are saved and (*ii*) the quality achieved since 100 landmarks (i.e., 0.711, higher than the KSVM).

Table 3: Results in terms of F_1 and saving in the Argument Boundary Detection task.

Model	Land.	Tr.Size	F_1	Saving
KSVM	-	10k	0.702	0.0%
KDA (SVM_{DCD})	100	100k	0.711 (0.618)	96.7%
	200	100k	0.737 (0.661)	93.3%
	300	100k	0.753 (0.686)	90.0%
	400	100k	**0.760** (0.704)	**86.6%**
	500	100k	0.754 (0.713)	83.3%

5 Discussion and Conclusions

In this work, we promoted a methodology to embed structured linguistic information within NNs, according to mathematically rich semantic similarity models, based on kernel functions. Structured data, such as trees, are transformed into dense vectors according to the Nyström methodology, and the NN is effective in capturing non-linearities in these representations, but still improving generalization at a reasonable complexity.

At the best our knowledge, this work is one of the few attempts to systematically integrate linguistic kernels within a deep neural network architecture. The problem of combining such methodologies has been studied in specific works, such as (Baldi et al., 2011; Cho and Saul, 2009; Yu et al., 2009). In Baldi et al. (2011) the authors propose a hybrid classifier, for bridging kernel methods and neural networks. In particular, they use the output of a kernelized k-nearest neighbors algorithm as input to a neural network. Cho and Saul (2009) introduced a family of kernel functions that mimic the computation of large multilayer neural networks. However, such kernels can be applied only on vector inputs. In Yu et al. (2009), deep neural networks for rapid visual recognition are trained with a novel regularization method taking advantage of kernels as an oracle representing prior knowledge. The authors transform the kernel regularizer into a loss function and carry out the neural network training by gradient descent. In Zhuang et al. (2011) a different approach has been promoted: a multiple (two) layer architecture of kernel functions, inspired by neural networks, is studied to find the best kernel combination in a Multiple Kernel Learning setting. In Mairal et al. (2014) the invariance properties of convolutional neural networks (LeCun et al., 1998) are modeled through kernel functions, resulting in a Convolutional Kernel Network. Other effort for combining NNs and kernel methods is described in Tymoshenko et al. (2016), where a SVM adopts a tree kernels combinations with embeddings learned through a CNN.

The approach here discussed departs from previous approaches in different aspects. First, a general framework is promoted: it is largely applicable to any complex kernel, e.g., structural kernels or combinations of them. The efficiency of the Nyström methodology encourages its adoption, especially when complex kernel computations are required. Notice that other low-dimensional approximations of kernel functions have been studied, as for example the randomized feature mappings proposed in Rahimi and Recht (2008). However, these assume that (*i*) instances have vectorial form and (*ii*) shift-invariant kernels are adopted. The Nyström method adopted here does not suffer of such limitations: as our target is the application to structured (linguistic) data, more general kernels, i.e., non-shift-invariant convolution kernels are needed.

Given the Nyström approximation, the learning setting corresponds to a general well-known neural network architecture, i.e., a multilayer perceptron, and does not require any manual feature engineering or the design of ad-hoc network architectures. The success in three different tasks confirms its large applicability without major changes or adaptations. Second, we propose a novel learning strategy, as the capability of kernel methods to represent complex search spaces is combined with the ability of neural networks to find non-linear so-

lutions to complex tasks. Last, the suggested KDA framework is *fully scalable*, as (*i*) the network can be parallelized on multiple machines, and (*ii*) the computation of the Nyström reconstruction vector $\mathbf{c}$ can be easily parallelized on multiple processing units, ideally l, as each unit can compute one c_i value. Future work will address experimentations with larger scale datasets; moreover, it is interesting to experiment with more landmarks in order to better understand the trade-off between the representation capacity of the Nyström approximation of the kernel functions and the over-fitting that can be introduced in a neural network architecture. Finally, the optimization of the KDA methodology through the suitable parallelization on multicore architectures, as well as the exploration of mechanisms for the dynamic reconstruction of kernel spaces (e.g., operating over H_{Ny}) also constitute interesting future research directions on this topic.

References

Paolo Annesi, Danilo Croce, and Roberto Basili. 2014. Semantic compositionality in tree kernels. In *Proceedings of CIKM 2014*. ACM.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of COLING-ACL*. Montreal, Canada.

Pierre Baldi, Chloe Azencott, and S. Joshua Swamidass. 2011. Bridging the gap between neural network and kernel methods: Applications to drug discovery. In *Proceedings of the 20th Italian Workshop on Neural Nets*. http://dl.acm.org/citation.cfm?id=1940632.1940635.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad Al-Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 task 3: Answer and question selection for question answering on arabic and english fora. In *Proceedings of SemEval-2016*.

Nicola Cancedda, Éric Gaussier, Cyril Goutte, and Jean-Michel Renders. 2003. Word-sequence kernels. *Journal of Machine Learning Research* 3:1059–1082.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* 2(3):27:1–27:27. https://doi.org/10.1145/1961189.1961199.

Youngmin Cho and Lawrence K. Saul. 2009. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, Curran Associates, Inc., pages 342–350. http://papers.nips.cc/paper/3628-kernel-methods-for-deep-learning.pdf.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proceedings of Neural Information Processing Systems (NIPS'2001)*. pages 625–632.

Danilo Croce and Roberto Basili. 2016. Large-scale kernel-based language learning through the ensemble nystrom methods. In *Proceedings of ECIR 2016*.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of EMNLP '11*. pages 1034–1046.

Ofer Dekel and Yoram Singer. 2006. Support vector machines on a budget. In *NIPS*. MIT Press, pages 345–352.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 task 3: Learning semantic relations between questions and comments. In *Proceedings of SemEval '16*.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015. Structural representations for learning relations between pairs of texts. In *Proceedings of ACL 2015*. Beijing, China, pages 1003–1013. http://www.aclweb.org/anthology/P15-1097.

Charles J. Fillmore. 1985. Frames and the semantics of understanding. *Quaderni di Semantica* 6(2):222–254.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics* 28(3):245–288.

David Haussler. 1999. Convolution kernels on discrete structures. In *Technical Report UCS-CRL-99-10*. University of California, Santa Cruz.

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

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the ICML 2008*. ACM, pages 408–415.

Richard Johansson and Pierre Nugues. 2008. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings EMNLP 2014*. Doha, Qatar, pages 1746–1751.

Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. 2012. Sampling methods for the nyström method. *J. Mach. Learn. Res.* 13:981–1006.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. of the IEEE* 86(11).

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225. https://transacl.org/ojs/index.php/tacl/article/view/570.

Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering* 12(3):229–249.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. 2014. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*. Berlin, Germany.

Alessandro Moschitti, Daniele Pighin, and Robert Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics* 34.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 task 3: Community question answering. In *Proceedings of SemEval-2016*.

Ali Rahimi and Benjamin Recht. 2008. Random features for large-scale kernel machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, Curran Associates, Inc., pages 1177–1184. http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf.

Magnus Sahlgren. 2006. *The Word-Space Model*. Ph.D. thesis, Stockholm University.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Building structures from classifiers for passage reranking. ACM, New York, NY, USA, CIKM '13, pages 969–978.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP '13*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. pages 1556–1566. http://aclweb.org/anthology/P/P15/P15-1150.pdf.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of NAACL 2016*. http://www.aclweb.org/anthology/N16-1152.

Zhuang Wang and Slobodan Vucetic. 2010. Online passive-aggressive algorithms on a budget. *Journal of Machine Learning Research - Proceedings Track* 9:908–915.

Christopher K. I. Williams and Matthias Seeger. 2001. Using the nyström method to speed up kernel machines. In *Proceedings of NIPS 2000*.

Kai Yu, Wei Xu, and Yihong Gong. 2009. Deep learning with kernel regularization for visual recognition. In *Advances in Neural Information Processing Systems 21*, Curran Associates, Inc., pages 1889–1896.

Jinfeng Zhuang, Ivor W. Tsang, and Steven C. H. Hoi. 2011. Two-layer multiple kernel learning. In *AISTATS*. JMLR.org, volume 15 of *JMLR Proceedings*, pages 909–917.

Topically Driven Neural Language Model

Jey Han Lau[1,2] **Timothy Baldwin**[2] **Trevor Cohn**[2]

[1] IBM Research

[2] School of Computing and Information Systems,
The University of Melbourne

jeyhan.lau@gmail.com, tb@ldwin.net, t.cohn@unimelb.edu.au

Abstract

Language models are typically applied at the sentence level, without access to the broader document context. We present a neural language model that incorporates document context in the form of a topic model-like architecture, thus providing a succinct representation of the broader document context outside of the current sentence. Experiments over a range of datasets demonstrate that our model outperforms a pure sentence-based model in terms of language model perplexity, and leads to topics that are potentially more coherent than those produced by a standard LDA topic model. Our model also has the ability to generate related sentences for a topic, providing another way to interpret topics.

1 Introduction

Topic models provide a powerful tool for extracting the macro-level content structure of a document collection in the form of the latent topics (usually in the form of multinomial distributions over terms), with a plethora of applications in NLP (Hall et al., 2008; Newman et al., 2010a; Wang and McCallum, 2006). A myriad of variants of the classical LDA method (Blei et al., 2003) have been proposed, including recent work on neural topic models (Cao et al., 2015; Wan et al., 2012; Larochelle and Lauly, 2012; Hinton and Salakhutdinov, 2009).

Separately, language models have long been a foundational component of any NLP task involving generation or textual normalisation of a noisy input (including speech, OCR and the processing of social media text). The primary purpose of a language model is to predict the probability of a span of text, traditionally at the sentence level, under the assumption that sentences are independent of one another, although recent work has started using broader local context such as the preceding sentences (Wang and Cho, 2016; Ji et al., 2016).

In this paper, we combine the benefits of a topic model and language model in proposing a topically-driven language model, whereby we jointly learn topics and word sequence information. This allows us to both sensitise the predictions of the language model to the larger document narrative using topics, and to generate topics which are better sensitised to local context and are hence more coherent and interpretable.

Our model has two components: a language model and a topic model. We implement both components using neural networks, and train them jointly by treating each component as a sub-task in a multi-task learning setting. We show that our model is superior to other language models that leverage additional context, and that the generated topics are potentially more coherent than LDA topics. The architecture of the model provides an extra dimensionality of topic interpretability, in supporting the generation of sentences from a topic (or mix of topics). It is also highly flexible, in its ability to be supervised and incorporate side information, which we show to further improve language model performance. An open source implementation of our model is available at: https://github.com/jhlau/topically-driven-language-model.

2 Related Work

Griffiths et al. (2004) propose a model that learns topics and word dependencies using a Bayesian framework. Word generation is driven by either LDA or an HMM. For LDA, a word is generated based on a sampled topic in the document. For the

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 355–365
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1033

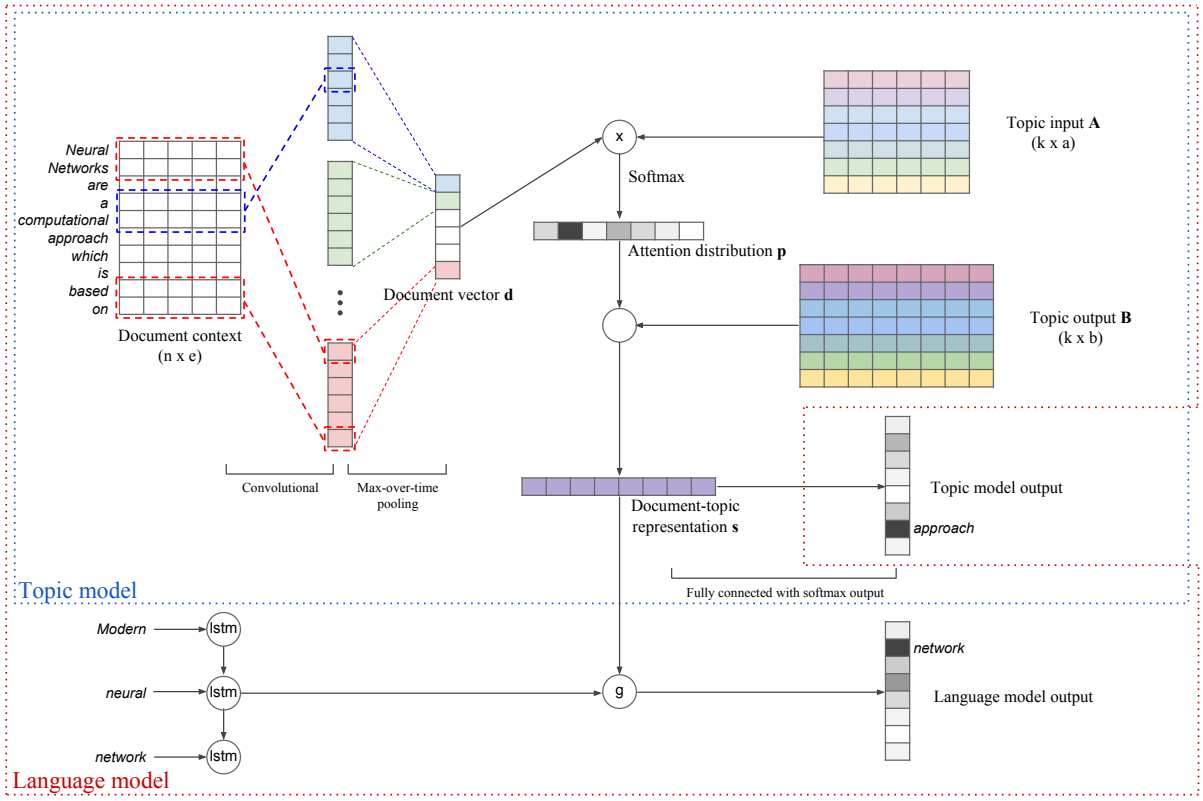


Figure 1: Architecture of `tdlm`. Scope of the models are denoted by dotted lines: blue line denotes the scope of the topic model, red the language model.

HMM, a word is conditioned on previous words. A key difference over our model is that their language model is driven by an HMM, which uses a fixed window and is therefore unable to track long-range dependencies.

Cao et al. (2015) relate the topic model view of documents and words — documents having a multinomial distribution over topics and topics having a multinomial distributional over words — from a neural network perspective by embedding these relationships in differentiable functions. With that, the model lost the stochasticity and Bayesian inference of LDA but gained non-linear complex representations. The authors further propose extensions to the model to do supervised learning where document labels are given.

Wang and Cho (2016) and Ji et al. (2016) relax the sentence independence assumption in language modelling, and use preceeding sentences as additional context. By treating words in preceeding sentences as a bag of words, Wang and Cho (2016) use an attentional mechanism to focus on these words when predicting the next word. The authors show that the incorporation of additional context helps language models.

3 Architecture

The architecture of the proposed topically-driven language model (henceforth "`tdlm`") is illustrated in Figure 1. There are two components in `tdlm`: a language model and a topic model. The language model is designed to capture word relations in sentences, while the topic model learns topical information in documents. The topic model works like an auto-encoder, where it is given the document words as input and optimised to predict them.

The topic model takes in word embeddings of a document and generates a document vector using a convolutional network. Given the document vector, we associate it with the topics via an attention scheme to compute a weighted mean of topic vectors, which is then used to predict a word in the document.

The language model is a standard LSTM language model (Hochreiter and Schmidhuber, 1997; Mikolov et al., 2010), but it incorporates the weighted topic vector generated by the topic model to predict succeeding words.

Marrying the language and topic models allows the language model to be topically driven, i.e. it models not just word contexts but also the document context where the sentence occurs, in the form of topics.

3.1 Topic Model Component

Let $\mathbf{x}_i \in \mathbb{R}^e$ be the e-dimensional word vector for the i-th word in the document. A document of n words is represented as a concatenation of its word vectors:

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus ... \oplus \mathbf{x}_n$$

where $\oplus$ denotes the concatenation operator. We use a number of convolutional filters to process the word vectors, but for clarity we will explain the network with one filter.

Let $\mathbf{w}_v \in \mathbb{R}^{eh}$ be a convolutional filter which we apply to a window of h words to generate a feature. A feature c_i for a window of words $\mathbf{x}_{i:i+h-1}$ is given as follows:

$$c_i = I(\mathbf{w}_v^\intercal \mathbf{x}_{i:i+h-1} + b_v)$$

where b_v is a bias term and I is the identity function.[1] A feature map $\mathbf{c}$ is a collection of features computed from all windows of words:

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}]$$

where $\mathbf{c} \in \mathbb{R}^{n-h+1}$. To capture the most salient features in $\mathbf{c}$, we apply a max-over-time pooling operation (Collobert et al., 2011), yielding a scalar:

$$d = \max_i c_i$$

In the case where we use a filters, we have $\mathbf{d} \in \mathbb{R}^a$, and this constitutes the vector representation of the document generated by the convolutional and max-over-time pooling network.

The topic vectors are stored in two lookup tables $\mathbf{A} \in \mathbb{R}^{k \times a}$ (input vector) and $\mathbf{B} \in \mathbb{R}^{k \times b}$ (output vector), where k is the number of topics, and a and b are the dimensions of the topic vectors.

To align the document vector $\mathbf{d}$ with the topics, we compute an attention vector which is used to compute a document-topic representation:[2]

$$\mathbf{p} = \text{softmax}(\mathbf{A}\mathbf{d}) \quad (1)$$

$$\mathbf{s} = \mathbf{B}^\intercal \mathbf{p} \quad (2)$$

where $\mathbf{p} \in \mathbb{R}^k$ and $\mathbf{s} \in \mathbb{R}^b$. Intuitively, $\mathbf{s}$ is a weighted mean of topic vectors, with the weighting given by the attention $\mathbf{p}$. This is inspired by the generative process of LDA, whereby documents are defined as having a multinomial distribution over topics.

Finally $\mathbf{s}$ is connected to a dense layer with softmax output to predict each word in the document, where each word is generated independently as a unigram bag-of-words, and the model is optimised using categorical cross-entropy loss. In practice, to improve efficiency we compute loss for predicting a sequence of m_1 words in the document, where m_1 is a hyper-parameter.

3.2 Language Model Component

The language model is implemented using LSTM units (Hochreiter and Schmidhuber, 1997):

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{v}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{v}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{v}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\hat{\mathbf{c}}_t &= \tanh(\mathbf{W}_c \mathbf{v}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

where $\odot$ denotes element-wise product; $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the input, forget and output activations respectively at time step t; and $\mathbf{v}_t$, $\mathbf{h}_t$ and $\mathbf{c}_t$ are the input word embedding, LSTM hidden state, and cell state, respectively. Hereinafter $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{b}$ are used to refer to the model parameters.

Traditionally, a language model operates at the sentence level, predicting the next word given its history of words in the sentence. The language model of `tdlm` incorporates topical information by assimilating the document-topic representation ($\mathbf{s}$) with the hidden output of the LSTM ($\mathbf{h}_t$) at each time step t. To prevent `tdlm` from memorising the next word via the topic model network, we exclude the current sentence from the document context.

[1] A non-linear function is typically used here, but preliminary experiments suggest that the identity function works best for `tdlm`.

[2] The attention mechanism was inspired by memory networks (Graves et al., 2014; Weston et al., 2014; Sukhbaatar et al., 2015; Tran et al., 2016). We explored various attention styles (including traditional schemes which use one vector for a topic), but found this approach to work best.

We use a gating unit similar to a GRU (Cho et al., 2014; Chung et al., 2014) to allow tdlm to learn the degree of influence of topical information on the language model:

$$\begin{aligned}\mathbf{z}_t &= \sigma(\mathbf{W}_z\mathbf{s} + \mathbf{U}_z\mathbf{h}_t + \mathbf{b}_z)\\ \mathbf{r}_t &= \sigma(\mathbf{W}_r\mathbf{s} + \mathbf{U}_r\mathbf{h}_t + \mathbf{b}_r)\\ \hat{\mathbf{h}}_t &= \tanh(\mathbf{W}_h\mathbf{s} + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_t) + \mathbf{b}_h)\\ \mathbf{h}'_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_t + \mathbf{z}_t \odot \hat{\mathbf{h}}_t\end{aligned} \quad (3)$$

where $\mathbf{z}_t$ and $\mathbf{r}_t$ are the update and reset gate activations respectively at timestep t. The new hidden state $\mathbf{h}'_t$ is connected to a dense layer with linear transformation and softmax output to predict the next word, and the model is optimised using standard categorical cross-entropy loss.

3.3 Training and Regularisation

tdlm is trained using minibatches and SGD.[3] For the language model, a minibatch consists of a batch of sentences, while for the topic model it is a batch of documents (each predicting a sequence of m_1 words).

We treat the language and topic models as subtasks in a multi-task learning setting, and train them jointly using categorical cross-entropy loss. Most parameters in the topic model are shared by the language model, as illustrated by their scopes (dotted lines) in Figure 1.

Hyper-parameters of tdlm are detailed in Table 1. Word embeddings for the topic model and language model components are not shared, although their dimensions are the same (e).[4] For m_1, m_2 and m_3, sequences/documents shorter than these thresholds are padded. Sentences longer than m_2 are broken into multiple sequences, and documents longer than m_3 are truncated. Optimal hyper-parameter settings are tuned using the development set; the presented values are used for experiments in Sections 4 and 5.

To regularise tdlm, we use dropout regularisation (Srivastava et al., 2014). We apply dropout to $\mathbf{d}$ and $\mathbf{s}$ in the topic model, and to the input word embedding and hidden output of the LSTM in the language model (Pham et al., 2013; Zaremba et al., 2014).

4 Language Model Evaluation

We use standard language model perplexity as the evaluation metric. In terms of dataset, we use document collections from 3 sources: APNEWS, IMDB and BNC. APNEWS is a collection of Associated Press[5] news articles from 2009 to 2016. IMDB is a set of movie reviews collected by Maas et al. (2011). BNC is the written portion of the British National Corpus (BNC Consortium, 2007), which contains excerpts from journals, books, letters, essays, memoranda, news and other types of text. For APNEWS and BNC, we randomly sub-sample a set of documents for our experiments.

For preprocessing, we tokenise words and sentences using Stanford CoreNLP (Klein and Manning, 2003). We lowercase all word tokens, filter word types that occur less than 10 times, and exclude the top 0.1% most frequent word types.[6] We additionally remove stopwords for the topic model document context.[7] All datasets are partitioned into training, development and test sets; preprocessed dataset statistics are presented in Table 2.

We tune hyper-parameters of tdlm based on development set language model perplexity. In general, we find that optimal settings are fairly robust across collections, with the exception of m_3, as document length is collection dependent; optimal hyper-parameter values are given in Table 1. In terms of LSTM size, we explore 2 settings: a small model with 1 LSTM layer and 600 hidden units, and a large model with 2 layers and 900 hidden units.[8] For the topic number, we experiment with 50, 100 and 150 topics. Word embeddings are pre-trained 300-dimension word2vec Google News vectors.[9]

For comparison, we compare tdlm with:[10]

vanilla-lstm: A standard LSTM language model, using the same tdlm hyper-parameters where applicable. This is the baseline model.

lclm: A larger context language model that incorporates context from preceding sentences (Wang and Cho, 2016), by treating the preceding sentence as a bag of words, and using an

[3] We use Adam as the optimiser (Kingma and Ba, 2014).

[4] Word embeddings are updated during training.

[5] https://www.ap.org/en-gb/.

[6] For the topic model, we remove word tokens that correspond to these filtered word types; for the language model we represent them as ⟨unk⟩ tokens (as for unseen words in test).

[7] We use Mallet's stopword list: https://github.com/mimno/Mallet/tree/master/stoplists.

[8] Multi-layer LSTMs are vanilla stacked LSTMs without skip connections (Gers and Schmidhuber, 2000) or depth-gating (Yao et al., 2015).

[9] https://code.google.com/archive/p/word2vec/.

[10] Note that all models use the same pre-trained word2vec vectors.

Hyper-parameter	Value	Description
m_1	3	Output sequence length for topic model
m_2	30	Sequence length for language model
m_3	300,150,500	Maximum document length
n_{batch}	64	Minibatch size
n_{layer}	1,2	Number of LSTM layers
n_{hidden}	600,900	LSTM hidden size
n_{epoch}	10	Number of training epochs
k	100,150,200	Number of topics
e	300	Word embedding size
h	2	Convolutional filter width
a	20	Topic input vector size or number of features for convolutional filter
b	50	Topic output vector size
l	0.001	Learning rate of optimiser
p_1	0.4	Topic model dropout keep probability
p_2	0.6	Language model dropout keep probability

Table 1: `tdlm` hyper-parameters; we experiment with 2 LSTM settings and 3 topic numbers, and m_3 varies across the three domains (APNEWS, IMDB, and BNC).

Collection	Training		Development		Test	
	#Docs	#Tokens	#Docs	#Tokens	#Docs	#Tokens
APNEWS	50K	15M	2K	0.6M	2K	0.6M
IMDB	75K	20M	12.5K	0.3M	12.5K	0.3M
BNC	15K	18M	1K	1M	1K	1M

Table 2: Preprocessed dataset statistics.

attentional mechanism when predicting the next word. An additional hyper-parameter in `lclm` is the number of preceeding sentences to incorporate, which we tune based on a development set (to 4 sentences in each case). All other hyper-parameters (such as n_{batch}, e, n_{epoch}, k_2) are the same as `tdlm`.

`lstm+lda`: A standard LSTM language model that incorporates LDA topic information. We first train an LDA model (Blei et al., 2003; Griffiths and Steyvers, 2004) to learn 50/100/150 topics for APNEWS, IMDB and BNC.[11] For a document, the LSTM incorporates the LDA topic distribution ($\mathbf{q}$) by concatenating it with the output hidden state ($\mathbf{h}_t$) to predict the next word (i.e. $\mathbf{h}'_t = \mathbf{h}_t \oplus \mathbf{q}$). That is, it incorporates topical information into the language model, but unlike `tdlm` the language model and topic model are trained separately.

We present language model perplexity performance in Table 3. All models outperform the baseline `vanilla-lstm`, with `tdlm` performing the best across all collections. `lclm` is competitive over the BNC, although the superiority of `tdlm` for the other collections is substantial. `lstm+lda` performs relatively well over APNEWS and IMDB, but very poorly over BNC.

The strong performance of `tdlm` over `lclm` suggests that compressing document context into topics benefits language modelling more than using extra context words directly.[12] Overall, our results show that topical information can help language modelling and that joint inference of topic and language model produces the best results.

5 Topic Model Evaluation

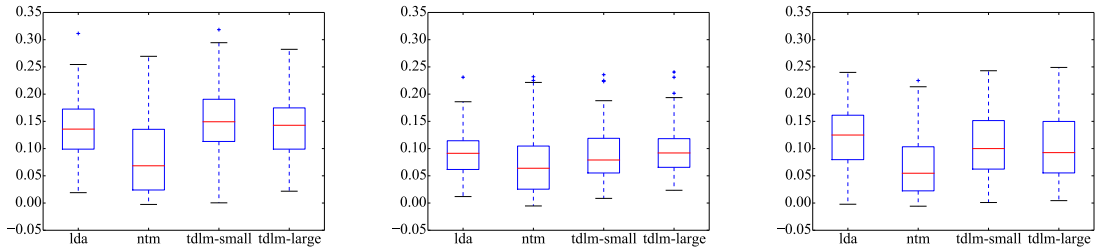
We saw that `tdlm` performs well as a language model, but it is also a topic model, and like LDA it produces: (1) a probability distribution over topics for each document (Equation (1)); and (2) a probability distribution over word types for each topic.

[11] Based on Gibbs sampling; $\alpha = 0.1$, $\beta = 0.01$.

[12] The context size of `lclm` (4 sentences) is technically smaller than `tdlm` (full document), however, note that increasing the context size does not benefit `lclm`, as the context size of 4 gives the best performance.

Domain	LSTM Size	vanilla-lstm	lclm	lstm+lda			tdlm		
				50	100	150	50	100	150
APNEWS	small	64.13	54.18	57.05	55.52	54.83	53.00	52.75	**52.65**
	large	58.89	50.63	52.72	50.75	50.17	48.96	48.97	**48.21**
IMDB	small	72.14	67.78	69.58	69.64	69.62	63.67	**63.45**	63.82
	large	66.47	67.86	63.48	63.04	62.78	58.99	59.04	**58.59**
BNC	small	102.89	87.47	96.42	96.50	96.38	87.42	**85.99**	86.43
	large	94.23	80.68	88.42	87.77	87.28	82.62	81.83	**80.58**

Table 3: Language model perplexity performance of all models over APNEWS, IMDB and BNC. Boldface indicates best performance in each row.



(a) APNEWS (b) IMDB (c) BNC

Figure 2: Boxplots of topic coherence of all models; number of topics = 100.

Recall that $\mathbf{s}$ is a weighted mean of topic vectors for a document (Equation (2)). Generating the vocabulary distribution for a particular topic is therefore trivial: we can do so by treating $\mathbf{s}$ as having maximum weight (1.0) for the topic of interest, and no weight (0.0) for all other topics. Let $\mathbf{B}_t$ denote the topic output vector for the t-th topic. To generate the multinomial distribution over word types for the t-th topic, we replace $\mathbf{s}$ with $\mathbf{B}_t$ before computing the softmax over the vocabulary.

Topic models are traditionally evaluated using model perplexity. There are various ways to estimate test perplexity (Wallach et al., 2009), but Chang et al. (2009) show that perplexity does not correlate with the coherence of the generated topics. Newman et al. (2010b); Mimno et al. (2011); Aletras and Stevenson (2013) propose automatic approaches to computing topic coherence, and Lau et al. (2014) summarises these methods to understand their differences. We propose using automatic topic coherence as a means to evaluate the topic model aspect of `tdlm`.

Following Lau et al. (2014), we compute topic coherence using normalised PMI ("NPMI") scores. Given the top-n words of a topic, coherence is computed based on the sum of pairwise NPMI scores between topic words, where the word probabilities used in the NPMI calculation are based on co-occurrence statistics mined from English Wikipedia with a sliding window (Newman et al., 2010b; Lau et al., 2014).[13]

Based on the findings of Lau and Baldwin (2016), we average topic coherence over the top-5/10/15/20 topic words. To aggregate topic coherence scores for a model, we calculate the mean coherence over topics.

In terms of datasets, we use the same document collections (APNEWS, IMDB and BNC) as the language model experiments (Section 4). We use the same hyper-parameter settings for `tdlm` and do not tune them.

For comparison, we use the following topic models:

`lda`: We use a LDA model as a baseline topic model. We use the same LDA models as were used to learn topic distributions for `lstm+lda` (Section 4).

[13] We use this toolkit to compute topic coherence: https://github.com/jhlau/topic_interpretability.

Topic No.	System	Coherence		
		APNEWS	IMDB	BNC
50	lda	.125	.084	**.106**
	ntm	.075	.064	.081
	tdlm-small	**.149**	**.104**	.102
	tdlm-large	.130	.088	.095
100	lda	.136	.092	**.119**
	ntm	.085	.071	.070
	tdlm-small	**.152**	.087	.106
	tdlm-large	.142	**.097**	.101
150	lda	.134	**.094**	**.119**
	ntm	.078	.075	.072
	tdlm-small	**.147**	.085	.100
	tdlm-large	.145	.091	.104

Table 4: Mean topic coherence of all models over APNEWS, IMDB and BNC. Boldface indicates the best performance for each dataset and topic setting.

ntm: ntm is a neural topic model proposed by Cao et al. (2015). The document-topic and topic-word multinomials are expressed from a neural network perspective using differentiable functions. Model hyper-parameters are tuned using development loss.

Topic model performance is presented in Table 4. There are two models of tdlm (tdlm-small and tdlm-large), which specify the size of its LSTM model (1 layer+600 hidden vs. 2 layers+900 hidden; see Section 4). tdlm achieves encouraging results: it has the best performance over APNEWS, and is competitive over IMDB. lda, however, produces more coherent topics over BNC. Interestingly, coherence appears to increase as the topic number increases for lda, but the trend is less pronounced for tdlm. ntm performs the worst of the 3 topic models, and manual inspection reveals that topics are in general not very interpretable. Overall, the results suggest that tdlm topics are competitive: at best they are more coherent than lda topics, and at worst they are as good as lda topics.

To better understand the spread of coherence scores and impact of outliers, we present box plots for all models (number of topics = 100) over the 3 domains in Figure 2. Across all domains, ntm has poor performance and larger spread of scores. The difference between lda and tdlm is small (tdlm > lda in APNEWS, but lda < tdlm in BNC), which is consistent with our previous observation that tdlm topics are competitive with lda topics.

Partition	#Docs	#Tokens
Training	9314	2.6M
Development	2000	0.5M
Test	7532	1.7M

Table 5: 20NEWS preprocessed statistics.

6 Extensions

One strength of tdlm is its flexibility, owing to it taking the form of a neural network. To showcase this flexibility, we explore two simple extensions of tdlm, where we: (1) build a supervised model using document labels (Section 6.1); and (2) incorporate additional document metadata (Section 6.2).

6.1 Supervised Model

In datasets where document labels are known, supervised topic model extensions are designed to leverage the additional information to improve modelling quality. The supervised setting also has an additional advantage in that model evaluation is simpler, since models can be quantitatively assessed via classification accuracy.

To incorporate supervised document labels, we treat document classification as another sub-task in tdlm. Given a document and its label, we feed the document through the topic model network to generate the document-topic representation $\mathbf{s}$, and connect it to another dense layer with softmax output to generate the probability distribution over classes.

During training, we have additional minibatches for the documents. We start the document classification training after the topic and language models have completed training in each epoch.

We use 20NEWS in this experiment, which is a popular dataset for text classification. 20NEWS is a collection of forum-like messages from 20 newsgroups categories. We use the "bydate" version of the dataset, where the train and test partition is separated by a specific date. We sample 2K documents from the training set to create the development set. For preprocessing we tokenise words and sentence using Stanford CoreNLP (Klein and Manning, 2003), and lowercase all words. As with previous experiments (Section 4) we additionally filter low/high frequency word types and stopwords. Preprocessed dataset statistics are presented in Table 5.

For comparison, we use the same two topic

Topic No.	System	Accuracy
50	lda	.567
	ntm	**.649**
	tdlm	.606
100	lda	.581
	ntm	**.639**
	tdlm	.602
150	lda	.597
	ntm	**.628**
	tdlm	.601

Table 6: 20NEWS classification accuracy. All models are supervised extensions of the original models. Boldface indicates the best performance for each topic setting.

Topic No.	Metadata	Coherence	Perplexity
50	No	.128	52.45
	Yes	**.131**	**51.80**
100	No	**.142**	52.14
	Yes	.139	**51.76**
150	No	.135	52.25
	Yes	**.143**	**51.58**

Table 7: Topic coherence and language model perplexity by incorporating classification tags on APNEWS. Boldface indicates optimal coherence and perplexity performance for each topic setting.

models as in Section 5: ntm and lda. Both ntm and lda have natural supervised extensions (Cao et al., 2015; McAuliffe and Blei, 2008) for incorporating document labels. For this task, we tune the model hyper-parameters based on development accuracy.[14] Classification accuracy for all models is presented in Table 6. We present tdlm results using only the small setting of LSTM (1 layer + 600 hidden), as we found there is little gain when using a larger LSTM.

ntm performs very strongly, outperforming both lda and tdlm by a substantial margin. Comparing lda and tdlm, tdlm achieves better performance, especially when there is a smaller number of topics. Upon inspection of the topics we found that ntm topics are much less coherent than those of lda and tdlm, consistent with our observations from Section 5.

[14]Most hyper-parameter values for tdlm are similar to those used in the language and topic model experiments; the only exceptions are: $a = 80$, $b = 100$, $n_{epoch} = 20$, $m_3 = 150$. The increase in parameters is unsurprising, as the additional supervision provides more constraint to the model.

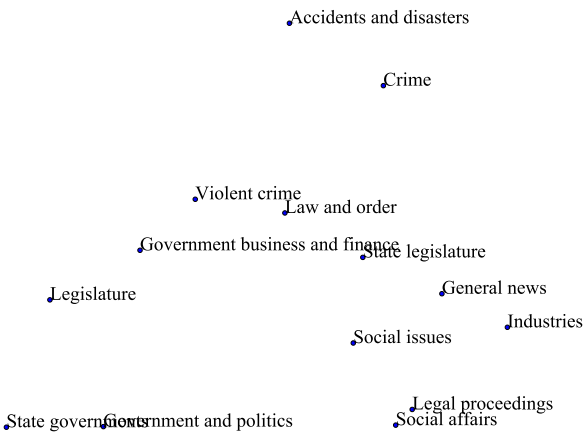


Figure 3: Scatter plots of tag embeddings (model=150 topics)

6.2 Incorporating Document Metadata

In APNEWS, each news article contains additional document metadata, including subject classification tags, such as "General News", "Accidents and Disasters", and "Military and Defense". We present an extension to incorporate document metadata in tdlm to demonstrate its flexibility in integrating this additional information.

As some of the documents in our original APNEWS sample were missing tags, we re-sampled a set of APNEWS articles of the same size as our original, all of which have tags. In total, approximately 1500 unique tags can be found among the training articles.

To incorporate these tags, we represent each of them as a learnable vector and concatenate it with the document vector before computing the attention distribution. Let $\mathbf{z}_i \in \mathbb{R}^f$ denote the f-dimension vector for the i-th tag. For the j-th document, we sum up all tags associated with it:

$$\mathbf{e} = \sum_{i=1}^{n_{tags}} \mathbb{I}(i,j)\mathbf{z}_i$$

where n_{tags} is the total number of unique tags, and function $\mathbb{I}(i,j)$ returns 1 is the i-th tag is in the j-th document or 0 otherwise. We compute $\mathbf{d}$ as before (Section 3.1), and concatenate it with the summed tag vector: $\mathbf{d}' = \mathbf{d} \oplus \mathbf{e}$.

We train two versions of tdlm on the new APNEWS dataset: (1) the vanilla version that ignores the tag information; and (2) the extended version which incorporates tag information.[15] We exper-

[15]Model hyper-parameters are the same as the ones used in the language (Section 4) and topic model (Section 5) experiments.

Topic	Generated Sentences
protesters suspect gunman officers occupy gun arrests suspects shooting officer	• police say a suspect in the shooting was shot in the chest and later shot and killed by a police officer . • a police officer shot her in the chest and the man was killed . • police have said four men have been killed in a shooting in suburban london .
film awards actress comedy music actor album show nominations movie	• it 's like it 's not fair to keep a star in a light , " he says . • but james , a four-time star , is just a ⟨unk⟩. • a ⟨unk⟩ adaptation of the movie " the dark knight rises " won best picture and he was nominated for best drama for best director of " ⟨unk⟩, " which will be presented sunday night .
storm snow weather inches flooding rain service winds tornado forecasters	• temperatures are forecast to remain above freezing enough to reach a tropical storm or heaviest temperatures . • snowfall totals were one of the busiest in the country . • forecasters say tornado irene 's strong winds could ease visibility and funnel clouds of snow from snow monday to the mountains .
virus nile flu vaccine disease outbreak infected symptoms cough tested	• he says the disease was transmitted by an infected person . • ⟨unk⟩ says the man 's symptoms are spread away from the heat . • meanwhile in the ⟨unk⟩, the virus has been common in the mojave desert .

Table 8: Generated sentences for APNEWS topics.

imented with a few values for the tag vector size (f) and find that a small value works well; in the following experiments we use $f = 5$. We evaluate the models based on language model perplexity and topic model coherence, and present the results in Table 7.[16]

In terms of language model perplexity, we see a consistent improvement over different topic settings, suggesting that the incorporation of tags improves modelling. In terms of topic coherence, there is a small but encouraging improvement (with one exception).

To investigate whether the vectors learnt for these tags are meaningful, we plot the top-14 most frequent tags in Figure 3.[17] The plot seems reasonable: there are a few related tags that are close to each other, e.g. "State government" and "Government and politics"; "Crime" and "Violent Crime"; and "Social issues" and "Social affairs".

7 Discussion

Topics generated by topic models are typically interpreted by way of their top-N highest probability words. In `tdlm`, we can additionally generate sentences related to the topic, providing another way to understand the topics. To do this, we can constrain the topic vector for the language model to be the topic output vector of a particular topic (Equation (3)).

We present 4 topics from a APNEWS model ($k = 100$; LSTM size = "large") and 3 randomly generated sentences conditioned on each topic in Table 8.[18] The generated sentences highlight the content of the topics, providing another interpretable aspect for the topics. These results also reinforce that the language model is driven by topics.

8 Conclusion

We propose `tdlm`, a topically driven neural language model. `tdlm` has two components: a language model and a topic model, which are jointly trained using a neural network. We demonstrate that `tdlm` outperforms a state-of-the-art language model that incorporates larger context, and that its topics are potentially more coherent than LDA topics. We additionally propose simple extensions of `tdlm` to incorporate information such as document labels and metadata, and achieved encouraging results.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and valuable suggestions. This work was funded in part by the Australian Research Council.

References

Nikos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In *Proceedings of the Tenth International Workshop on Computational Semantics (IWCS-10)*. Potsdam, Germany, pages 13–22.

[16] As the vanilla `tdlm` is trained on the new APNEWS dataset, the numbers are slightly different to those in Tables 3 and 4.

[17] The 5-dimensional vectors are compressed using PCA.

[18] Words are sampled with temperature = 0.75. Generation is terminated when a special end symbol is generated or when sentence length is greater than 40 words.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

BNC Consortium. 2007. The British National Corpus, version 3 (BNC XML Edition). Distributed by Oxford University Computing Services on behalf of the BNC Consortium. http://www.natcorp.ox.ac.uk/.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Proceedings of the 29th Annual Conference on Artificial Intelligence (AAAI-15)*. Austin, Texas, pages 2210–2216.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems 21 (NIPS-09)*. Vancouver, Canada, pages 288–296.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar, pages 103–111.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning and Representation Learning Workshop*. Montreal, Canada, pages 103–111.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Felix A. Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'2000)*. Como, Italy, pages 198–194.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR* abs/1410.5401.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101:5228–5235.

Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2004. Integrating topics and syntax. In *Advances in Neural Information Processing Systems 17 (NIPS-05)*. Vancouver, Canada, pages 537–544.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*. Honolulu, USA, pages 363–371.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. 2009. Replicated softmax: an undirected topic model. In *Advances in Neural Information Processing Systems 21 (NIPS-09)*. Vancouver, Canada, pages 1607–1614.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9:1735–1780.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2016. Document context language models. In *Proceedings of ICLR-16 Workshop, 2016*. Toulon, France.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*. Sapporo, Japan, pages 423–430.

Hugo Larochelle and Stanislas Lauly. 2012. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems 25*. pages 2708–2716.

Jey Han Lau and Timothy Baldwin. 2016. The sensitivity of topic coherence evaluation to topic cardinality. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2016)*. San Diego, USA, pages 483–487.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the EACL (EACL 2014)*. Gothenburg, Sweden, pages 530–539.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL HLT 2011)*. Portland, Oregon, USA, pages 142–150.

Jon D. McAuliffe and David M. Blei. 2008. Supervised topic models. In *Advances in Neural Information Processing Systems 20 (NIPS-08)*. Vancouver, Canada, pages 121–128.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*. Makuhari, Japan, pages 1045–1048.

David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*. Edinburgh, UK, pages 262–272.

David Newman, Timothy Baldwin, Lawrence Cavedon, Sarvnaz Karimi, David Martinez, and Justin Zobel. 2010a. Visualizing document collections and search results using topic mapping. *Journal of Web Semantics* 8(2–3):169–175.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010b. Automatic evaluation of topic coherence. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010)*. Los Angeles, USA, pages 100–108.

Vu Pham, Christopher Kermorvant, and Jérôme Louradour. 2013. Dropout improves recurrent neural networks for handwriting recognition. *CoRR* abs/1312.4569.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28 (NIPS-15)*. Montreal, Canada, pages 2440–2448.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2016)*. San Diego, California, pages 321–331.

Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *Proceedings of the 26th International Conference on Machine Learning (ICML-09)*. Montreal, Canada, pages 1105–1112.

Li Wan, Leo Zhu, and Rob Fergus. 2012. A hybrid neural network-latent topic model. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*. La Palma, Canary Islands, pages 1287–1294.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. Berlin, Germany, pages 1319–1329.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Philadelphia, USA, pages 424–433.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. 2015. Depth-gated LSTM. *CoRR* abs/1508.03790.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR* abs/1409.2329.

Handling Cold-Start Problem in Review Spam Detection by Jointly Embedding Texts and Behaviors

Xuepeng Wang[1,2], **Kang Liu**[1], and **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
{xpwang, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Solving the cold-start problem in review spam detection is an urgent and significant task. It can help the on-line review websites to relieve the damage of spammers in time, but has never been investigated by previous work. This paper proposes a novel neural network model to detect review spam for the cold-start problem, by learning to represent the new reviewers' review with jointly embedded textual and behavioral information. Experimental results prove the proposed model achieves an effective performance and possesses preferable domain-adaptability. It is also applicable to a large-scale dataset in an unsupervised way.

1 Introduction

With the rapid growth of products reviews at the web, it has become common for people to read reviews before making a purchase decision. The reviews usually contain abundant consumers' personal experiences. It has led to a significant influence on financial gains and fame for businesses. Existing studies have shown that an extra half-star rating on Yelp causes restaurants to sell out 19% points more frequently (Anderson and Magruder, 2012), and a one-star increase in Yelp rating leads to a 5-9 % increase in revenue (Luca, 2011). This, unfortunately, gives strong incentives for imposters (called spammers) to game the system. They post fake reviews or opinions (called review spam) to promote or to discredit some targeted products and services. The news from BBC has shown that around 25% of Yelp reviews could be fake.[1] Therefore, it is urgent to detect review spam, to ensure that the online review continues to be trusted.

Jindal and Liu (2008) make the first step to detect review spam. Most efforts are devoted to exploring effective linguistic and behavioral features by subsequent work to distinguish such spam from the real reviews. However, to notice such patterns or form behavioral features, developers should take a long time to observe the data, because the features are based on statistics. For instance, the feature *activity window* proposed by Mukherjee et al. (2013c) is to measure the activity freshness of reviewers. It usually takes several months to count the difference of timestamps between the last and first reviews for reviewers. When the features show themselves finally, some major damages might have already been done. Thus, *it is important to design algorithms that can detect review spam as soon as possible, ideally, right after they are posted by the new reviewers*. It is a cold-start problem which is the focus of this paper.

In this paper, we assume that we must identify fake reviews immediately when a new reviewer posts just one review. Unfortunately, it is very difficult because the available information for detecting fake reviews is very poor. Traditional behavioral features based on the statistics can only work well on users' abundant behaviors. The more behavioral information obtained, the more effective the traditional behavioral features are (see experiments in Section 3). In the scenario of cold-start, a new reviewer only has a behavior: post a review. As a result, we can not get effective behavioral features from the data. Although, the linguistic features of reviews do not need to take much time to form, Mukherjee et al. (2013c) have proved that the linguistic features are not effective enough in detecting real-life fake reviews from the commercial websites, where we also obtain the same observation (the details are shown in Section 3).

[1] http://www.bbc.com/news/technology-24299742

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 366–376
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1034

Therefore, the main difficulty of the cold-start spam problem is that there are no sufficient behaviors of the new reviewers for constructing effective behavioral features. Nevertheless, there is ample textual and behavioral information contained in the abundant reviews posted by the existing reviewers (Figure 1). We could employ behavioral information of existing similar reviewers to a new reviewer to approximate his behavioral features. We argue that a reviewer's individual characteristics such as background information, motivation, and interactive behavior style have a great influence on a reviewer's textual and behavioral information. So the textual information and the behavioral information of a reviewer are correlated with each other (similar argument in Li et al. (2016)). For example, the students of the college are likely to choose the youth hostel during summer vacation and tend to comment the room price in their reviews. But the financial analysts on a business trip may tend to choose the business hotel, the environment and service are what they care about in their reviews.

To augment the behavioral information of the new reviewers in the cold-start problem, we first try to find the textual information which is similar with that of the new reviewer, from the existing reviews. There are several ways to model the textual information of the review spam, such as Unigram (Mukherjee et al., 2013c), POS (Ott et al., 2011) and LIWC (Linguistic Inquiry and Word Count) (Newman et al., 2003). We employ the CNN (Convolutional Neural Network) to model the review text, which has been proved that it can capture complex global semantic information that is difficult to express using traditional discrete manual features (Ren and Zhang, 2016). Then we employ the behavioral information which is correlated with the found textual information to approximate the behavioral information of the new reviewer. An intuitive approach is to search the most similar existing review for the new review, then take the found reviewer's behavioral features as the new reviewers' features (detailed in Section 5.3). However, there is abundant behavioral information in the review graph (Figure 1), it is difficult for the traditional discrete manual behavioral features to record the global behavioral information (Wang et al., 2016). Moreover, the traditional features can not capture the reviewer's individual characteristics, because there is no explicit characteristic tag available in the review system (experiments in Section 5.3). So, we propose a neural network model to jointly encode the textual and behavioral information into the review embeddings for detecting the review spam in the cold-start problem. By encoding the review graph structure (Figure 1), the proposed model can record the global footprints of the existing reviewers in an unsupervised way, and further record the reviewers' latent characteristic information in the footprints. The jointly learnt review embeddings can model the correlation of the reviewers' textual and behavioral information. When a new reviewer posts a review, the proposed model can represent the review with the similar textual information and the correlated behavioral information encoded in the word embeddings. Finally, the embeddings of the new review are fed into a classifier to identify whether it is spam or not.

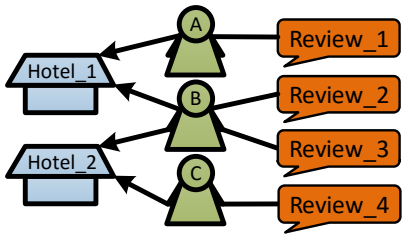


Figure 1: Part of review graph simplified from Yelp.

In summary, our major contributions include:

- To our best knowledge, this is the first work that explores the cold-start problem in review spam detection. We qualitatively and quantitatively prove that the traditional linguistic and behavioral features are not effective enough in detecting review spam for the cold-start task.
- We propose a neural network model to jointly encode the textual and behavioral information into the review embeddings for the cold-start spam detection task. It is an unsupervised distributional representation model which can learn from large scale unlabeled review data.
- Experimental results on two domains (hotel and restaurant) give good confidence that the proposed model performs effectively in the cold-start spam detection task.

2 Related Work

Jindal and Liu (2008) make the first step to detect review spam. Subsequent work devoted most

efforts to explore effective features and spammer-like clues.

Linguistic features: Ott et al. (2011) applied psychological and linguistic clues to identify review spam; Harris (2012) explored several writing style features. Syntactic stylometry for review spam detection was investigated in Feng et al. (2012a); Xu and Zhao (2012) using deep linguistic features for finding deceptive opinion spam; Li et al. (2013) studied the topics in the review spam; Li et al. (2014b) further analyzed the general difference of language usage. Fornaciari and Poesio (2014) proved the effectiveness of the N-grams in detecting deceptive Amazon book reviews. The effectiveness of the N-grams was also explored in Cagnina and Rosso (2015). Li et al. (2014a) proposed a positive-unlabeled learning method based on unigrams and bigrams; Kim et al. (2015) carried out a frame-based deep semantic analysis. Hai et al. (2016) exploited the relatedness of multiple review spam detection tasks and available unlabeled data to address the scarcity of labeled opinion spam data by using linguistic features. Besides, (Ren and Zhang, 2016) proved that the CNN model is more effective than the RNN and the traditional discrete manual linguistic features. Hovy (2016) used N-gram generative models to produce reviews and evaluated their effectiveness.

Behavioral features: Lim et al. (2010) analyzed reviewers' rating behavioral features; Jindal et al. (2010) identified unusual review patterns which can represent suspicious behaviors of reviews; Li et al. (2011) proposed a two-view semi-supervised co-training method base on behavioral features. Feng et al. (2012b) study the distributions of individual spammers' behaviors. The group spammers' behavioral features were studied in Mukherjee et al. (2012). Temporal patterns of spammers were investigated by Xie et al. (2012), Fei et al. (2013); Li et al. (2015) explored the temporal and spatial patterns. The review graph was analyzed by Wang et al. (2011), Akoglu et al. (2013); Mukherjee et al. (2013a) studied the spamicity of reviewers. Mukherjee et al. (2013c), Mukherjee et al. (2013b) proved that reviewers' behavioral features are more effective than reviews' linguistic features for detecting review spam. Based on this conclusion, recently, researchers (Rayana and Akoglu, 2015; KC and Mukherjee, 2016) have put more efforts in employing reviewers' behavioral features for detecting review spam, the intuition behind which is to capture the reviewers' actions and supposes that those reviews written with spammer-like behaviors would be spam. Wang et al. (2016) explored a method to learn the review representation with global behavioral information. Viviani and Pasi (2017) concentrated on the aggregation process with respect to each single veracity feature.

Features	P	R	F1	A
LF	54.5	71.1	61.7	55.9
LF+BF	63.4	52.6	57.5	61.1
LF+BF_abundant	69.1	63.5	66.2	67.5

(a) Hotel

Features	P	R	F1	A
LF	53.8	80.8	64.6	55.8
LF+BF	58.1	61.2	59.6	58.5
LF+BF_abundant	56.6	78.2	65.7	59.1

(b) Restaurant

Table 1: SVM classification results across linguistic features (LF, bigrams here (Mukherjee et al., 2013b)), behavioral features (BF: RL, RD, MCS (Mukherjee et al., 2013b)) and behavioral features with abundant behavioral information (BF_abundant). Both training and testing use balanced data (50:50).

3 Whether Traditional Features are Effective

As a new reviewer posted just one review and we have to identify it immediately, the major challenge of the cold-start task is that the available information about the new reviewer is very poor. The new reviewer only provides us with one review record. For most traditional features based on the statistics, they can not form themselves or make no sense, such as the *percentage of reviews written at weekends* (Li et al., 2015), the *entropy of rating distribution of user's review* (Rayana and Akoglu, 2015). To investigate whether traditional features are effective in the cold-start task, we conducted experiments on the Yelp dataset in Mukherjee et al. (2013c). We trained SVM models with different features on the existing reviews posted before January 1, 2012, and tested on the new reviews which just posted by the new reviewers after January 1, 2012. Results are shown in Table 1.

3.1 Linguistic Features' Poor Performance

The linguistic features need not take much time to form. But Mukherjee et al. (2013c) have proved that the linguistic features are not effective enough in detecting real-life fake reviews from the commercial websites, compared with the performances on the crowd source datasets (Ott et al., 2011). They showed that the word bigrams perform better than the other linguistic features, such as LIWC (Newman et al., 2003; Pennebaker et al., 2007), part-of-speech sequence patterns (Mukherjee and Liu, 2010), deep syntax (Feng et al., 2012a), information gain (Mukherjee et al., 2013c) and so on. So, we conduct experiments with the word bigrams feature. As shown in Table 1 (a, b) row 1, the word bigrams result in only around 55% in accuracy in both the hotel and restaurant domains. It indicates that the most effective traditional linguistic feature (i.e., the word bigrams) can't detect the review spam effectively in the cold start task.

3.2 Behavioral Features only Work Well with Abundant Information

Because there is not enough available information about the new reviewer, for most traditional behavioral features based on the statistical mechanism, they couldn't form themselves or make no sense. We investigated the previous work and found that there are three behavioral features can be applied to the cold-start task. They are proposed by Mukherjee et al. (2013b), i.e., 1.*Review length (RL)* : the length of the new review posted by the new reviewer; 2.*Reviewer deviation (RD)*: the absolute rating deviation of the new reviewer's review from other reviews on the same business; 3.*Maximum content similarity (MCS)* : the maximum content similarity (using cosine similarity) between the new reviewer's review with other reviews on the same business.

Table 1 (a, b) row 2 shows the experiment results by the combinations of the bigrams feature and the three behavioral features described above. The behavioral features make around 5% improvement in accuracy in the hotel domain (2.7% in the restaurant domain) as compared with only using bigrams. The accuracy is improved but it is just near 60% in average. It indicates that the traditional features are not effective enough with poor behavioral information. What's more, the behavioral features cause around 4.6% decrease in F1-score and around 19% decrease in Recall in both hotel and restaurant domains. It is obvious that there is more false-positive review spam caused by the behavioral features as compared to only using bigrams. It further indicates that the traditional behavioral features' discrimination for review spam gets to be weakened by the poor behavioral information.

To go a step further, we carried experiments with the three behavioral features which are formed on abundant behavioral information. When the new reviewers continue to post more reviews in after weeks, their behavioral information gets to be more. Then the review system could obtain sufficient data to extract behavior features as compared to the poor information in the cold-start period. So the behavioral features with abundant information make an obvious improvement in accuracy (6.4%) in the hotel domain (Table 1 (a) row 3) as compared with the results in Table 1 (a) row 2. But it is only 0.6% in the restaurant domain. By statistics on the datasets, we found that the new reviewers posted about 54.4 reviews in average after their first post in the hotel domain, but it is only 10 reviews in average for the new reviewers in the restaurant domain. The added behavioral information in the hotel domain is richer than that in the restaurant domain. It indicates that:

- the traditional behavioral features can only work well with abundant behavioral information;
- the more behavioral information can be obtained, the more effective the traditional behavioral features are.

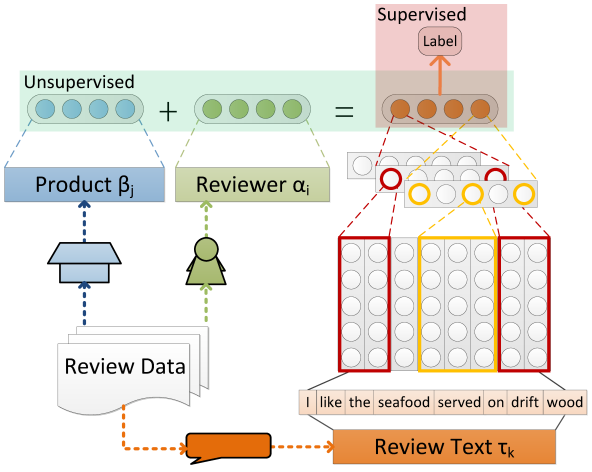


Figure 2: Illustration of our model.

4 The Proposed Model

The difficulty of detecting review spam in the cold-start task is that the available behavioral information of new reviewers is very poor. The new reviewer just posted one review and we have to filter it out immediately, there is not any historical review provided to us. As we argued, the textual information and the behavioral information of a reviewer are correlated with each other. So, to augment the behavioral information of new reviewers, we try to find the textual information which is similar with that of the new reviewer, from existing reviews. Then we take the behavioral information which is correlated with the found textual information as the most possible behavioral information of the new reviewer. For this purpose, we propose a neural network model to jointly encode the textual and behavioral information into the review embeddings for detecting the review spam in the cold-start problem (shown in Figure 2). When a new reviewer posts a review, the neural network can represent the review with the similar textual information and the correlated behavioral information encoded in the word embeddings. Finally, embeddings of the new review are fed into a classifier to identify whether it is spam or not.

4.1 Behavioral Information Encoding

In Figure 1, there is a part of review graph which is simplified from the Yelp website. As it shows, the review graph contains the global behavioral information (footprints) of the existing reviewers. Because the motivations of the spammers and the real reviewers are totally different, the distributions of the behavioral information of them are different (Mukherjee et al., 2013a). There are businesses (even highly reputable ones) paying people to write fake reviews for them to promote their products/services and/or to discredit their competitors (Liu, 2015). So the behavioral footprints of the spammers are decided by the demands of the businesses. But the real reviewers only post reviews to the product or services they have actually experienced. Their behavioral footprints are influenced by their own characteristics. Previous work extracts behavioral features for reviewers from these behavioral information. But it is impractical to the new reviewers in the cold-start task. Moreover, the traditional discrete features can not effectively record the global behavioral information (Wang et al., 2016). Besides, there is no explicit characteristic tag available in the review system, and we need to find a way to record the reviewers' latent characters information in footprints.

Therefore we encode these behavioral information into our model by utilizing an embedding learning model which is similar with TransE (Bordes et al., 2013). TransE is a model which can encode the graph structure, and represent the nodes and edges (head, translation/relation, tail) in low dimension vector space. TransE has been proved that it is good at describing the global information of the graph structure by the work about distributional representation for knowledge base (Guu et al., 2015). We consider that each reviewer in review graph describes the product in his/her own view and writes the review. When we represent the product, reviewer, and review in low dimension vector space, the reviewer embeddings can be taken as a translation vector, which has translated the product embeddings to the review embeddings. So, as shown in Figure 2, we take the products (hotels/restaurants) as the head part of the TransE network in our model, take the reviewers as the translation (relation) part and take the review as the tail part. By learning from the existing large scale unlabeled reviews of the review graph, we can encode the global behavioral information into our model without extracting any traditional behavioral feature, and record reviewers' latent characteristics information.

More formally, we minimize a margin-based criterion over the training set:

$$\mathcal{L} = \sum_{(\boldsymbol{\beta},\boldsymbol{\alpha},\boldsymbol{\tau})\in S} \sum_{(\boldsymbol{\beta}',\boldsymbol{\alpha},\boldsymbol{\tau}')\in S'} \max \{0, 1 + d(\boldsymbol{\beta}+\boldsymbol{\alpha},\boldsymbol{\tau}) - d(\boldsymbol{\beta}'+\boldsymbol{\alpha},\boldsymbol{\tau}')\} \quad (1)$$

S denotes the training set of triples $(\boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\tau})$ composed product $\boldsymbol{\beta}$ ($\boldsymbol{\beta} \in B$, products set (head part)), reviewer $\boldsymbol{\alpha}$ ($\boldsymbol{\alpha} \in A$, reviewers set (translation part)) and review text embeddings learnt by the CNN $\boldsymbol{\tau}$ ($\boldsymbol{\tau} \in T$, review texts set (tail part)).

$$S' = \{(\boldsymbol{\beta}', \boldsymbol{\alpha}, \boldsymbol{\tau}) | \boldsymbol{\beta}' \in B\} \cup \{(\boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\tau}') | \boldsymbol{\tau}' \in T\} \quad (2)$$

The set of corrupted triplets S' (Equation (2)), is composed of training triplets with either the product or review text replaced by a random chosen one (but not both at the same time).

$$d(\boldsymbol{\beta}+\boldsymbol{\alpha},\boldsymbol{\tau}) = \|\boldsymbol{\beta}+\boldsymbol{\alpha}-\boldsymbol{\tau}\|_2^2, \quad s.t. \|\boldsymbol{\beta}\|_2^2 = \|\boldsymbol{\alpha}\|_2^2 = \|\boldsymbol{\tau}\|_2^2 = 1 \quad (3)$$

Domain	Hotel	Restaurant
#reviews	688328	788471
#reviewers	5132	35593
date range	2004.10.23 2012.09.26	2004.10.12 2012.10.02
%before 2012.01.01	99.01%	97.40%

Table 2: Yelp Whole Dataset Statistics (Labeled and Unlabeled).

$d(\boldsymbol{\beta} + \boldsymbol{\alpha}, \boldsymbol{\tau})$ is the dissimilarity function with the squared euclidean distance.

4.2 Textual Information Encoding

To encode the textual information into our model, we adopt a convolutional neural network (CNN) to learn to represent the existing reviews. By statistics, we find that a review usually refers to several aspects of the products or services. For example, a hotel review may comment the room price, the free WiFi, and the bathroom at the same time. Compared with the recurrent neural network (RNN), the CNN can do a better job of modeling the different aspects of a review. Ren and Zhang (2016) have proved that the CNN can capture complex global semantic information and detect review spam more effectively, compared with traditional discrete manual features and the RNN model. As shown in Figure 2, we take the learnt embeddings τ of reviews by the CNN as the tail part.

Specifically, we denote the review text consisting of n words as $\{w_1, w_2, ..., w_n\}$, the word embeddings $\boldsymbol{e}(w_i) \in R^D$, D is the word vector dimension. We take the concatenation of the word embeddings in a fixed length window size Z as the input of the linear layer, which is denoted as $I_i \in R^{D \times Z}$. So the output of the linear layer H_i is calculated by $H_{k,i} = W_k \cdot I_i + b_i$, where $W_k \in R^{D \times Z}$ is the weight matrix of filter k. We utilize a max pooling layer to get the output of each filter. Then we take $tanh$ as the activation function and concatenate the outputs as the final review embeddings, which is denoted as $\boldsymbol{\tau}_i$.

4.3 Jointly Information Encoding

To model the correlation of the textual and behavioral information, we employ the jointly information encoding. By jointly learning from the global review graph, the textual and behavioral information of existing spammers and real reviewers are embedded into the word embeddings.

Domain	Hotel	Restaurant
fake	802	8368
non-fake	4876	50149
%fake	14.1%	14.3%
#reviews	5678	58517
#reviewers	5124	35593

Table 3: Yelp Labeled Dataset Statistics.

Dataset	Train	Test
date range	2004.10.23 2012.01.01	2012.01.01 2012.09.26
#reviews	1132	422

(a) Hotel

Dataset	Train	Test
date range	2004.10.12 2012.01.01	2012.01.01 2012.10.02
#reviews	14012	2368

(b) Restaurant

Table 4: The Balanced Datasets Statistics for Training and Testing the Classifier from Table 3.

In addition, the rating usually represents the sentiment polarity of a review, e.g., five star means 'like' and one star means 'dislike'. The spammers often review their target products with a low rating for discredited purpose, and with a high rating for promoted purpose. To encode the semantics of the sentiment polarity into the review embeddings, we learn the embeddings of 1-5 stars rating in our model at the same time. They are taken as the constraints of the review embeddings during the joint learning. They are calculated as:

$$\mathcal{C} = \sum_{(\boldsymbol{\tau},\boldsymbol{\gamma}) \in \Gamma} \sum_{(\boldsymbol{\tau},\boldsymbol{\gamma}') \in \Gamma'} \max\{0, 1 + g(\boldsymbol{\tau},\boldsymbol{\gamma}) - g(\boldsymbol{\tau},\boldsymbol{\gamma}')\} \quad (4)$$

The set of corrupted tuples Γ' is composed of training tuples Γ with the rating of review replaced by its opposite rating (i.e., 1 by 5, 2 by 4, 3 by 1 or 5). $g(\boldsymbol{\tau},\boldsymbol{\gamma}) = \|\boldsymbol{\tau} - \boldsymbol{\gamma}\|_2^2$, norm constraints: $\|\boldsymbol{\gamma}\|_2^2 = 1$.

The final joint loss function is as follows:

$$\mathcal{L}_{\mathcal{J}} = (1 - \theta)\mathcal{L} + \theta\mathcal{C} \quad (5)$$

where θ is a hyper-parameter.

Features	P	R	F1	A	
LF	54.5	71.1	61.7	55.9	1
LF+BF	63.4	52.6	57.5	61.1	2
BF_EditSim+LF	55.3	69.7	61.6	56.6	3
BF_W2Vsim+W2V	58.4	65.9	61.9	59.5	4
Ours_RE	62.1	68.3	**65.1**	**63.3**	5
Ours_RE+RRE+PRE	63.6	71.2	**67.2**	**65.3**	6

(a) Hotel

P	R	F1	A	
53.8	80.8	64.6	55.8	1
58.1	61.2	59.6	58.5	2
53.9	82.2	65.1	56.0	3
56.3	73.4	63.7	58.2	4
58.4	75.1	**65.7**	**60.8**	5
59.0	78.8	**67.5**	**62.0**	6

(b) Restaurant

Table 5: SVM classification results across linguistic features (LF, bigrams here (Mukherjee et al., 2013b)), behavioral features (BF: RL, RD, MCS (Mukherjee et al., 2013b)); the SVM classification results by the intuitive method that finding the most similar existing review by edit distance ratio and take the found reviewers' behavioral features as approximation (BF_EditSim+LF), and results by the intuitive method that finding the most similar existing review by averaged pre-trained word embeddings (using Word2Vec) (BF_W2Vsim+W2V); and the SVM classification results across the learnt review embeddings (RE), the learnt review's rating embeddings (RRE), the learnt product's average rating embeddings (PRE) by our model. Improvements of our model are statistically significant with $p<0.005$ based on paired *t*-test.

5 Experiments

5.1 Datasets and Evaluation Metrics

Datasets: To evaluate the proposed method, we conducted experiments on Yelp dataset that was used in (Mukherjee et al., 2013b,c; Rayana and Akoglu, 2015). The statistics of the Yelp dataset are listed in Table 2 and Table 3. The reviewed product here refers to a hotel or restaurant. We take the existing reviews posted before January 1, 2012 as the datasets for training our embedding learning model, and take the first new reviews which just posted by the new reviewers after January 1, 2012 as the test datasets. Table 4 displays the statistics of the balanced datasets for training and testing the classifier.

Evaluation Metrics: We select precision (P), recall (R), F1-Score (F1), accuracy (A) as metrics.

5.2 Our Model v.s. the Traditional Features

To illustrate the effectiveness of our model, we conduct experiments on the public datasets, and make comparison with the most effective traditional linguistic features, e.g., bigrams, and the three practicable traditional behavioral features (RL, RD, MCS (Mukherjee et al., 2013b)) referred in Section 3.2. The results are shown in Table 5. For our model, we set the dimension of embeddings to 100, the number of CNN filters to 100, θ to 0.1, Z to 2. The hyper-parameters are tuned by grid search on the development dataset. The product and reviewer embeddings are randomly initialized from a uniform distribution (Socher et al., 2013). The word embeddings are initialized with 100-dimensions vectors pre-trained by the CBOW model (Word2Vec) (Mikolov et al., 2013). As Table 5 showed, our model observably performs better in detecting review spam for the cold-start task in both hotel and restaurant domains.

Review Embeddings Compared with the traditional linguistic features, e.g., bigrams, using the review embeddings learnt by our model, results in around 3.4% improvement in F1 and around 7.4% improvement in A in the hotel domain (1.1% in F1 and 5.0% in A for the restaurant domain, shown in Tabel 5 (a,b) rows 1, 5). Compared with the combination of the bigrams and the traditional behavioral features, using the review embeddings learnt by our model, results in around 7.6% improvement in F1 and around 2.2% improvement in A in the hotel domain (6.1% in F1 and 2.3% in A for the restaurant domain, shown in Tabel 5 (a,b) rows 2, 5). The F1-Score (F1) of the classification under the balance distribution reflects the ability to detect the review spam. The accuracy (A) of the classification under the balance distribution reflects the ability to identify both the review spam and the real review. The experiment results indicate that our model performs significantly better than the traditional methods in F1 and A at the same time. The learnt review embeddings with encoded linguistic and behavioral information are more effective in detecting review

Features	P	R	F1	A	
LF	54.5	71.1	61.7	55.9	1
Ours_CNN	61.2	51.7	56.1	59.5	2
Ours_RE	62.1	68.3	**65.1**	**63.3**	3

(a) Hotel

P	R	F1	A	
53.8	80.8	64.6	55.8	1
56.9	58.8	57.8	57.1	2
58.4	75.1	**65.7**	**60.8**	3

(b) Restaurant

Table 6: SVM classification results across linguistic features (LF, bigrams here (Mukherjee et al., 2013b)), the learnt review embeddings (RE) ; and the classification results by only using our CNN. Both training and testing use balanced data (50:50). Improvements of our model are statistically significant with $p<0.005$ based on paired *t*-test.

spam for the cold-start task.

Rating Embeddings As we referred in Section 4.3, the rating of a review usually means the sentiment polarity of a real reviewer or the motivation of a spammer. As shown in Table 5 (a,b) rows 6, adding the rating embeddings of the products (hotel/restaurant) and reviews renders even higher F1 and A. We suppose that different rating embeddings are encoded with different semantic meanings. They reflect the semantic divergences between the average rating of the product and the review rating. In results, using RE+RRE+PRE which makes the best performance of our model, results in around 5.5% improvement in F1 and around 9.4% improvement in A in the hotel domain (2.9% in F1 and 6.2% in A for the restaurant domain, shown in Tabel 5 (a,b) rows 1, 6), compared with the LF. Using RE+RRE+PRE results in around 9.7% improvement in F1 and around 4.2% improvement in A in the hotel domain (7.9% in F1 and 3.5% in A for the restaurant domain, shown in Tabel 5 (a,b) rows 2, 6), compared with the LF+BF.

The experiment results prove that our model is effective. The improvements in both the F1 and A prove that our model performs well in both detecting the review spam and identifying the real review. Furthermore, the improvements in both the hotel and restaurant domains prove that our model possesses preferable domain-adaptability [2]. It can learn to represent the reviews with global linguistic and behavioral information from large-scale unlabeled existing reviews.

[2]The improvements in hotel domain are greater than that in restaurant domain. The possible reason is the proportion of the available training data in hotel domain is higher than that in restaurant domain (99.01% vs. 97.40% in Table 2).

5.3 Our Jointly Embeddings v.s. the Intuitive Methods

As mentioned in Section 1, to approximate the behavioral information of the new reviewers, there are other intuitive methods. So we conduct experiments with two intuitive methods as a comparison. One is finding the most similar existing review by edit distance ratio and taking the found reviewers' behavioral features as an approximation, and then training the classifier on the behavioral features and bigrams (BF_EditSim+LF). The other is finding the most similar existing review by cosine similarity of review embeddings which is the average of the pre-trained word embeddings (using Word2Vec), and then training the classifier on the behavioral features and review embeddings (BF_W2Vsim+W2V). As shown in Table 5, our joint embeddings (Ours_RE and Ours_RE+RRE+PRE) obviously perform better than the intuitive methods, such as the Ours_RE is 3.8% (Accuracy) and 3.2% (F1) better than BF_W2Vsim+W2V in the hotel domain. The experiments indicate that our joint embeddings do a better job in capturing the reviewer's characteristics and modeling the correlation of textual and behavioral information.

5.4 The Effectiveness of Encoding the Global Behavioral Information

To further evaluate the effectiveness of encoding the global behavioral information in our model, we build an independent supervised convolutional neural network which has the same structure and parameter settings with the CNN part of our model. There is not any review graphic or behavioral information in this independent supervised CNN (Tabel 6 (a,b) row 2). As shown in Tabel 6 (a,b) rows 2, 3, compared with the review embeddings learnt by the independent supervised CNN, using

the review embeddings learnt by our model results in around 9.0% improvement in F1 and around 3.8% improvement in A in the hotel domain (7.9% in F1 and 3.7% in A for the restaurant domain). The results show that our model can represent the new reviews posted by the new reviewers with the correlated behavioral information encoded in the word embeddings. The transE part of our model has effectively recorded the behavioral information of the review graph. Thus, our model is more effective by jointly embedding the textual and behavioral informations, it helps to augment the possible behavioral information of the new reviewer.

5.5 The Effectiveness of CNN

Compared with the the most effective linguistic features, e.g., bigrams, our independent supervised convolutional neural network performs better in A than F1 (shown in Tabel 5 (a,b) rows 1, 2). It indicates that the CNN do a better job in identifying the real review than the review spam. We suppose that the possible reason is that the CNN is good at modeling the different semantic aspects of a review. And the real reviewers usually tend to describe different aspects of a hotel or restaurant according to their real personal experiences, but the spammers can only forge fake reviews with their own infinite imagination. Mukherjee et al. (2013b) also proved that different psychological states of the minds of the spammers and non-spammers, lead to significant linguistic differences between review spam and non-spam.

6 Conclusion and Future Work

This paper analyzes the importance and difficulty of the cold-start challenge in review spam combat. We propose a neural network model that jointly embeds the existing textual and behavioral information for detecting review spam in the cold-start task. It can learn to represent the new review of the new reviewer with the similar textual information and the correlated behavioral information in an unsupervised way. Then, a classifier is applied to detect the review spam. Experimental results prove the proposed model achieves an effective performance and possesses preferable domain-adaptability. It is also applicable to a large-scale dataset in an unsupervised way. To our best knowledge, this is the first work to handle the cold-start problem in review spam detection. We are going to explore more effective models in future.

Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 61533018) and the National Basic Research Program of China (No. 2014CB340503). And this research work was also supported by Google through focused research awards program. We would like to thank Prof. Bing Liu for useful advice, and the anonymous reviewers for their detailed comments and suggestions.

References

Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion fraud detection in online reviews by network effects. *ICWSM* 13:2–11.

Michael Anderson and Jeremy Magruder. 2012. Learning from the crowd: Regression discontinuity estimates of the effects of an online review database*. *The Economic Journal* 122(563):957–989.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. pages 2787–2795.

Leticia Cagnina and Paolo Rosso. 2015. *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Association for Computational Linguistics, chapter Classification of deceptive opinions using a low dimensionality representation, pages 58–66. https://doi.org/10.18653/v1/W15-2909.

Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Exploiting burstiness in reviews for review spammer detection. In *ICWSM*. Citeseer.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012a. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 171–175. http://aclweb.org/anthology/P12-2034.

Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. 2012b. Distributional footprints of deceptive product reviews. In *ICWSM*.

Tommaso Fornaciari and Massimo Poesio. 2014. Identifying fake amazon reviews as learning from crowds. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 279–287. https://doi.org/10.3115/v1/E14-1030.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 318–327. https://doi.org/10.18653/v1/D15-1038.

Zhen Hai, Peilin Zhao, Peng Cheng, Peng Yang, Xiao-Li Li, and Guangxia Li. 2016. Deceptive review spam detection via exploiting task relatedness and unlabeled data. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1817–1826. http://aclweb.org/anthology/D16-1187.

C Harris. 2012. Detecting deceptive opinion spam using human computation. In *Workshops at AAAI on Artificial Intelligence*.

Dirk Hovy. 2016. The enemy in your own camp: How well can we detect statistically-generated fake reviews–an adversarial study. In *The 54th Annual Meeting of the Association for Computational Linguistics*. page 351. http://www.aclweb.org/anthology/385.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the First WSDM*. ACM, pages 219–230.

Nitin Jindal, Bing Liu, and Ee-Peng Lim. 2010. Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th CIKM*. ACM, pages 1549–1552.

Santosh KC and Arjun Mukherjee. 2016. On the temporal dynamics of opinion spamming: Case studies on yelp. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 369–379.

Seongsoon Kim, Hyeokyoon Chang, Seongwoon Lee, Minhwan Yu, and Jaewoo Kang. 2015. Deep semantic frame-based deceptive opinion spam analysis. In *Proceedings of the 24th CIKM*. ACM, pages 1131–1140.

Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. 2011. Learning to identify review spam. In *IJCAI Proceedings*. volume 22, page 2488.

Huayi Li, Zhiyuan Chen, Arjun Mukherjee, Bing Liu, and Jidong Shao. 2015. Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In *Ninth International AAAI Conference on Web and Social Media*.

Huayi Li, Bing Liu, Arjun Mukherjee, and Jidong Shao. 2014a. Spotting fake reviews using positive-unlabeled learning. *Computación y Sistemas* 18(3):467–475.

Jiwei Li, Claire Cardie, and Sujian Li. 2013. Topicspam: a topic-model based approach for spam detection. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 217–221. http://aclweb.org/anthology/P13-2039.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155* .

Jiwei Li, Myle Ott, Claire Cardie, and Eduard Hovy. 2014b. Towards a general rule for identifying deceptive opinion spam. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1566–1576. https://doi.org/10.3115/v1/P14-1147.

Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. 2010. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th CIKM*. ACM, pages 939–948.

Bing Liu. 2015. *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press.

Michael Luca. 2011. Reviews, reputation, and revenue: The case of yelp. com. *Com (September 16, 2011). Harvard Business School NOM Unit Working Paper* (12-016).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013a. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD*. ACM, pages 632–640.

Arjun Mukherjee and Bing Liu. 2010. Improving gender classification of blog authors. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 207–217. http://aclweb.org/anthology/D10-1021.

Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st WWW*. ACM, pages 191–200.

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013b. Fake review detection: Classification and analysis of real and pseudo reviews. Technical report, Technical Report UIC-CS-2013-03, University of Illinois at Chicago.

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie S Glance. 2013c. What yelp fake review filter might be doing? In *ICWSM*.

Matthew L Newman, James W Pennebaker, Diane S Berry, and Jane M Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and social psychology bulletin* 29(5):665–675.

Myle Ott, Yejin Choi, Claire Cardie, and T. Jeffrey Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 309–319. http://aclweb.org/anthology/P11-1032.

JW Pennebaker, CK Chung, M Ireland, A Gonzales, and RJ Booth. 2007. The development and psychometric properties of liwc2007. austin, tx.

Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pages 985–994.

Yafeng Ren and Yue Zhang. 2016. Deceptive opinion spam detection using neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 140–150. http://aclweb.org/anthology/C16-1014.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, D. Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1631–1642. http://aclweb.org/anthology/D13-1170.

Marco Viviani and Gabriella Pasi. 2017. Quantifier guided aggregation for the veracity assessment of online reviews. *International Journal of Intelligent Systems* 32(5):481–501.

Guan Wang, Sihong Xie, Bing Liu, and Philip S Yu. 2011. Review graph based online store review spammer detection. In *Proceedings of the 11th ICDM*. IEEE, pages 1242–1247.

Xuepeng Wang, Kang Liu, Shizhu He, and Jun Zhao. 2016. Learning to represent review with tensor decomposition for spam detection. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 866–875. http://aclweb.org/anthology/D16-1083.

Sihong Xie, Guan Wang, Shuyang Lin, and Philip S Yu. 2012. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th KDD*. ACM, pages 823–831.

Qiongkai Xu and Hai Zhao. 2012. Using deep linguistic features for finding deceptive opinion spam. In *Proceedings of COLING 2012: Posters*. The COLING 2012 Organizing Committee, pages 1341–1350. http://aclweb.org/anthology/C12-2131.

Learning Cognitive Features from Gaze Data for Sentiment and Sarcasm Classification using Convolutional Neural Network

Abhijit Mishra[†], **Kuntal Dey**[†], **Pushpak Bhattacharyya**[⋆]
[†]IBM Research, India
[⋆]Indian Institute of Technology Bombay, India
[†]{abhijimi, kuntadey}@in.ibm.com
[⋆]pb@cse.iitb.ac.in

Abstract

Cognitive NLP systems- *i.e.*, NLP systems that make use of behavioral data - augment traditional text-based features with cognitive features extracted from eye-movement patterns, EEG signals, brain-imaging *etc.*. Such extraction of features is typically manual. We contend that manual extraction of features may not be the best way to tackle text subtleties that characteristically prevail in complex classification tasks like *sentiment analysis* and *sarcasm detection*, and that even the extraction and choice of features should be delegated to the learning system. We introduce a framework to automatically extract cognitive features from the *eye-movement / gaze* data of human readers reading the text and use them as features along with textual features for the tasks of sentiment polarity and sarcasm detection. Our proposed framework is based on Convolutional Neural Network (CNN). The CNN *learns* features from both gaze and text and uses them to classify the input text. We test our technique on published sentiment and sarcasm labeled datasets, enriched with gaze information, to show that using a combination of automatically learned text and gaze features often yields better classification performance over (i) CNN based systems that rely on text input alone and (ii) existing systems that rely on handcrafted gaze and textual features.

1 Introduction

Detection of sentiment and sarcasm in user-generated short reviews is of primary importance for social media analysis, recommendation and dialog systems. Traditional sentiment analyzers and sarcasm detectors face challenges that arise at *lexical*, *syntactic*, *semantic* and *pragmatic* levels (Liu and Zhang, 2012; Mishra et al., 2016c). Feature-based systems (Akkaya et al., 2009; Sharma and Bhattacharyya, 2013; Poria et al., 2014) can aptly handle lexical and syntactic challenges (*e.g.* learning that the word *deadly* conveys a strong positive sentiment in opinions such as *Shane Warne is a deadly bowler*, as opposed to *The high altitude Himalayan roads have deadly turns*). It is, however, extremely difficult to tackle subtleties at semantic and pragmatic levels. For example, the sentence *I really love my job. I work 40 hours a week to be this poor.* requires an NLP system to be able to understand that the opinion holder has not expressed a positive sentiment towards her / his job. In the absence of explicit clues in the text, it is difficult for automatic systems to arrive at a correct classification decision, as they often lack external knowledge about various aspects of the text being classified.

Mishra et al. (2016b) and Mishra et al. (2016c) show that NLP systems based on cognitive data (or simply, *Cognitive NLP* systems) , that leverage eye-movement information obtained from human readers, can tackle the semantic and pragmatic challenges better. The hypothesis here is that human gaze activities are related to the cognitive processes in the brain that combine the "external knowledge" that the reader possesses with textual clues that she / he perceives. While incorporating behavioral information obtained from gaze-data in NLP systems is intriguing and quite plausible, especially due to the availability of low cost eye-tracking machinery (Wood and Bulling, 2014; Yamamoto et al., 2013), few methods exist for text classification, and they rely on handcrafted features extracted from gaze data (Mishra et al., 2016b,c). These systems have limited capabilities due to two reasons: (a) Manually designed gaze based features may not adequately

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 377–387
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1035

capture all forms of textual subtleties (b) Eye-movement data is not as intuitive to analyze as text which makes the task of designing manual features more difficult. So, in this work, **instead of handcrafting the gaze based and textual features, we try to learn feature representations from both gaze and textual data using Convolutional Neural Network (CNN)**. We test our technique on two publicly available datasets enriched with eye-movement information, used for *binary classification* tasks of sentiment polarity and sarcasm detection. Our experiments show that the automatically extracted features often help to achieve significant classification performance improvement over (a) existing systems that rely on handcrafted gaze and textual features and (b) CNN based systems that rely on text input alone. The datasets used in our experiments, resources and other relevant pointers are available at http://www.cfilt.iitb.ac.in/cognitive-nlp

The rest of the paper is organized as follows. Section 2 discusses the motivation behind using readers' eye-movement data in a text classification setting. In Section 3, we argue why CNN is preferred over other available alternatives for feature extraction. The CNN architecture is proposed and discussed in Section 4. Section 5 describes our experimental setup and results are discussed in Section 6. We provide a detailed analysis of the results along with some insightful observations in Section 7. Section 8 points to relevant literature followed by Section 9 that concludes the paper.

Terminology

A *fixation* is a relatively long stay of gaze on a visual object (such as words in text) where as a *sacccade* corresponds to quick shifting of gaze between two positions of rest. Forward and backward saccades are called *progressions* and *regressions* respectively. A *scanpath* is a line graph that contains fixations as nodes and saccades as edges.

2 Eye-movement and Linguistic Subtleties

Presence of linguistic subtleties often induces (a) surprisal (Kutas and Hillyard, 1980; Malsburg et al., 2015), due to the underlying disparity /context incongruity or (b) higher cognitive load (Rayner and Duffy, 1986), due to the presence of lexically and syntactically complex structures. While surprisal accounts for irregular saccades (Malsburg et al., 2015), higher cognitive load results in longer fixation duration (Kliegl et al., 2004).

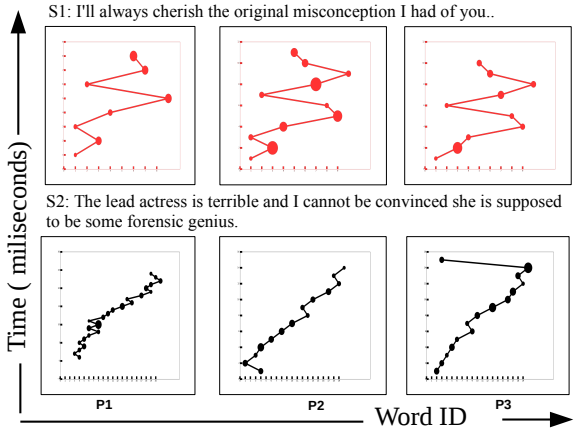


Figure 1: Scanpaths of three participants for two sentences (Mishra et al., 2016b). Sentence *S1* is sarcastic but *S2* is not. Length of the straight lines represents saccade distance and size of the circles represents fixation duration

Mishra et al. (2016b) find that presence of sarcasm in text triggers either *irregular saccadic patterns* or *unusually high duration fixations* than non-sarcastic texts (illustrated through example scanpath representations in Figure 1). For sentiment bearing texts, highly subtle eye-movement patterns are observed for semantically/pragmatically complex negative opinions (expressing irony, sarcasm, thwarted expectations, etc.) than the simple ones (Mishra et al., 2016b). The association between linguistic subtleties and eye-movement patterns could be captured through sophisticated feature engineering that considers both gaze and text inputs. In our work, CNN takes the onus of feature engineering.

3 Why Convolutional Neural Network?

CNNs have been quite effective in learning *filters* for image processing tasks, filters being used to transform the input image into more informative feature space (Krizhevsky et al., 2012). Filters learned at various CNN layers are quite similar to handcrafted filters used for detection of edges, contours, and removal of redundant backgrounds. We believe, a similar technique can also be applied to eye-movement data, where the learned filters will, hopefully, extract informative cognitive features. For instance, for sarcasm, we expect the network to learn filters that detect long distance saccades (refer to Figure 2 for an analogical il-

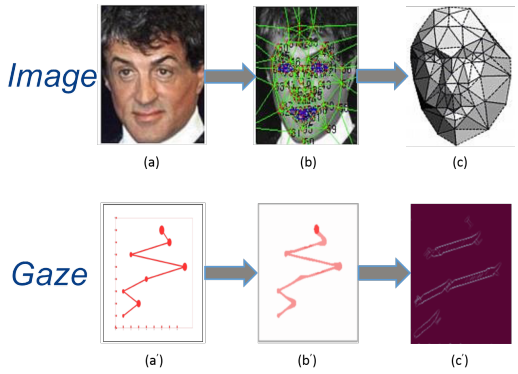


Figure 2: Illustrative analogy between CNN applied to images and scanpath representations showing why CNN can be useful for learning features from gaze patterns. Images partially taken from Taigman et al. (2014)

lustration). With more number of convolution filters of different dimensions, the network may extract multiple features related to different gaze attributes (such as fixations, progressions, regressions and skips) and will be free from any form of human bias that manually extracted features are susceptible to.

4 Learning Feature Representations: The CNN Architecture

Figure 3 shows the CNN architecture with two components for processing and extracting features from text and gaze inputs. The components are explained below.

4.1 Text Component

The text component is quite similar to the one proposed by Kim (2014) for sentence classification. Words (in the form of *one-hot* representation) in the input text are first replaced by their embeddings of dimension K (i^{th} word in the sentence represented by an embedding vector $x_i \in \mathbb{R}^K$). As per Kim (2014), a multi-channel variant of CNN (referred to as MULTICHANNELTEXT) can be implemented by using two channels of embeddings- one that remains static throughout training (referred to as STATICTEXT), and the other one that gets updated during training (referred to as NON-STATICTEXT). We separately experiment with static, non-static and multi-channel variants.

For each possible input channel of the text component, a given text is transformed into a tensor of fixed length N (padded with *zero-tensors* wherever necessary to tackle length variations) by concatenating the word embeddings.

$$x_{1:N} = x_1 \oplus x_2 \oplus x_3 \oplus ... \oplus x_N \quad (1)$$

where $\oplus$ is the concatenation operator. To extract *local features*[1], convolution operation is applied. Convolution operation involves a *filter*, $W \in \mathbb{R}^{HK}$, which is convolved with a window of H embeddings to produce a local feature for the H words. A local feature, c_i is generated from a window of embeddings $x_{i:i+H-1}$ by applying a non linear function (such as a hyperbolic tangent) over the convoluted output. Mathematically,

$$c_i = f(W.x_{i:i+H-1} + b) \quad (2)$$

where $b \in \mathbb{R}$ is the *bias* and f is the non-linear function. This operation is applied to each possible window of H words to produce a feature map (c) for the window size H.

$$c = [c_1, c_2, c_3, ..., c_{N-H+1}] \quad (3)$$

A global feature is then obtained by applying *max pooling* operation[2] (Collobert et al., 2011) over the feature map. The idea behind *max-pooling* is to capture the most important feature - one with the highest value - for each feature map.

We have described the process by which one feature is extracted from one filter (red bordered portions in Figure 3 illustrate the case of $H = 2$). The model uses multiple filters for each filter size to obtain multiple features representing the text. In the MULTICHANNELTEXT variant, for a window of H words, the convolution operation is separately applied on both the embedding channels. Local features learned from both the channels are concatenated before applying *max-pooling*.

4.2 Gaze Component

The gaze component deals with scanpaths of multiple participants annotating the same text. Scanpaths can be pre-processed to extract two sequences[3] of gaze data to form separate channels of input: (1) A sequence of normalized[4] durations of fixations (in milliseconds) in the order in which

[1] features specific to a region in case of images or window of words in case of text

[2] *mean pooling* does not perform well.

[3] like text-input, gaze sequences are padded where necessary

[4] scaled across participants using min-max normalization to reduce subjectivity

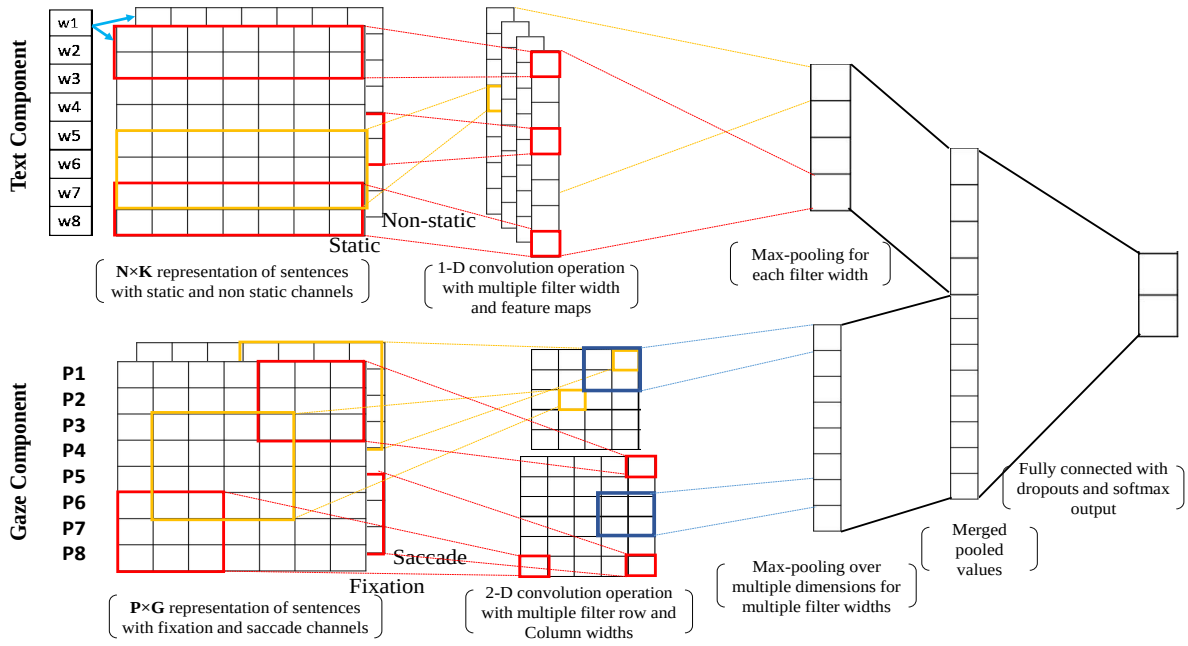


Figure 3: Deep convolutional model for feature extraction from both text and gaze inputs

they appear in the scanpath, and (2) A sequence of position of fixations (in terms of word id) in the order in which they appear in the scanpath. These channels are related to two fundamental gaze attributes such as fixation and saccade respectively. With two channels, we thus have three possible configurations of the gaze component such as (i) FIXATION, where the input is normalized fixation duration sequence, (ii) SACCADE, where the input is fixation position sequence, and (iii) MULTICHANNELGAZE, where both the inputs channels are considered.

For each possible input channel, the input is in the form of a $P \times G$ matrix (with $P \rightarrow$ number of participants and $G \rightarrow$ length of the input sequence). Each element of the matrix $g_{ij} \in \mathbb{R}$, with $i \in P$ and $j \in G$, corresponds to the j^{th} gaze attribute (either fixation duration or word id, depending on the channel) of the input sequence of the i^{th} participant. Now, unlike the text component, here we apply convolution operation across two dimensions *i.e.* choosing a two dimensional convolution filter $W \in \mathbb{R}^{JK}$ (for simplicity, we have kept $J = K$, thus , making the dimension of W, J^2). For the dimension size of J^2, a local feature c_{ij} is computed from the window of gaze elements $g_{ij:(i+J-1)(j+J-1)}$ by,

$$c_{ij} = f(W.g_{ij:(i+J-1)(j+J-1)} + b) \qquad (4)$$

where $b \in \mathbb{R}$ is the *bias* and f is a non-linear function. This operation is applied to each possible window of size J^2 to produce a feature map (c),

$$\begin{aligned} c = [&c_{11}, c_{12}, c_{13}, ..., c_{1(G-J+1)}, \\ &c_{21}, c_{22}, c_{23}, ..., c_{2(G-J+1)}, \\ &..., \\ &c_{(P-J+1)1}, c_{(P-J+1)2}, ..., c_{(P-J+1)(G-J+1)}] \end{aligned} \qquad (5)$$

A global feature is then obtained by applying *max pooling* operation. Unlike the text component, max-pooling operator is applied to a 2D window of local features size $M \times N$ (for simplicity, we set $M = N$, denoted henceforth as M^2). For the window of size M^2, the pooling operation on c will result in as set of global features $\hat{c_J} = max\{c_{ij:(i+M-1)(j+M-1)}\}$ for each possible i, j.

We have described the process by which one feature is extracted from one filter (of 2D window size J^2 and the max-pooling window size of M^2). In Figure 3, red and blue bordered portions illustrate the cases of $J^2 = [3, 3]$ and $M^2 = [2, 2]$ respectively. Like the text component, the gaze component also uses multiple filters for each filter size to obtain multiple features representing the gaze input. In the MULTICHANNELGAZE variant, for a 2D window of J^2, the convolution operation is separately applied on both fixation duration and saccade channels and local features learned from both the channels are concatenated before max-pooling is applied.

Once the global features are learned from both the text and gaze components, they are *merged*

and passed to a fully connected feed forward layer (with number of units set to 150) followed by a *SoftMax* layer that outputs the the probabilistic distribution over the class labels.

The gaze component of our network is not invariant of the order in which the scanpath data is given as input- *i.e.*, the P rows in the $P \times G$ can not be shuffled, even if each row is independent from others. The only way we can think of for addressing this issue is by applying convolution operations to all $P \times G$ matrices formed with all the permutations of P, capturing every possible ordering. Unfortunately, this makes the training process significantly less scalable, as the number of model parameters to be learned becomes huge. As of now, training and testing are carried out by keeping the order of the input constant.

5 Experiment Setup

We now share several details regarding our experiments below.

5.1 Dataset

We conduct experiments for two binary-classification tasks of sentiment and sarcasm using two publicly available datasets enriched with eye-movement information. Dataset 1 has been released by Mishra et al. (2016a). It contains 994 text snippets with 383 positive and 611 negative examples. Out of the 994 snippets, 350 are sarcastic. Dataset 2 has been used by Joshi et al. (2014) and it consists of 843 snippets comprising movie reviews and normalized tweets out of which 443 are positive, and 400 are negative. Eye-movement data of 7 and 5 readers is available for each snippet for dataset 1 and 2 respectively.

5.2 CNN Variants

With text component alone we have three variants such as STATICTEXT, NONSTATICTEXT and MULTICHANNELTEXT (refer to Section 4.1). Similarly, with gaze component we have variants such as FIXATION, SACCADE and MULTICHANNELGAZE (refer to Section 4.2). With both text and gaze components, 9 more variants could thus beexperimented with.

5.3 Hyper-parameters

For text component, we experiment with filter widths (H) of $[3, 4]$. For the gaze component, 2D filters (J^2) set to $[3 \times 3], [4 \times 4]$ respectively. The max pooling 2D window, M^2, is set to $[2 \times 2]$. In both gaze and text components, number of filters is set to 150, resulting in 150 feature maps for each window. These model hyper-parameters are fixed by trial and error and are possibly good enough to provide a first level insight into our system. Tuning of hyper-parameters might help in improving the performance of our framework, which is on our future research agenda.

5.4 Regularization

For regularization *dropout* is employed both on the embedding and the penultimate layers with a constraint on l_2-norms of the weight vectors (Hinton et al., 2012). Dropout prevents co-adaptation of hidden units by randomly dropping out - i.e., setting to zero - a proportion p of the hidden units during forward propagation. We set p to 0.25.

5.5 Training

We use ADADELTA optimizer (Zeiler, 2012), with a learning rate of 0.1. The input batch size is set to 32 and number of training iterations (epochs) is set to 200. 10% of the training data is used for validation.

5.6 Use of Pre-trained Embeddings:

Initializing the embedding layer with of pre-trained embeddings can be more effective than random initialization (Kim, 2014). In our experiments, we have used embeddings learned using the *movie reviews with one sentence per review* dataset (Pang and Lee, 2005). It is worth noting that, for a small dataset like ours, using a small data-set like the one from (Pang and Lee, 2005) helps in reducing the number model parameters resulting in faster learning of embeddings. The results are also quite close to the ones obtained using *word2vec* facilitated by Mikolov et al. (2013).

5.7 Comparison with Existing Work

For sentiment analysis, we compare our systems's accuracy (for both datasets 1 and 2) with Mishra et al. (2016c)'s systems that rely on handcrafted text and gaze features. For sarcasm detection, we compare Mishra et al. (2016b)'s sarcasm classifier with ours using dataset 1 (with available gold standard labels for sarcasm). We follow the same 10-fold train-test configuration as these existing works for consistency.

	Configuration	Dataset1			Dataset2		
		P	**R**	**F**	**P**	**R**	**F**
Traditional systems based on textual features	Näive Bayes	63.0	59.4	61.14	50.7	50.1	50.39
	Multi-layered Perceptron	69.0	69.2	69.2	66.8	66.8	66.8
	SVM (Linear Kernel)	72.8	73.2	72.6	70.3	70.3	70.3
Systems by Mishra et al. (2016c)	Gaze based (Best)	61.8	58.4	60.05	53.6	54.0	53.3
	Text + Gaze (Best)	**73.3**	**73.6**	**73.5**	**71.9**	**71.8**	**71.8**
CNN with only text input (Kim, 2014)	STATICTEXT	63.85	61.26	62.22	55.46	55.02	55.24
	NONSTATICTEXT	72.78	71.93	72.35	60.51	59.79	60.14
	MULTICHANNELTEXT	72.17	70.91	71.53	60.51	59.66	60.08
CNN with only gaze Input	FIXATION	60.79	58.34	59.54	53.95	50.29	52.06
	SACCADE	64.19	60.56	62.32	51.6	50.65	51.12
	MULTICHANNELGAZE	65.2	60.35	62.68	52.52	51.49	52
CNN with both text and gaze Input	STATICTEXT + FIXATION	61.52	60.86	61.19	54.61	54.32	54.46
	STATICTEXT + SACCADE	65.99	63.49	64.71	58.39	56.09	57.21
	STATICTEXT + MULTICHANNELGAZE	65.79	62.89	64.31	58.19	55.39	56.75
	NONSTATICTEXT + FIXATION	73.01	70.81	71.9	61.45	59.78	60.60
	NONSTATICTEXT + SACCADE	77.56	73.34	75.4	**65.13**	**61.08**	**63.04**
	NONSTATICTEXT + MULTICHANNELGAZE	**79.89**	**74.86**	**77.3**	63.93	60.13	62
	MULTICHANNELTEXT + FIXATION	74.44	72.31	73.36	60.72	58.47	59.57
	MULTICHANNELTEXT + SACCADE	**78.75**	**73.94**	**76.26**	63.7	60.47	62.04
	MULTICHANNELTEXT + MULTICHANNELGAZE	**78.38**	**74.23**	**76.24**	64.29	61.08	62.64

Table 1: Results for different traditional feature based systems and CNN model variants for the task of sentiment analysis. Abbreviations (P,R,F)→ Precision, Recall, F-score. SVM→Support Vector Machine

6 Results

In this section, we discuss the results for different model variants for sentiment polarity and sarcasm detection tasks.

6.1 Results for Sentiment Analysis Task

Table 1 presents results for sentiment analysis task. For dataset 1, different variants of our CNN architecture outperform the best systems reported by Mishra et al. (2016c), with a maximum F-score improvement of 3.8%. This improvement is statistically significant of $p < 0.05$ as confirmed by **McNemar test**. Moreover, we observe an F-score improvement of around 5% for CNNs with both gaze and text components as compared to CNNs with only text components (similar to the system by Kim (2014)), which is also statistically significant (with $p < 0.05$).

For dataset 2, CNN based approaches do not perform better than manual feature based approaches. However, variants with both text and gaze components outperform the ones with only text component (Kim, 2014), with a maximum F-score improvement of 2.9%. We observe that for dataset 2, training accuracy reaches 100 within 25 epochs with validation accuracy stable around 50%, indicating the possibility of overfitting. Tuning the regularization parameters specific to dataset 2 may help here. Even though CNN might not be proving to be a choice as good as handcrafted features for dataset 2, the bottom line remains that incorporation of gaze data into CNN consistently improves the performance over only-text-based CNN variants.

6.2 Results for Sarcasm Detection Task

For sarcasm detection, our CNN model variants outperform traditional systems by a maximum margin of 11.27% (Table 2). However, the improvement by adding the gaze component to the CNN network is just 1.34%, which is statistically insignificant over CNN with text component. While inspecting the sarcasm dataset, we observe a clear difference between the vocabulary of sarcasm and non-sarcasm classes in our dataset. This, perhaps, was captured well by the text component, especially the variant with only non-static embeddings.

7 Discussion

In this section, some important observations from our experiments are discussed.

7.1 Effect of Embedding Dimension Variation

Embedding dimension has proven to have a deep impact on the performance of neural systems (dos Santos and Gatti, 2014; Collobert et al., 2011).

	Configuration	P	R	F
Traditional systems based on textual features	Näive Bayes	69.1	60.1	60.5
	Multi-layered Perceptron	69.7	70.4	69.9
	SVM (Linear Kernel)	72.1	71.9	72
Systems by Riloff et al. (2013)	Text based (Ordered)	49	46	47
	Text + Gaze (Unordered)	46	41	42
System by Joshi et al. (2015)	Text based (best)	70.7	69.8	64.2
Systems by Mishra et al. (2016b)	Gaze based (Best)	73	73.8	73.1
	Text based (Best)	72.1	71.9	72
	Text + Gaze (Best)	76.5	75.3	75.7
CNN with only text input (Kim, 2014)	StaticText	67.17	66.38	66.77
	NonStaticText	84.19	**87.03**	85.59
	MultiChannelText	84.28	**87.03**	85.63
CNN with only gaze input	Fixation	74.39	69.62	71.93
	Saccade	68.58	68.23	68.40
	MultiChannelGaze	67.93	67.72	67.82
CNN with both text and gaze Input	StaticText + Fixation	72.38	71.93	72.15
	StaticText + Saccade	73.12	72.14	72.63
	StaticText + MultiChannelGaze	71.41	71.03	71.22
	NonStaticText + Fixation	**87.42**	85.2	86.30
	NonStaticText + Saccade	84.84	82.68	83.75
	NonStaticText + MultiChannelGaze	84.98	82.79	83.87
	MultiChannelText + Fixation	87.03	86.92	**86.97**
	MultiChannelText + Saccade	81.98	81.08	81.53
	MultiChannelText + MultiChannelGaze	83.11	81.69	82.39

Table 2: Results for different traditional feature based systems and CNN model variants for the task of sarcasm detection on dataset 1. Abbreviations (P,R,F)→ Precision, Recall, F-score

We repeated our experiments by varying the embedding dimensions in the range of [50-300][5] and observed that reducing embedding dimension improves the F-scores by a little margin. Small embedding dimensions are probably reducing the chances of over-fitting when the data size is small. We also observe that for different embedding dimensions, performance of CNN with both gaze and text components is consistently better than that with only text component.

7.2 Effect of Static / Non-static Text Channels

Non-static embedding channel has a major role in tuning embeddings for sentiment analysis by bringing adjectives expressing similar sentiment close to each other (*e.g, good and nice*), where as static channel seems to prevent over-tuning of embeddings (over-tuning often brings verbs like *love* closer to the pronoun *I* in embedding space, purely due to higher co-occurrence of these two words in sarcastic examples).

7.3 Effect of Fixation / Saccade Channels

For sentiment detection, saccade channel seems to be handing text having semantic incongruity (due to the presence of irony / sarcasm) better. Fixation channel does not help much, may be because of higher variance in fixation duration. For sarcasm detection, fixation and saccade channels perform with similar accuracy when employed separately. Accuracy reduces with gaze multichannel, may be because of higher variation of both fixations and saccades across sarcastic and non-sarcastic classes, as opposed to sentiment classes.

7.4 Effectiveness of the CNN-learned Features

To examine how good the features learned by the CNN are, we analyzed the features for a few example cases. Figure 4 presents some of the example test cases for the task of sarcasm detection. Example 1 contains sarcasm while examples 2, 3 and 4 are non-sarcastic. To see if there is any difference in the automatically learned features between text-only and combined text and gaze variants, we examine the feature vector (of dimension 150) for the examples obtained from different model variants. Output of the hidden layer after *merge* layer is considered as features learned by the network. We plot the features, in the form of color-bars, following Li et al. (2016) - denser col-

[5] a standard range (Liu et al., 2015; Melamud et al., 2016)

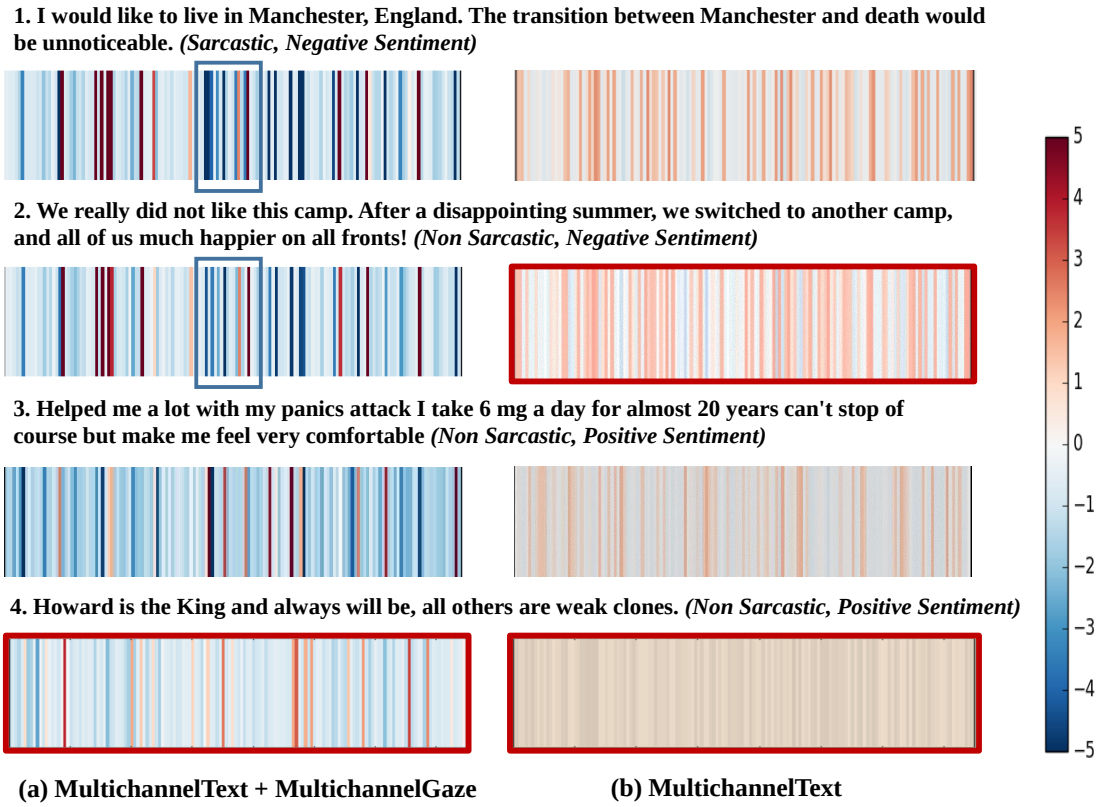


Figure 4: Visualization of representations learned by two variants of the network for sarcasm detection task. The output of the *Merge* layer (of dimension 150) are plotted in the form of colour-bars. Plots with thick red borders correspond to wrongly predicted examples.

ors representing feature with higher magnitude. In Figure 4, we show only two representative model variants *viz.,* MULTICHANNELTEXT and MULTICHANNELTEXT+ MULTICHANNELGAZE. As one can see, addition of gaze information helps to generate features with more subtle differences (marked by blue rectangular boxes) for sarcastic and non-sarcastic texts. It is also interesting to note that in the marked region, features for the sarcastic texts exhibit more intensity than the non-sarcastic ones - perhaps capturing the notion that *sarcasm typically conveys an intensified negative opinion*. This difference is not clear in feature vectors learned by text-only systems for instances like example 2, which has been incorrectly classified by MULTICHANNELTEXT. Example 4 is incorrectly classified by both the systems, perhaps due to lack of context. In cases like this, addition of gaze information does not help much in learning more distinctive features, as it becomes difficult for even humans to classify such texts.

8 Related Work

Sentiment and sarcasm classification are two important problems in NLP and have been the focus of research for many communities for quite some time. Popular sentiment and sarcasm detection systems are feature based and are based on unigrams, bigrams etc. (Dave et al., 2003; Ng et al., 2006), syntactic properties (Martineau and Finin, 2009; Nakagawa et al., 2010), semantic properties (Balamurali et al., 2011). For sarcasm detection, supervised approaches rely on **(a)** Unigrams and Pragmatic features (González-Ibánez et al., 2011; Barbieri et al., 2014; Joshi et al., 2015) **(b)** Stylistic patterns (Davidov et al., 2010) and patterns related to *situational disparity* (Riloff et al., 2013) and **(c)** Hastag interpretations (Liebrecht et al., 2013; Maynard and Greenwood, 2014). Recent systems are based on variants of deep neural network built on the top of embeddings. A few representative works in this direction for sentiment analysis are based on CNNs (dos Santos and Gatti, 2014; Kim, 2014; Tang et al., 2014), RNNs (Dong et al., 2014; Liu et al., 2015) and combined archi-

tecture (Wang et al., 2016). Few works exist on using deep neural networks for sarcasm detection, one of which is by (Ghosh and Veale, 2016) that uses a combination of RNNs and CNNs.

Eye-tracking technology is a relatively new NLP, with very few systems directly making use of gaze data in prediction frameworks. Klerke et al. (2016) present a novel multi-task learning approach for sentence compression using labeled data, while, Barrett and Søgaard (2015) discriminate between grammatical functions using gaze features. The closest works to ours are by Mishra et al. (2016b) and Mishra et al. (2016c) that introduce feature engineering based on both gaze and text data for sentiment and sarcasm detection tasks. These recent advancements motivate us to explore the cognitive NLP paradigm.

9 Conclusion and Future Directions

In this work, we proposed a multimodal ensemble of features, automatically learned using variants of CNNs from text and readers' eye-movement data, for the tasks of sentiment and sarcasm classification. On multiple published datasets for which gaze information is available, our systems could often achieve significant performance improvements over (a) systems that rely on handcrafted gaze and textual features and (b) CNN based systems that rely on text input alone. An analysis of the learned features confirms that the combination of automatically learned features is indeed capable of representing deep linguistic subtleties in text that pose challenges to sentiment and sarcasm classifiers. Our future agenda includes: (a) optimizing the CNN framework hyper-parameters (*e.g.*, filter width, dropout, embedding dimensions, *etc.*) to obtain better results, (b) exploring the applicability of our technique for document-level sentiment analysis and (c) applying our framework to related problems, such as emotion analysis, text summarization, and question-answering, where considering textual clues alone may not prove to be sufficient.

Acknowledgments

We thank Anoop Kunchukuttan, Joe Cheri Ross, and Sachin Pawar, research scholars of the Center for Indian Language Technology (CFILT), IIT Bombay for their valuable inputs.

References

Cem Akkaya, Janyce Wiebe, and Rada Mihalcea. 2009. Subjectivity word sense disambiguation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. ACL, pages 190–199.

AR Balamurali, Aditya Joshi, and Pushpak Bhattacharyya. 2011. Harnessing wordnet senses for supervised sentiment classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 1081–1091.

Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Modelling sarcasm in twitter, a novel approach. *ACL 2014* page 50.

Maria Barrett and Anders Søgaard. 2015. Using reading behavior to predict grammatical functions. In *Proceedings of the Sixth Workshop on Cognitive Aspects of Computational Language Learning*. Association for Computational Linguistics, Lisbon, Portugal, pages 1–5.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Kushal Dave, Steve Lawrence, and David M Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*. ACM, pages 519–528.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 107–116.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL (2)*. pages 49–54.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING*.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of NAACL-HLT*. pages 161–169.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 581–586.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* .

Aditya Joshi, Abhijit Mishra, Nivvedan Senthamilselvan, and Pushpak Bhattacharyya. 2014. Measuring sentiment annotation complexity of text. In *ACL (Daniel Marcu 22 June 2014 to 27 June 2014)*. ACL.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. *Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics, Beijing, China* page 757.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. *arXiv preprint arXiv:1604.03357* .

Reinhold Kliegl, Ellen Grabner, Martin Rolfs, and Ralf Engbert. 2004. Length, frequency, and predictability effects of words on eye movements in reading. *European Journal of Cognitive Psychology* 16(1-2):262–284.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. pages 1097–1105.

Marta Kutas and Steven A Hillyard. 1980. Reading senseless sentences: Brain potentials reflect semantic incongruity. *Science* 207(4427):203–205.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of NAACL-HLT*. pages 681–691.

Christine Liebrecht, Florian Kunneman, and Antal van den Bosch. 2013. The perfect solution for detecting sarcasm in tweets# not. *WASSA 2013* page 29.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*, Springer, pages 415–463.

Pengfei Liu, Shafiq R Joty, and Helen M Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*. pages 1433–1443.

Titus Malsburg, Reinhold Kliegl, and Shravan Vasishth. 2015. Determinants of scanpath regularity in reading. *Cognitive science* 39(7):1675–1703.

Justin Martineau and Tim Finin. 2009. Delta tfidf: An improved feature space for sentiment analysis. *ICWSM* 9:106.

Diana Maynard and Mark A Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *Proceedings of LREC*.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. In *NAACL HLT 2016*. pages 1030–1040.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *HLT-NAACL*. volume 13, pages 746–751.

Abhijit Mishra, Diptesh Kanojia, and Pushpak Bhattacharyya. 2016a. Predicting readers' sarcasm understandability by modeling gaze behavior. In *Proceedings of AAAI*.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2016b. Harnessing cognitive features for sarcasm detection. *ACL 2016* page 156.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2016c. Leveraging cognitive features for sentiment analysis. *CoNLL 2016* page 156.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *NAACL-HLT*. Association for Computational Linguistics, pages 786–794.

Vincent Ng, Sajib Dasgupta, and SM Arifin. 2006. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the COLING/ACL on Main conference poster sessions*. Association for Computational Linguistics, pages 611–618.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 115–124.

Soujanya Poria, Erik Cambria, Gregoire Winterstein, and Guang-Bin Huang. 2014. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. *Knowledge-Based Systems* 69:45–63.

Keith Rayner and Susan A Duffy. 1986. Lexical complexity and fixation times in reading: Effects of word frequency, verb complexity, and lexical ambiguity. *Memory & Cognition* 14(3):191–201.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of Empirical Methods in Natural Language Processing*. pages 704–714.

Raksha Sharma and Pushpak Bhattacharyya. 2013. Detecting domain dedicated polar words. In *Proceedings of the International Joint Conference on Natural Language Processing*.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 1701–1708.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. pages 208–212.

Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 225–230.

Erroll Wood and Andreas Bulling. 2014. Eyetab: Model-based gaze estimation on unmodified tablet computers. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, pages 207–210.

Michiya Yamamoto, Hironobu Nakagawa, Koichi Egawa, and Takashi Nagamatsu. 2013. Development of a mobile tablet pc with gaze-tracking function. In *Human Interface and the Management of Information. Information and Interaction for Health, Safety, Mobility and Complex Environments*, Springer, pages 421–429.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

An Unsupervised Neural Attention Model for Aspect Extraction

Ruidan He[†‡], **Wee Sun Lee**[†], **Hwee Tou Ng**[†], and **Daniel Dahlmeier**[‡]
[†]Department of Computer Science, National University of Singapore
[‡]SAP Innovation Center Singapore
[†]{ruidanhe,leews,nght}@comp.nus.edu.sg
[‡]d.dahlmeier@sap.com

Abstract

Aspect extraction is an important and challenging task in aspect-based sentiment analysis. Existing works tend to apply variants of topic models on this task. While fairly successful, these methods usually do not produce highly coherent aspects. In this paper, we present a novel neural approach with the aim of discovering coherent aspects. The model improves coherence by exploiting the distribution of word co-occurrences through the use of neural word embeddings. Unlike topic models which typically assume independently generated words, word embedding models encourage words that appear in similar contexts to be located close to each other in the embedding space. In addition, we use an attention mechanism to de-emphasize irrelevant words during training, further improving the coherence of aspects. Experimental results on real-life datasets demonstrate that our approach discovers more meaningful and coherent aspects, and substantially outperforms baseline methods on several evaluation tasks.

1 Introduction

Aspect extraction is one of the key tasks in sentiment analysis. It aims to extract entity aspects on which opinions have been expressed (Hu and Liu, 2004; Liu, 2012). For example, in the sentence *"The beef was tender and melted in my mouth"*, the aspect term is *"beef"*. Two sub-tasks are performed in aspect extraction: (1) extracting all aspect terms (e.g., *"beef"*) from a review corpus, (2) clustering aspect terms with similar meaning into categories where each category represents a single aspect (e.g., cluster *"beef"*, *"pork"*, *"pasta"*, and *"tomato"* into one aspect *food*).

Previous works for aspect extraction can be categorized into three approaches: rule-based, supervised, and unsupervised. Rule-based methods usually do not group extracted aspect terms into categories. Supervised learning requires data annotation and suffers from domain adaptation problems. Unsupervised methods are adopted to avoid reliance on labeled data needed for supervised learning.

In recent years, Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and its variants (Titov and McDonald, 2008; Brody and Elhadad, 2010; Zhao et al., 2010; Mukherjee and Liu, 2012) have become the dominant unsupervised approach for aspect extraction. LDA models the corpus as a mixture of topics (aspects), and topics as distributions over word types. While the mixture of aspects discovered by LDA-based models may describe a corpus fairly well, we find that the individual aspects inferred are of poor quality – aspects often consist of unrelated or loosely-related concepts. This may substantially reduce users' confidence in using such automated systems. There could be two primary reasons for the poor quality. Conventional LDA models do not directly encode word co-occurrence statistics which are the primary source of information to preserve topic coherence (Mimno et al., 2011). They implicitly capture such patterns by modeling word generation from the document level, assuming that each word is generated independently. Furthermore, LDA-based models need to estimate a distribution of topics for each document. Review documents tend to be short, thus making the estimation of topic distributions more difficult.

In this work, we present a novel neural approach to tackle the weaknesses of LDA-based methods. We start with neural word embeddings that al-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 388–397
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1036

ready map words that usually co-occur within the same context to nearby points in the embedding space (Mikolov et al., 2013). We then filter the word embeddings within a sentence using an attention mechanism (Bahdanau et al., 2015) and use the filtered words to construct aspect embeddings. The training process for aspect embeddings is analogous to autoencoders, where we use dimension reduction to extract the common factors among embedded sentences and reconstruct each sentence through a linear combination of aspect embeddings. The attention mechanism de-emphasizes words that are not part of any aspect, allowing the model to focus on aspect words. We call our proposed model *Attention-based Aspect Extraction* (ABAE).

In contrast to LDA-based models, our proposed method explicitly encodes word-occurrence statistics into word embeddings, uses dimension reduction to extract the most important aspects in the review corpus, and uses an attention mechanism to remove irrelevant words to further improve coherence of the aspects.

We have conducted extensive experiments on large review data sets. The results show that ABAE is effective in discovering meaningful and coherent aspects. It substantially outperforms baseline methods on multiple evaluation tasks. In addition, ABAE is intuitive and structurally simple. It can also easily scale to a large amount of training data. Therefore, it is a promising alternative to LDA-based methods proposed previously.

2 Related Work

The problem of aspect extraction has been well studied in the past decade. Initially, methods were mainly based on manually defined rules. Hu and Liu (2004) proposed to extract different product features through finding frequent nouns and noun phrases. They also extracted opinion terms by finding the synonyms and antonyms of opinion seed words through WordNet. Following this, a number of methods have been proposed based on frequent item mining and dependency information to extract product aspects (Zhuang et al., 2006; Somasundaran and Wiebe, 2009; Qiu et al., 2011). These models heavily depend on predefined rules which work well only when the aspect terms are restricted to a small group of nouns.

Supervised learning approaches generally model aspect extraction as a standard sequence labeling problem. Jin and Ho (2009) and Li *et al.* (2010) proposed to use hidden Markov models (HMM) and conditional random fields (CRF), respectively with a set of manually-extracted features. More recently, different neural models (Yin et al., 2016; Wang et al., 2016) were proposed to automatically learn features for CRF-based aspect extraction. Rule-based models are usually not refined enough to categorize the extracted aspect terms. On the other hand, supervised learning requires large amounts of labeled data for training purposes.

Unsupervised approaches, especially topic models, have been proposed subsequently to avoid reliance on labeled data. Generally, the outputs of those models are word distributions or rankings for each aspect. Aspects are naturally obtained without separately performing extraction and categorization. Most existing works (Brody and Elhadad, 2010; Zhao et al., 2010; Mukherjee and Liu, 2012; Chen et al., 2014) are based on variants and extensions of LDA (Blei et al., 2003). Recently, Wang et al. (2015) proposed a restricted Boltzmann machine (RBM)-based model to simultaneously extract aspects and relevant sentiments of a given review sentence, treating aspects and sentiments as separate hidden variables in RBM. However, the RBM-based model proposed in (Wang et al., 2015) relies on a substantial amount of prior knowledge such as part-of-speech (POS) tagging and sentiment lexicons. A biterm topic model (BTM) that generates co-occurring word pairs was proposed in (Yan et al., 2013). We experimentally compare ABAE and BTM on multiple tasks in this paper.

Attention models (Mnih et al., 2014) have recently gained popularity in training neural networks and have been applied to various natural language processing tasks, including machine translation (Bahdanau et al., 2015; Luong et al., 2015), sentence summarization (Rush et al., 2015), sentiment classification (Chen et al., 2016; Tang et al., 2016), and question answering (Hermann et al., 2015). Rather than using all available information, attention mechanism aims to focus on the most pertinent information for a task. Unlike previous works, in this paper, we apply attention to an unsupervised neural model. Our experimental results demonstrate its effectiveness under an unsupervised setting for aspect extraction.

3 Model Description

We describe the Attention-based Aspect Extraction (ABAE) model in this section. The ultimate goal is to learn a set of aspect embeddings, where each aspect can be interpreted by looking at the nearest words (representative words) in the embedding space. We begin by associating each word w in our vocabulary with a feature vector $\mathbf{e}_w \in \mathbb{R}^d$. We use word embeddings for the feature vectors as word embeddings are designed to map words that often co-occur in a context to points that are close by in the embedding space (Mikolov et al., 2013). The feature vectors associated with the words correspond to the rows of a word embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times d}$, where V is the vocabulary size. We want to learn embeddings of aspects, where aspects share the same embedding space with words. This requires an aspect embedding matrix $\mathbf{T} \in \mathbb{R}^{K \times d}$, where K, the number of aspects defined, is much smaller than V. The aspect embeddings are used to approximate the aspect words in the vocabulary, where the aspect words are filtered through an attention mechanism.

Each input sample to ABAE is a list of indexes for words in a review sentence. Given such an input, two steps are performed as shown in Figure 1. First, we filter away non-aspect words by down-weighting them using an attention mechanism, and construct a sentence embedding $\mathbf{z}_s$ from weighted word embeddings. Then, we try to reconstruct the sentence embedding as a linear combination of aspect embeddings from $\mathbf{T}$. This process of dimension reduction and reconstruction, where ABAE aims to transform sentence embeddings of the filtered sentences ($\mathbf{z}_s$) into their reconstructions ($\mathbf{r}_s$) with the least possible amount of distortion, preserves most of the information of the aspect words in the K embedded aspects. We next describe the process in detail.

3.1 Sentence Embedding with Attention Mechanism

We construct a vector representation $\mathbf{z}_s$ for each input sentence s in the first step. In general, we want the vector representation to capture the most relevant information with regards to the aspect (topic) of the sentence. We define the sentence embedding $\mathbf{z}_s$ as the weighted summation of word embeddings $\mathbf{e}_{w_i}$, $i = 1, ..., n$ corresponding to the word indexes in the sentence.

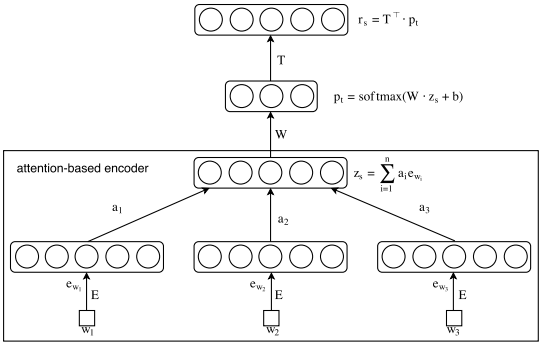


Figure 1: An example of the ABAE structure.

$$\mathbf{z}_s = \sum_{i=1}^{n} a_i \mathbf{e}_{w_i}. \tag{1}$$

For each word w_i in the sentence, we compute a positive weight a_i which can be interpreted as the probability that w_i is the right word to focus on in order to capture the main topic of the sentence. The weight a_i is computed by an attention model, which is conditioned on the embedding of the word $\mathbf{e}_{w_i}$ as well as the global context of the sentence:

$$a_i = \frac{\exp(d_i)}{\sum_{j=1}^{n} \exp(d_j)} \tag{2}$$

$$d_i = \mathbf{e}_{w_i}^{\top} \cdot \mathbf{M} \cdot \mathbf{y}_s \tag{3}$$

$$\mathbf{y}_s = \frac{1}{n} \sum_{i=1}^{n} \mathbf{e}_{w_i} \tag{4}$$

where $\mathbf{y}_s$ is simply the average of the word embeddings, which we believe captures the global context of the sentence. $\mathbf{M} \in \mathbb{R}^{d \times d}$ is a matrix mapping between the global context embedding $\mathbf{y}_s$ and the word embedding $\mathbf{e}_w$ and is learned as part of the training process. We can think of the attention mechanism as a two-step process. Given a sentence, we first construct its representation by averaging all the word representations. Then the weight of a word is assigned by considering two things. First, we filter the word through the transformation $\mathbf{M}$ which is able to capture the relevance of the word to the K aspects. Then we capture the relevance of the filtered word to the sentence by taking the inner product of the filtered word to the global context $\mathbf{y}_s$.

3.2 Sentence Reconstruction with Aspect Embeddings

We have obtained the sentence embedding. Now we describe how to compute the reconstruction of the sentence embedding. As shown in Figure 1, the reconstruction process consists of two steps of transitions, which is similar to an autoencoder. Intuitively, we can think of the reconstruction as a linear combination of aspect embeddings from $\mathbf{T}$:

$$\mathbf{r}_s = \mathbf{T}^\top \cdot \mathbf{p}_t \tag{5}$$

where $\mathbf{r}_s$ is the reconstructed vector representation, $\mathbf{p}_t$ is the weight vector over K aspect embeddings, where each weight represents the probability that the input sentence belongs to the related aspect. $\mathbf{p}_t$ can simply be obtained by reducing $\mathbf{z}_s$ from d dimensions to K dimensions and then applying a softmax non-linearity that yields normalized non-negative weights:

$$\mathbf{p}_t = softmax(\mathbf{W} \cdot \mathbf{z}_s + \mathbf{b}) \tag{6}$$

where $\mathbf{W}$, the weighted matrix parameter, and $\mathbf{b}$, the bias vector, are learned as part of the training process.

3.3 Training Objective

ABAE is trained to minimize the reconstruction error. We adopted the contrastive max-margin objective function used in previous work (Weston et al., 2011; Socher et al., 2014; Iyyer et al., 2016). For each input sentence, we randomly sample m sentences from our training data as negative samples. We represent each negative sample as $\mathbf{n}_i$ which is computed by averaging its word embeddings. Our objective is to make the reconstructed embedding $\mathbf{r}_s$ similar to the target sentence embedding $\mathbf{z}_s$ while different from those negative samples. Therefore, the unregularized objective J is formulated as a hinge loss that maximize the inner product between $\mathbf{r}_s$ and $\mathbf{z}_s$ and simultaneously minimize the inner product between $\mathbf{r}_s$ and the negative samples:

$$J(\theta) = \sum_{s \in D} \sum_{i=1}^{m} \max(0, 1 - \mathbf{r}_s \mathbf{z}_s + \mathbf{r}_s \mathbf{n}_i) \tag{7}$$

where D represents the training data set and $\theta = \{\mathbf{E}, \mathbf{T}, \mathbf{M}, \mathbf{W}, \mathbf{b}\}$ represents the model parameters.

Domain	#Reviews	#Labeled sentences
Restaurant	52,574	3,400
Beer	1,586,259	9,245

Table 1: Dataset description.

3.4 Regularization Term

We hope to learn vector representations of the most representative aspects for a review dataset. However, the aspect embedding matrix $\mathbf{T}$ may suffer from redundancy problems during training. To ensure the diversity of the resulting aspect embeddings, we add a regularization term to the objective function J to encourage the uniqueness of each aspect embedding:

$$U(\theta) = \|\mathbf{T}_n \cdot \mathbf{T}_n^\top - \mathbf{I}\| \tag{8}$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{T}_n$ is $\mathbf{T}$ with each row normalized to have length 1. Any non-diagonal element $t_{ij} (i \neq j)$ in the matrix $\mathbf{T}_n \cdot \mathbf{T}_n^\top$ corresponds to the dot product of two different aspect embeddings. U reaches its minimum value when the dot product between any two different aspect embeddings is zero. Thus the regularization term encourages orthogonality among the rows of the aspect embedding matrix $\mathbf{T}$ and penalizes redundancy between different aspect vectors. Our final objective function L is obtained by adding J and U:

$$L(\theta) = J(\theta) + \lambda U(\theta) \tag{9}$$

where λ is a hyperparameter that controls the weight of the regularization term.

4 Experimental Setup

4.1 Datasets

We evaluate our method on two real-word datasets. The detailed statistics of the datasets are summarized in Table 1.

(1) **Citysearch corpus**: This is a restaurant review corpus widely used by previous works (Ganu et al., 2009; Brody and Elhadad, 2010; Zhao et al., 2010), which contains over 50,000 restaurant reviews from Citysearch New York. Ganu et al. (2009) also provided a subset of 3,400 sentences from the corpus with manually labeled aspects. These annotated sentences are used for evaluation of aspect identification. There are six manually defined aspect labels: *Food, Staff, Ambience, Price, Anecdotes,* and *Miscellaneous*.

(2) **BeerAdvocate**: This is a beer review corpus introduced in (McAuley et al., 2012), containing over 1.5 million reviews. A subset of 1,000 reviews, corresponding to 9,245 sentences, are annotated with five aspect labels: *Feel, Look, Smell, Taste,* and *Overall*.

4.2 Baseline Methods

To validate the performance of ABAE, we compare it against a number of baselines:

(1) **LocLDA** (Brody and Elhadad, 2010): This method uses a standard implementation of LDA. In order to prevent the inference of global topics and direct the model towards rateable aspects, each sentence is treated as a separate document.

(2) **k-means**: We initialize the aspect matrix T by using the k-means centroids of the word embeddings. To show the power of ABAE, we compare its performance with using the k-means centroids directly.

(3) **SAS** (Mukherjee and Liu, 2012): This is a hybrid topic model that jointly discovers both aspects and aspect-specific opinions. This model has been shown to be competitive among topic models in discovering meaningful aspects (Mukherjee and Liu, 2012; Wang et al., 2015).

(4) **BTM** (Yan et al., 2013): This is a biterm topic model that is specially designed for short texts such as texts from social media and review sites. The major advantage of BTM over conventional LDA models is that it alleviates the problem of data sparsity in short documents by directly modeling the generation of unordered word-pair co-occurrences (biterms) over the corpus. It has been shown to perform better than conventional LDA models in discovering coherent topics.

4.3 Experimental Settings

Review corpora are preprocessed by removing punctuation symbols, stop words, and words appearing less than 10 times. For LocLDA, we use the open-source implementation GibbsLDA++[1] and for BTM, we use the implementation released by (Yan et al., 2013)[2]. We tune the hyperparameters of all topic model baselines on a held-out set with grid search using the topic coherence metric to be introduced later in Eq 10: for LocLDA, the Dirichlet priors $\alpha = 0.05$ and $\beta = 0.1$; for SAS and BTM, $\alpha = 50/K$ and $\beta = 0.1$. We run 1,000 iterations of Gibbs sampling for all topic models.

For the ABAE model, we initialize the word embedding matrix $\mathbf{E}$ with word vectors trained by word2vec with negative sampling on each dataset, setting the embedding size to 200, window size to 10, and negative sample size to 5. The parameters we use for training word embeddings are standard with no specific tuning to our data. We also initialize the aspect embedding matrix $\mathbf{T}$ with the centroids of clusters resulting from running k-means on word embeddings. Other parameters are initialized randomly. During the training process, we fix the word embedding matrix $\mathbf{E}$ and optimize other parameters using Adam (Kingma and Ba, 2014) with learning rate 0.001 for 15 epochs and batch size of 50. We set the number of negative samples per input sample m to 20, and the orthogonality penalty weight λ to 1 by tuning the hyperparameters on a held-out set with grid search. The results reported for all models are the average over 10 runs.

Following (Brody and Elhadad, 2010; Zhao et al., 2010), we set the number of aspects for the restaurant corpus to 14. We experimented with different number of aspects from 10 to 20 for the beer corpus. The results showed no major difference, so we also set it to 14. As in previous work (Brody and Elhadad, 2010; Zhao et al., 2010), we manually mapped each inferred aspect to one of the gold-standard aspects according to its top ranked representative words. In ABAE, representative words of an aspect can be found by looking at its nearest words in the embedding space using cosine as the similarity metric.

5 Evaluation and Results

We describe the evaluation tasks and report the experimental results in this section. We evaluate ABAE on two criteria:

- Is it able to find meaningful and semantically coherent aspects?
- Is it able to improve aspect identification performance on real-world review datasets?

5.1 Aspect Quality Evaluation

Table 2 presents all 14 aspects inferred by ABAE for the restaurant domain. Compared to gold-

[1] http://gibbslda.sourceforge.net
[2] http://code.google.com/p/btm/

Inferred Aspects	Representative Words	Gold Aspects
Main Dishes	beef, duck, pork, mahi, filet, veal	Food
Dessert	gelato, banana, caramel, cheesecake, pudding, vanilla	
Drink	bottle, selection, cocktail, beverage, pinot, sangria	
Ingredient	cucumber, scallion, smothered, stewed, chilli, cheddar	
General	cooking, homestyle, traditional, cuisine, authentic, freshness	
Physical Ambience	wall, lighting, ceiling, wood, lounge, floor	Ambience
Adjectives	intimate, comfy, spacious, modern, relaxing, chic	
Staff	waitstaff, server, staff, waitress, bartender, waiter	Staff
Service	unprofessional, response, condescending, aggressive, behavior, rudeness	
Price	charge, paid, bill, reservation, came, dollar	Price
Anecdotes	celebrate, anniversary, wife, fiance, recently, wedding	Anecdotes
Location	park, street, village, avenue, manhattan, brooklyn	Misc.
General	excellent, great, enjoyed, best, wonderful, fantastic	
Other	aged, reward, white, maison, mediocrity, principle	

Table 2: List of inferred aspects for restaurant reviews (left), with top representative words for each inferred aspect (middle), and the corresponding gold-standard aspect labels (right). Inferred aspect labels (left) were assigned manually.

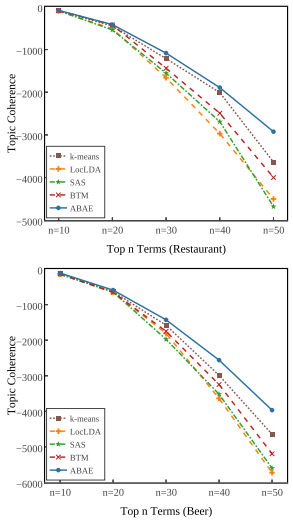


Figure 2: Average coherence score versus number of top n terms for the restaurant domain (top) and beer domain (bottom).

standard labels, the inferred aspects are more fine-grained. For example, it can distinguish main dishes from desserts, and drinks from food.

5.1.1 Coherence Score

In order to objectively measure the quality of aspects, we use *coherence score* as a metric which has been shown to correlate well with human judgment (Mimno et al., 2011). Given an aspect z and a set of top N words of z, $S^z = \{w_1^z, ..., w_N^z\}$, the coherence score is calculated as follows:

$$C(z; S^z) = \sum_{n=2}^{N} \sum_{l=1}^{n-1} log \frac{D_2(w_n^z, w_l^z) + 1}{D_1(w_l^z)} \quad (10)$$

where $D_1(w)$ is the document frequency of word w and $D_2(w_1, w_2)$ is the co-document frequency of words w_1 and w_2. A higher coherence score indicates a better aspect interpretability, i.e., more meaningful and semantically coherent.

Figure 2 shows the average *coherence score* of each model which is computed as $\frac{1}{K}\sum_{k=1}^{K} C(z_k; S^{z_k})$ on both the restaurant domain and beer domain. From the results, we make the following observations: (1) ABAE outperforms previous models for all ranked buckets. (2) BTM performs slightly better than LocLDA and SAS. This may be because BTM directly models the generation of biterms, while conventional LDA just implicitly captures such patterns by modeling word generation from the document level. (3) It is interesting to note that performing k-means on the word embeddings is sufficient to perform better than all topic model baselines, including BTM. This indicates that neural word embedding is a better model for capturing co-occurrence than LDA, even for BTM which specifically models the generation of co-occurring word pairs.

	k-means	LocLDA	SAS	BTM	ABAE
Restaurant	11	8	9	9	11
Beer	9	8	8	9	10

Table 3: Number of coherent aspects. K (number of aspects) = 14 for all models.

5.1.2 User Evaluation

As we want to discover a set of aspects that the human user finds agreeable, it is also necessary

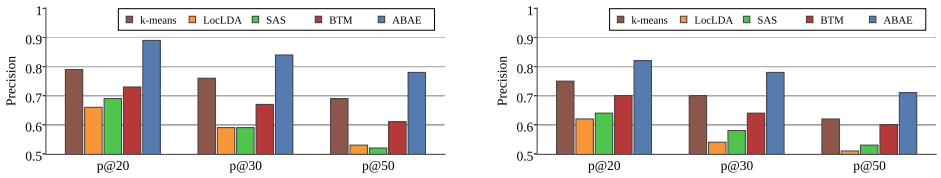


Figure 3: Average $p@n$ over all coherent aspects for the restaurant domain (left) and beer domain (right).

to carry out user evaluation directly. Following the experimental setting in (Chen et al., 2014), we recruited three human judges. Each aspect is labeled as coherent if the majority of judges assess that most of its top 50 terms coherently represent a product aspect. The numbers of coherent aspects discovered by each model are shown in Table 3. ABAE discovers the most number of coherent aspects compared with other models.

For a coherent aspect, each of its top terms is labeled as correct if and only if the majority of judges assess that it reflects the related aspect. We adopt *precision@n* (or $p@n$) to evaluate the results, which was also used in (Mukherjee and Liu, 2012; Chen et al., 2014). Figure 3 shows the average $p@n$ results over all coherent aspects for each domain. We can see that the user evaluation results correlate well with the coherence scores shown in Figure 2, where ABAE substantially outperforms all other models for all ranked buckets, especially for large values of n.

5.2 Aspect Identification

We evaluate the performance of sentence-level aspect identification on both domains using the annotated sentences shown in Table 1. The evaluation criterion is to judge how well the predictions match the true labels, measured by precision, recall, and F_1 scores. The results[4] are shown in Table 4 and Table 5.

Given a review sentence, ABAE first assigns an inferred aspect label which corresponds to the highest weight in $\mathbf{p}_t$ calculated as shown in Equation 6 . And we then assign the gold-standard label to the sentence according to the mapping between inferred aspects and gold-standard labels.

[3] k-means assigns a sentence an inferred aspect whose embedding is the closest to the averaged word embeddings of the sentence.

[4] Note that the values of P/R/F_1 reported are the average over 10 runs (except some values taken from published results in Table 4). Thus the F_1 values cannot be computed directly from corresponding P/R values

Aspect	Method	Precision	Recall	F_1
Food	LocLDA	0.898	0.648	0.753
	ME-LDA	0.874	0.787	0.828
	SAS	0.867	0.772	0.817
	BTM	0.933	0.745	0.816
	SERBM	0.891	**0.854**	**0.872**
	k-means[3]	0.931	0.647	0.755
	ABAE	**0.953**	0.741	0.828
Staff	LocLDA	0.804	0.585	0.677
	ME-LDA	0.779	0.540	0.638
	SAS	0.774	0.556	0.647
	BTM	**0.828**	0.579	0.677
	SERBM	0.819	0.582	0.680
	k-means	0.789	0.685	0.659
	ABAE	0.802	**0.728**	**0.757**
Ambience	LocLDA	0.603	0.677	0.638
	ME-LDA	0.773	0.558	0.648
	SAS	0.780	0.542	0.640
	BTM	0.813	0.599	0.685
	SERBM	0.805	0.592	0.682
	k-means	0.730	0.637	0.677
	ABAE	**0.815**	**0.698**	**0.740**

Table 4: Aspect identification results on the restaurant domain. The results of LocLDA and ME-LDA are taken from (Zhao et al., 2010); the results of SAS and SERBM are taken from (Wang et al., 2015).

For the restaurant domain, we follow the experimental settings of previous work (Brody and Elhadad, 2010; Zhao et al., 2010; Wang et al., 2015) to make our results comparable. To do that, (1) we only used the single-label sentences for evaluation to avoid ambiguity (about 83% of labeled sentences have a single label), and (2) we only evaluated on three major aspects, namely *Food*, *Staff*, and *Ambience*. The other aspects do not show clear patterns in either word usage or writing style, which makes these aspects very hard for even humans to identify. Besides the baseline models, we also compare the results with other published models, including MaxEnt-LDA (ME-LDA) (Zhao et al., 2010) and SERBM (Wang et al., 2015). SERBM has reported state-of-the-art results for aspect identification on the restaurant corpus to date. However, SERBM relies on a substantial amount of prior knowledge.

Aspect	Method	Precision	Recall	F_1
Feel	k-means	0.720	0.815	0.737
	LocLDA	**0.938**	0.537	0.675
	SAS	0.783	0.695	0.730
	BTM	0.892	0.687	0.772
	ABAE	0.815	**0.824**	**0.816**
Taste	k-means	0.533	0.413	0.456
	LocLDA	0.399	**0.655**	0.487
	SAS	0.543	0.496	0.505
	BTM	0.616	0.467	**0.527**
	ABAE	**0.637**	0.358	0.456
Smell	k-means	**0.844**	0.295	0.422
	LocLDA	0.560	0.488	0.489
	SAS	0.336	0.673	0.404
	BTM	0.541	0.549	0.527
	ABAE	0.483	**0.744**	**0.575**
Taste+Smell	k-means	0.697	0.828	0.740
	LocLDA	0.651	**0.873**	0.735
	SAS	0.804	0.759	0.769
	BTM	0.885	0.760	0.815
	ABAE	**0.897**	0.853	**0.866**
Look	k-means	0.915	0.696	0.765
	LocLDA	0.963	0.676	0.774
	SAS	0.958	0.705	0.806
	BTM	0.953	0.854	0.872
	ABAE	**0.969**	**0.882**	**0.905**
Overall	k-means	0.693	0.648	0.639
	LocLDA	0.558	0.690	0.603
	SAS	0.618	0.664	0.619
	BTM	**0.699**	0.715	0.700
	ABAE	0.654	**0.828**	**0.725**

Table 5: Aspect identification results on the beer domain.

We make the following observations from Table 4: (1) ABAE outperforms all other models on F_1 score for aspects *Staff* and *Ambience*. (2) The F_1 score of ABAE for *Food* is worse than SERBM while its precision is very high. We analyzed the errors and found that most of the sentences we failed to recognize as *Food* are general descriptions without specific food words appearing. For example, the true label for the sentence "*The food is prepared quickly and efficiently.*" is *Food*. ABAE assigns *Staff* to it as the highly focused words according to the attention mechanism are *quickly* and *efficiently* which are more related to *Staff*. In fact, although this sentence contains the word *food*, we think it is a rather general description of service. (3) ABAE substantially outperforms k-means for this task although both methods perform well for extracting coherent aspects as shown in Figure 2 and Figure 3. This shows the power brought by the attention mechanism, which is able to capture the main topic of a sentence by only focusing on aspect-related words.

For the beer domain, in addition to the five gold-standard aspect labels, we also combined *Taste* and *Smell* to form a single aspect – *Taste+Smell*. This is because these two aspects are very similar

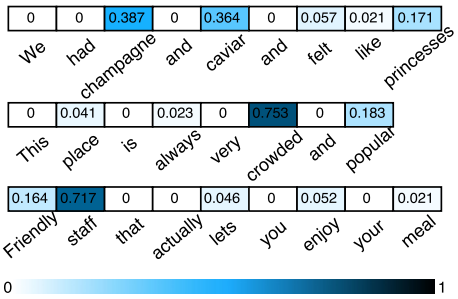


Figure 4: Visualization of the attention layer.

and many words can be used to describe both aspects. For example, the words *spicy*, *bitter*, *fresh*, *sweet*, etc. are top ranked representative words in both aspects, which makes it very hard even for humans to distinguish them. Since *Taste* and *Smell* are highly correlated and difficult to separate in real life, a natural way to evaluate is to treat them as a single aspect.

We can see from Table 5 that due to the issue described above, all models perform poorly on *Taste* and *Smell*. ABAE outperforms previous models in F_1 scores on all aspects except for *Taste*. The results demonstrate the capability of ABAE in identifying separable aspects.

Aspect	Method	Precision	Recall	F_1
Food	ABAE$^-$	0.898	0.739	0.791
	ABAE	**0.953**	**0.741**	**0.828**
Staff	ABAE$^-$	0.784	0.669	0.693
	ABAE	**0.802**	**0.728**	**0.757**
Ambience	ABAE$^-$	0.782	0.660	0.703
	ABAE	**0.815**	**0.698**	**0.740**

Table 6: Comparison between ABAE and ABAE$^-$ on aspect identification on the restaurant domain.

5.3 Validating the Effectiveness of Attention Model

Figure 4 shows the weights of words assigned by the attention model for some example sentences. As we can see, the weights learned by the model correspond very strongly with human intuition. In order to evaluate how attention model affects the overall performance of ABAE, we conduct experiments to compare ABAE and ABAE$^-$ on aspect identification, where ABAE$^-$ denotes the model in which the attention layer is switched off and sentence embedding is calculated by averaging its word embeddings: $z_s = \frac{1}{n}\sum_{i=1}^{n} e_{w_i}$. The results on the restaurant domain are shown in Table 6. ABAE achieves substantially higher precision and recall on all aspects compared with

ABAE$^-$, which demonstrates the effectiveness of the attention mechanism.

6 Conclusion

We have presented ABAE, a simple yet effective neural attention model for aspect extraction. In contrast to LDA models, ABAE explicitly captures word co-occurrence patterns and overcomes the problem of data sparsity present in review corpora. Our experimental results demonstrated that ABAE not only learns substantially higher quality aspects, but also more effectively captures the aspects of reviews than previous methods. To the best of our knowledge, we are the first to propose an unsupervised neural approach for aspect extraction. ABAE is intuitive and structurally simple, and also scales up well. All these benefits make it a promising alternative to LDA-based methods in practice.

Acknowledgements

This research is partially funded by the Economic Development Board and the National Research Foundation of Singapore.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Zhiyuan Chen, Arjun Mukherjee, and Bing Liu. 2014. Aspect extraction with automated prior knowledge learning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Gayatree Ganu, Noemie Elhadad, and Amélie Marian. 2009. Beyond the stars: Improving rating predictions using review text content. In *Proceedings of the 12th International Workshop on the Web and Databases*.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationship. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Wei Jin and Hung Hay Ho. 2009. A novel lexicalized HMM-based learning framework for web opinion mining. In *Proceedings of the 26th International Conference on Machine Learning*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 2nd International Conference on Learning Representations*.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool publishers.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Proceedings of the 12th IEEE International Conference on Data Mining*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics* 37:9–27.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics* 2.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International World Wide Web Conference*.

Linlin Wang, Kang Liu, Zhu Cao, Jun Zhao, and Gerard de Melo. 2015. Sentiment-aspect extraction based on restricted Boltzmann machines. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Wenya Wang, Sinno J. Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd International World Wide Web Conference*.

Yichun Yin, Furu Wei, Li Dong, Kaiming Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*.

Other Topics You May Also Agree or Disagree: Modeling Inter-Topic Preferences using Tweets and Matrix Factorization

Akira Sasaki, Kazuaki Hanawa, Naoaki Okazaki, and **Kentaro Inui**
Graduate School of Information Sciences
Tohoku University
{aki-s, hanawa, okazaki, inui}@ecei.tohoku.ac.jp

Abstract

We present in this paper our approach for modeling inter-topic preferences of Twitter users: for example, *those who agree with the Trans-Pacific Partnership (TPP) also agree with free trade*. This kind of knowledge is useful not only for stance detection across multiple topics but also for various real-world applications including public opinion surveys, electoral predictions, electoral campaigns, and online debates. In order to extract users' preferences on Twitter, we design linguistic patterns in which people agree and disagree about specific topics (e.g., "$\underline{A}$ is completely wrong"). By applying these linguistic patterns to a collection of tweets, we extract statements agreeing and disagreeing with various topics. Inspired by previous work on item recommendation, we formalize the task of modeling inter-topic preferences as matrix factorization: representing users' preferences as a user-topic matrix and mapping both users and topics onto a latent feature space that abstracts the preferences. Our experimental results demonstrate both that our proposed approach is useful in predicting missing preferences of users and that the latent vector representations of topics successfully encode inter-topic preferences.

1 Introduction

Social media have changed the way people shape public opinion. The latest survey by the Pew Research Center reported that a majority of US adults (62%) obtain news via social media, and of those, 18% do so often (Gottfried and Shearer, 2016). Given that news and opinions are shared and amplified by friend networks of individuals (Jamieson and Cappella, 2008), individuals are thereby isolated from information that does not fit well with their opinions (Pariser, 2011). Ironically, cutting-edge social media technologies promote ideological groups even with its potential to deliver diverse information.

A large number of studies already analyzed discussions, interactions, influences, and communities on social media along the political spectrum from liberal to conservative (Adamic and Glance, 2005; Zhou et al., 2011; Cohen and Ruths, 2013; Bakshy et al., 2015; Wong et al., 2016). Even though these studies provide intuitive visualizations and interpretations along the liberal-conservative axis, political analysts argue that the axis is flawed and insufficient for representing public opinion and ideologies (Kerlinger, 1984; Maddox and Lilie, 1984).

A potential solution for analyzing multiple axes of the political spectrum on social media is stance detection (Thomas et al., 2006; Somasundaran and Wiebe, 2009; Murakami and Raymond, 2010; Anand et al., 2011; Walker et al., 2012; Mohammad et al., 2016; Johnson and Goldwasser, 2016), whose task is to determine whether the author of a text is for, neutral, or against a topic (e.g., *free trade*, *immigration*, *abortion*). However, stance detection across different topics is extremely difficult. Anand et al. (2011) reported that a sophisticated method with topic-dependent features substantially improved the performance of stance detection within a topic, but such an approach could not outperform a baseline method with simple n-gram features when evaluated across topics. More recently, all participants of SemEval 2016 Task 6A (with five topics) could not outperform the baseline supervised method using n-gram features (Mohammad et al., 2016).

In addition, stance detection encounters dif-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 398–408
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1037

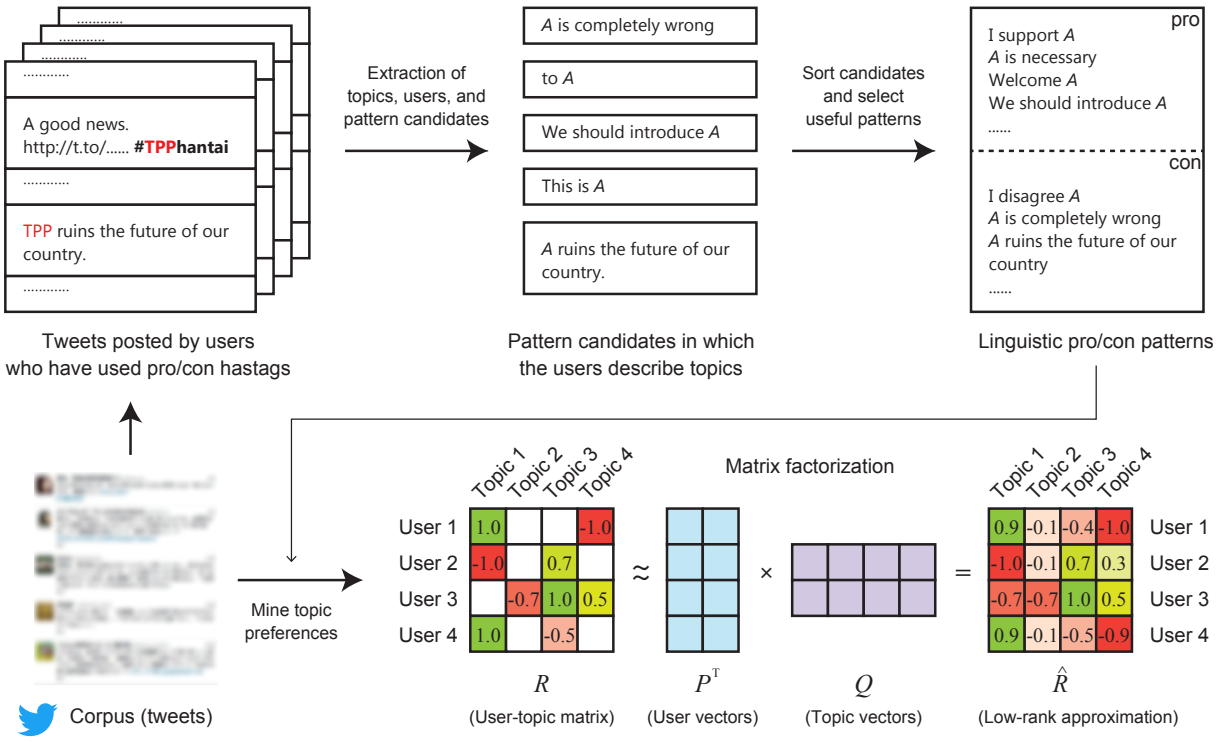


Figure 1: An overview of this study.

ficulties with different user types. Cohen and Ruths (2013) observed that existing methods on stance detection fail on "ordinary" users because such methods primarily obtain training and test data from politically vocal users (e.g., politicians); for example, they found that a stance detector trained on a dataset with politicians achieved 91% accuracy on other politicians but only achieved 54% accuracy on "ordinary" users. Establishing a bridge across different topics and users remains a major challenge not only in stance detection, but also in social media analytics.

An important component in establishing this bridge is commonsense knowledge about topics. For example, consider a topic *a revision of Article 96 of the Japanese Constitution*. We infer that the statement "we should maintain armed forces" tends to favor this topic even without any lexical overlap between the topic and the statement. This inference is reasonable because: the writer of the statement favors *armed forces*; those who favor *armed forces* also favor *a revision of Article 9*[1]; and those who favor *a revision of Article 9* also favor *a revision of Article 96*[2]. In general, this kind of commonsense knowledge can be expressed in the format: *those who agree/disagree with topic A also agree/disagree with topic B*. We call this kind of knowledge *inter-topic preference* throughout this paper.

We conjecture that previous work on stance detection indirectly learns inter-topic preferences within the same target through the use of n-gram features on a supervision data. In contrast, in the present paper, we directly acquire inter-topic preferences from an unlabeled corpus of tweets. This acquired knowledge regarding inter-topic preferences is useful not only for stance detection, but also for various real-world applications including public opinion survey, electoral campaigns, electoral predictions, and online debates.

Figure 1 provides an overview of this work. In our system, we extract linguistic patterns in which people agree and disagree about specific topics (e.g., "A is completely wrong"); to accomplish this, as described in Section 2.1, we make use of hashtags within a large collection of tweets. The patterns are then used to extract instances of users' preferences regarding various topics, as detailed in Section 2.2. Inspired by previous work on item recommendation, in Section 3, we formalize the task of modeling inter-topic preferences as a matrix factorization: representing a sparse user-topic matrix (i.e., the extracted instances) with the prod-

[1] Article 9 prohibits armed forces in Japan.

[2] Article 96 specifies high requirements for making amendments to Constitution of Japan (including Article 9).

uct of low-rank user and topic matrices. These low-rank matrices provide *latent vector representations* of both users and topics. This approach is also useful for completing preferences of "ordinary" (i.e., less vocal) users, which fills the gap between different types of users.

The contributions of this paper are threefold.

1. To the best of our knowledge, this is the first study that models inter-topic preferences for unlimited targets on real-world data.

2. Our experimental results show that this approach can accurately predict missing topic preferences of users accurately (80–94%).

3. Our experimental results also demonstrate that the latent vector representations of topics successfully encode inter-topic preferences, e.g., *those who agree with nuclear power plants also agree with nuclear fuel cycles*.

This study uses a Japanese Twitter corpus because of its availability from the authors, but the core idea is applicable to any language.

2 Mining Topic Preferences of Users

In this section, we describe how we collect statements in which users agree or disagree with various topics on Twitter, which then serves as source data for modeling inter-topic preferences. More formally, we are interested in acquiring a collection of tuples (u, t, v), where: $u \in U$ is a user; U is the set of all users on Twitter; $t \in T$ is a topic; T is the set of all topics; and $v \in \{+1, -1\}$ is $+1$ when the user u agrees with the topic t and -1 otherwise (i.e., disagreement).

Throughout this work, we use a corpus consisting of 35,328,745,115 Japanese tweets (7,340,730 users) crawled from February 6, 2013 to September 30, 2016. We removed retweets from the corpus.

2.1 Mining Linguistic Patterns of Agreement and Disagreement

We use linguistic patterns to extract tuples (u, t, v) from the aforementioned corpus. More specifically, when a tweet message matches to one of linguistic patterns of agreement (e.g., "$\underline{t}$ is necessary"), we regard that the author u of the tweet agrees with topic t. Conversely, a statement of disagreement is identified by linguistic patterns for disagreement (e.g., "$\underline{t}$ is unacceptable").

In order to design linguistic patterns, we focus on hashtags appearing in the corpus that have been popular clues for locating subjective statements such as sentiments (Davidov et al., 2010), emotions (Qadir and Riloff, 2014), and ironies (Van Hee et al., 2016). Hashtags are also useful for finding strong supporters and critics, as well as their target topics; for example, `#immigrantsWelcome` indicates that the author favors *immigrants*; and `#StopAbortion` is against *abortion*.

Based on this intuition, we design regular expressions for both *pro hashtags* "`#(.+)sansei`"[3] and *con hashtags* "`#(.+)hantai`"[4], where `(.+)` matches a target topic. These regular expressions can find users who have strong preferences to topics. Using this approach, we extracted 31,068 occurrences of pro/con hashtags used by 18,582 users for 4,899 topics. We regard the set of topics found using this procedure as set of target topics T in this study.

Each time we encounter a tweet containing a pro/con hashtag, we searched for corresponding textual statements as follows. Suppose that a tweet includes a hashtag (e.g., `#TPPsansei`) for a topic t (e.g., *TPP*). Assuming that the author of the given tweet does not change their attitude toward a topic over time, we search for other tweets posted by the same author that also have the topic keyword t. This process retrieves tweets like "I support TPP." Then, we replace the topic keyword into a variable A to extract patterns, e.g., "I support $\underline{A}$." Here, the definition of the pattern unit is language specific. For Japanese tweets, we simply recognize a pattern that starts with a variable (i.e., topic) and ends at the end of the sentence[5].

Because this procedure also extracts useless patterns such as "to A" and "this is A", we manually choose useful patterns in a systematic way: sort patterns in descending order of the number of users who use the pattern; and check the sorted list of patterns manually; and remove useless patterns.

[3] Unlike English hashtags, we systematically attach a noun *sansei*, which stands for *pro* (agreement) in Japanese, to a topic, for example, `#TPPsansei`. This paper uses the alphabetical expression `sansei` only for explanation; the actual pattern uses Chinese characters corresponding to *sansei*.

[4] A Japanese noun *hantai* stands for *con* (disagreement), for example, `#TPPhantai`. This paper uses the alphabetical expression `hantai` only for explanation; the actual pattern uses Chinese characters corresponding to *hantai*.

[5] In English, this treatment roughly corresponds to extracting a verb phrase with the variable A.

Using this approach, we obtained 100 pro patterns (e.g., "welcome A" and "A is necessary") and 100 con patterns ("do not let A" and "I don't want A").

2.2 Extracting Instances of Topic Preferences

By using the pro and con patterns acquired using the approach described in Section 2.1, we extract instances of (u, t, v) as follows. When a sentence in a tweet whose author is user u matches one of the pro patterns (e.g., "$\underline{t}$ *is necessary*") and the topic t is included in the set of target topics T, we recognize this as an instance of $(u, t, +1)$. Similarly, when a sentence matches one of the con patterns (e.g., "I don't want $\underline{t}$") and the topic t is included in the set of target topics T, we recognize this as an instance of $(u, t, -1)$. Using this approach, we collected 25,805,909 tuples corresponding to 3,302,613 users and 4,899 topics. Because these collected tuples included comparatively infrequent users and topics, we removed users and topics that appeared less than five times. In addition, there were also meaningless frequent topics such as "of" and "it". Therefore, we sorted topics in descending order of their co-occurrence frequencies with each of the pro patterns and con patterns, and then removed meaningless topics in the top 100 topics. This resulted in 9,961,509 tuples regarding 273,417 users and 2,323 topics.

3 Matrix Factorization

Using the methods described in Section 2, from the corpus, we collected a number of instances of users' preferences regarding various topics. However, Twitter users do not necessarily express preferences for all topics. In addition, it is by nature impossible to predict whether a new (i.e., nonexistent in the data) user agrees or disagrees with given topics. Therefore, in this section, we apply matrix factorization (Koren et al., 2009) in order to predict missing values, inspired by research regarding item recommendation (Bell and Koren, 2007; Dror et al., 2011). In essence, matrix factorization maps both users and topics onto a latent feature space that abstracts topic preferences of users.

Here, let R be a sparse matrix of $|U|\times|T|$. Only when a user u expresses a preference for topic t do we compute an element of the sparse matrix $r_{u,t}$,

$$r_{u,t} = \frac{\#(u,t,+1) - \#(u,t,-1)}{\#(u,t,+1) + \#(u,t,-1)} \quad (1)$$

Here, $\#(u,t,+1)$ and $\#(u,t,-1)$ represent the numbers of occurrences of instances $(u,t,+1)$ and $(u,t,-1)$, respectively. Thus, an element $r_{u,t}$ approaches $+1$ as the user u favors the topic t, and -1 otherwise. If the user u does not make any statement regarding the topic t (i.e., neither $(u,t,+1)$ nor $(u,t,-1)$ exists in the data), we do not fill the corresponding element, leaving it as a missing value.

Matrix factorization decomposes the sparse matrix R into low-dimensional matrices $P \in \mathbb{R}^{k\times|U|}$ and $Q \in \mathbb{R}^{k\times|T|}$, where k is a parameter that specifies the number of dimensions of the latent space. We minimize the following objective function to find the matrices P and Q,

$$\min_{P,Q} \sum_{(u,t)\in R} \Big((r_{u,t} - \boldsymbol{p}_u{}^\intercal \boldsymbol{q}_t)^2 + \lambda_P \|\boldsymbol{p}_u\|^2 + \lambda_Q \|\boldsymbol{q}_t\|^2 \Big). \quad (2)$$

Here, $(u,t) \in R$ is repeated for elements filled in the sparse matrix R, $\boldsymbol{p}_u \in \mathbb{R}^k$ and $\boldsymbol{q}_v \in \mathbb{R}^k$ are u column vectors of P and v column vectors of Q, respectively, and $\lambda_P \geq 0$ and $\lambda_Q \geq 0$ represent coefficients of regularization terms. We call $\boldsymbol{p}_u$ and $\boldsymbol{q}_t$ the *user vector* and *topic vector*, respectively.

Using these user and topic vectors, we can predict an element $\hat{r}_{u,t}$ that may be missing in the original matrix R,

$$\hat{r}_{u,t} \simeq \boldsymbol{p}_u{}^\intercal \boldsymbol{q}_t. \quad (3)$$

We use `libmf`[6] (Chin et al., 2015) to solve the optimization problem in Equation 2. We set regularization coefficients $\lambda_P = 0.1$ and $\lambda_Q = 0.1$ and use default values for the other parameters of `libmf`.

4 Evaluation

4.1 Determining the Dimension Parameter k

How good is the low-rank approximation found by matrix factorization? And can we find the "sweet spot" for the number of dimensions k of the latent space? We investigate the reconstruction error of matrix factorization using different values of k to answer these questions. We use Root Mean Squared Error (RMSE) to measure error,

$$RMSE = \sqrt{\frac{\sum_{(u,t)\in R} (\boldsymbol{p}_u{}^\intercal \boldsymbol{q}_t - r_{u,t})^2}{N}}. \quad (4)$$

[6] https://github.com/cjlin1/libmf

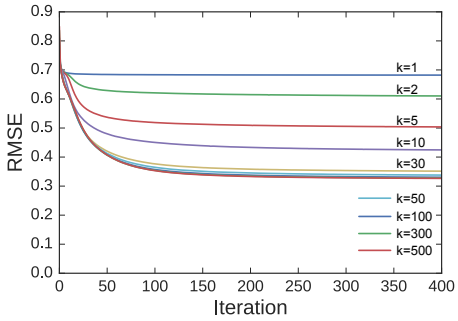


Figure 2: Reconstruction error (RMSE) of matrix factorization with different k.

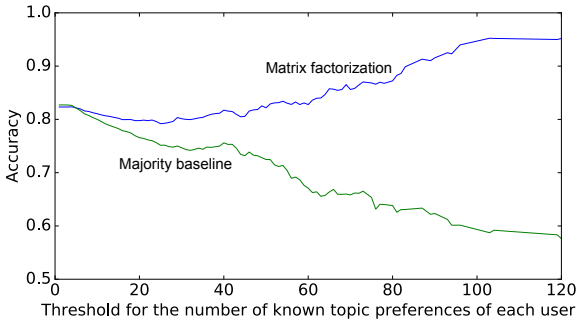


Figure 3: Prediction accuracy when changing the threshold for the number of known topic preferences of each user.

Here, N is the number of elements in the sparse matrix R (i.e., the number of known values).

Figure 2 shows RMSE values over iterations of `libmf` with the dimension parameter $k \in \{1, 2, 5, 10, 30, 50, 100, 300, 500\}$. We observed that the reconstruction error decreased as the iterative method of `libmf` progressed. The larger the number of dimensions k was, the smaller the reconstruction error became; the lowest reconstruction error was 0.3256 with $k = 500$. We also observed the error with $k = 1$, which corresponds to mapping users and topics onto one dimension similarly to the political spectrum of liberal and conservative. Judging from the relatively high RMSE values with $k = 1$, we conclude that it may be difficult to represent everything in the data using a one-dimensional axis. Based on this result, we concluded that matrix factorization with $k = 100$ is sufficient for reconstructing the original matrix R and therefore used this parameter value for the rest of our experiments.

4.2 Predicting Missing Topic Preferences

How accurately can the user and topic vectors predict missing topic preferences? To answer this question, we evaluate the accuracy in predicting hidden preferences in the matrix R as follows. First, we randomly selected 5% of existing elements in R and let Y represent the collection of the selected elements (test set). We then perform matrix factorization on the sparse matrix without the selected elements of Y, that is, only with the remaining 95% elements of R (training set). We define the accuracy of the prediction as

$$\frac{1}{|Y|} \sum_{u,t \in Y} \mathbb{1}\left(\text{sign}(\hat{r}_{u,t}) = \text{sign}(r_{u,t})\right) \quad (5)$$

Here, $r_{u,t}$ denotes the actual (i.e., self-declared) preference values, $\hat{r}_{u,t}$ represents the preference value predicted by Equation 3, $\text{sign}(.)$ represents the sign of the argument, and $\mathbb{1}(.)$ yields 1 only when the condition described in the argument holds and 0 otherwise. In other words, Equation 5 computes the proportion of correct predictions to all predictions, assuming zero to be the decision boundary between pro and con.

Figure 3 plots prediction accuracy values calculated from different sets of users. Here the x-axis represents a threshold θ, which filters out users whose declarations of topic preferences are no greater than θ topics. In other words, Figure 3 shows prediction accuracy when we know user preferences for at least θ topics. For comparison, we also include the majority baseline that predicts pro and con based on the majority of preferences regarding each topic in the training set.

Our proposed method was able to predict missing preferences with an 82.1% accuracy for users stating preferences for at least five topics. This accuracy increased as our method received more information regarding the users, reaching a 94.0% accuracy when $\theta = 100$. This result again indicates that our proposed method reasonably utilizes known preferences to complete missing preferences.

In contrast, the performance of the majority baseline decreased as it received more information regarding the users. Because this result was rather counter-intuitive, we examined the cause of this phenomenon. Consequently, this result turned out to be reasonable because preferences of vocal users deviated from those of the average users. Figure 4 illustrates this finding, showing the mean of variances of preference values $r_{u,t}$ across self-

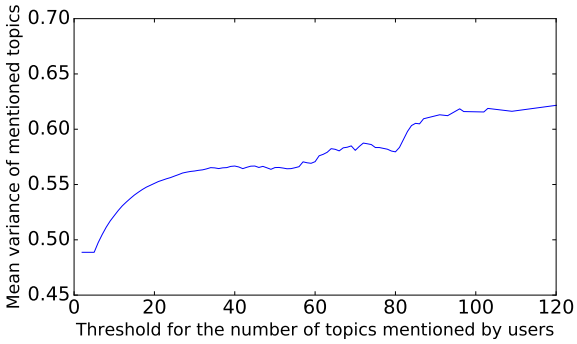


Figure 4: Mean variance of preference values of self-declared topics when changing the threshold for the number of self-declared topics.

declared topics. In the figure, the x-axis represents a threshold θ, which filters out users whose statements of topic preferences are no greater than θ topics. We observe that the mean variance increased as we focused on vocal users. Overall, these results demonstrate the usefulness of user and topic vectors in predicting missing preferences.

Table 1 shows examples in which missing preferences of two users were predicted from known statements of agreements and disagreements[7]. In the table, predicted topics are accompanied by the corresponding $\hat{r}_{u,t}$ value in parentheses. As an example, our proposed method predicted that the user A, who is positive toward *regime change* but negative toward *Okinawa US military base*, may also be positive toward *vote of non-confidence to Cabinet* but negative toward *construction of a new base*.

4.3 Inter-topic Preferences

Do the topic vectors obtained by matrix factorization capture inter-topic preferences, such as "People who agree with A also agree with B"?

Because no dataset exists for this evaluation, we created a dataset of pairwise inter-topic preferences by using a crowdsourcing service[8]. Sampling topic pairs randomly, we collected 150 topic pairs whose cosine similarities of topic vectors were below -0.6, 150 pairs whose cosine similarities were between -0.6 and 0.6, and 150 pairs whose cosine similarities were above 0.6. In this way, we obtained 450 topic pairs for evaluation.

Given a pair of topics A and B, a crowd worker was asked to choose a label from the following three options: (a) *those who agree/disagree with topic A may also agree/disagree with topic B*; (b) *those who agree/disagree with topic A may conversely disagree/agree with topic B*; (c) *otherwise (no association between A and B)*. Creating twenty pairs of topics as gold data, we removed labeling results from workers whose accuracy is less than 90%.

Consequently, we obtained 6–10 human judgements for every topic pair. Regarding (a) as $+1$ point, (b) as -1 point, and (c) as 0 point, we computed the mean of the points (i.e., average human judgements) for each topic pair. Spearman's rank correlation coefficient (ρ) between cosine similarity values of topic vectors and human judgements was 0.2210. We could observe a moderate correlation even though inter-topic preferences collected in this manner were highly subjective.

In addition to the quantitative evaluation, as summarized in Table 2, we also checked similar topics for three controversial topics, *Liberal Democratic Party (LDP)*, *constitutional amendment* and *right of foreigners to vote* (Table 2). Topics similar to LDP included synonymous ones (e.g., *Abe's LDP* and *Abe administration*) and other topics promoted by the *LDP* (e.g., *resuming nuclear power plant operations*, *bus rapid transit (BRT)* and *hate speech countermeasure law*). Considering that people who support the LDP may also tend to favor its policies, we found these results reasonable. As for the other example, *constitutional amendment* had a feature vector that was similar to that of *amendment of Article 9*, *enforcement of specific secret protection law* and *security related law*. From these results, we concluded that topic vectors were able to capture inter-topic preferences.

5 Related Work

In this section, we summarize the related work that spreads across various research fields.

Social Science and Political Science A number of of studies analyze social phenomena regarding political activities, political thoughts, and public opinions on social media. These studies

[7] We anonymized user names in these examples. In addition, we removed topics that are too discriminatory or aggressive to other countries and races. Even though the experimental results of this paper do not necessarily reflect our idea, we do not think it is a good idea to distribute politically incorrect ideas through this paper.

[8] We used Yahoo! Crowdsourcing, a Japanese online service for crowdsourcing.
http://crowdsourcing.yahoo.co.jp/

User	Type	Topic
A	Agreement (declared)	regime change, capital relocation
	Disagreement (declared)	Okinawa US military base, nuclear weapons, TPP, Abe Cabinet, Abe government, nuclear cycle, right to collective defense, nuclear power plant, Abenomics
	Agreement (predicted)	same-sex partnership ordinance (0.9697), vote of non-confidence to Cabinet (0.9248), national people's government (0.9157), abolition of tax (0.8978)
	Disagreement (predicted)	steamrollering war bill (-1.0522), worsening dispatch law (-1.0301), Sendai nuclear power plant (-1.0269), war bill (-1.0190), construction of a new base (-1.0186), Abe administration (-1.0173), landfill Henoko (-1.0158), unreasonable arrest (-1.0113)
B	Agreement (declared)	visit shrine, marriage
	Disagreement(declared)	tax increase, conscription, amend Article 9
	Agreement (predicted)	national people's government (0.8467), abolition of tax (0.8300), same-sex partnership ordinance (0.7700), security bills (0.6736)
	Disagreement (predicted)	corporate tax cuts (-1.0439), Liberal Democratic Party's draft constitution (-1.0396), radioactivity (-1.0276), rubble (-1.0159), nuclear cycle (-1.0143)

Table 1: Examples of agreement/disagreement topics predicted for two sample users A and B, with predicted score $\hat{r}_{u,v}$ shown in parenthesis.

Topic	Topics with a high degree of cosine similarity
Liberal Democratic Party (LDP)	Abe's LDP (0.3937), resuming nuclear power plant operations (0.3765), bus rapid transit (BRT) (0.3410), hate speech countermeasure law (0.3373), Henoko relocation (0.3353), C-130 (0.3338), Abe administration (0.3248), LDP & Komeito (0.2898), Prime Minister Abe (0.2835)
constitutional amendment	amendment of Article 9 (0.4520), enforcement of specific secret protection law (0.4399), security related law (0.4242), specific confidentiality protection law (0.4022), security bill amendment (0.3977), defense forces (0.3962), my number law (0.3874), collective self-defense rights (0.3687), militarist revival (0.3567)
right of foreigners to vote	human rights law (0.5405), anti-discrimination law (0.5376), hate speech countermeasure law (0.5080), foreigner's life protection (0.4553), immigration refugee (0.4520), co-organized Olympics (0.4379)

Table 2: Topics identified as being similar to the three controversial topics shown in the left column.

model the political spectrum from liberal to conservative (Adamic and Glance, 2005; Zhou et al., 2011; Cohen and Ruths, 2013; Bakshy et al., 2015; Wong et al., 2016), political parties (Tumasjan et al., 2010; Boutet et al., 2013; Makazhanov and Rafiei, 2013), and elections (O'Connor et al., 2010; Conover et al., 2011).

Employing a single axis (e.g., liberal to conservative) or a few axes (e.g., political parties and candidates of elections), these studies provide intuitive visualizations and interpretations along the respective axes. In contrast, this study is the first attempt to recognize and organize various axes of topics on social media with no prior assumptions regarding the axes. Therefore, we think our study provides a new tool for computational social science and political science that enables researchers to analyze and interpret phenomena on social media.

Next, we describe previous research focused on acquiring lexical knowledge of politics. Sim et al. (2013) measured ideological positions of candidates in US presidential elections from their speeches. The study first constructs "cue lexicons" from political writings labeled with ideologies by domain experts, using sparse additive generative models (Eisenstein et al., 2011). These constructed cue lexicons were associated with such ideologies as *left*, *center*, and *right*. Representing each speech of a candidate with cue lexicons, they inferred the proportions of ideologies of the candidate. The study requires a predefined set of labels and text data associated with the labels.

Bamman and Smith (2015) presented an unsupervised method for assessing the political stance of a proposition, such as "global warming is a hoax," along the political spectrum of liberal to conservative. In their work, a proposition was represented by a tuple in the form $\langle \text{subject}, \text{predicate} \rangle$, for example, $\langle \textit{global warming}, \textit{hoax} \rangle$. They presented a generative model for users, subjects, and predicates to find a one-dimensional latent space that corresponded to the political spectrum.

Similar to our present work, their work (Bamman and Smith, 2015) did not require labeled data

to map users and topics (i.e., subjects) onto a latent feature space. In their paper, they reported that the generative model outperformed Principal Component Analysis (PCA), which is a method for matrix factorization. Empirical results here probably reflected the underlying assumptions that PCA treats missing elements as zero and not as missing data. In contrast, in the present work, we properly distinguish missing values from zero, excluding missing elements of the original matrix from the objective function of Equation 2. Further, this work demonstrated the usefulness of the latent space, that is, topic and user vectors, in predicting missing topic preferences of users and inter-topic preferences.

Fine-grained Opinion Analysis The method presented in Section 2 is an instance of fine-grained opinion analysis (Wiebe et al., 2005; Choi et al., 2006; Johansson and Moschitti, 2010; Yang and Cardie, 2013; Deng and Wiebe, 2015), which extracts a tuple of a subjective opinion, a holder of the opinion, and a target of the opinion from text. Although these previous studies have the potential to improve the quality of the user-topic matrix R, unfortunately, no corpus or resource is available for the Japanese language. We do not currently have a large collection of English tweets, but combining fine-grained opinion analysis with matrix factorization is an immediate future work.

Causality Relation Some of inter-topic preferences in this work can be explained by causality relation, for example, "TPP promotes free trade." A number of previous studies acquire instances of causal relation (Girju, 2003; Do et al., 2011) and promote/suppress relation (Hashimoto et al., 2012; Fluck et al., 2015) from text. The causality knowledge is useful for predicting (hypotheses of) future events (Radinsky et al., 2012; Radinsky and Davidovich, 2012; Hashimoto et al., 2015).

Inter-topic preferences, however, also include pairs of topics in which causality relation hardly holds. As an example, it is unreasonable to infer that *nuclear plant* and *railroading of bills* have a causal relation, but those who dislike *nuclear plant* also oppose *railroading of bills* because presumably they think the governing political parties rush the bill for resuming a nuclear plant. In this study, we model these inter-topic preferences based on preferences of the public. That said, we have as a promising future direction of our work plans to incorporate approaches to acquire causality knowledge.

6 Conclusion

In this paper, we presented a novel approach for modeling inter-topic preferences of users on Twitter. Designing linguistic patterns for identifying support and opposition statements, we extracted users' preferences regarding various topics from a large collection of tweets. We formalized the task of modeling inter-topic preferences as a matrix factorization that maps both users and topics onto a latent feature space that abstracts users' preferences. Through our experimental results, we demonstrated that our approach was able to accurately predict missing topic preferences of users (80–94%) and that our latent vector representations of topics properly encoded inter-topic preferences.

For our immediate future work, we plan to embed the topic and user vectors to create a cross-topic stance detector. It is possible to generalize our work to model heterogeneous signals, such as interests and behaviors of people, for example, "those who are interested in A also support B," and "those who favor A also vote for B". Therefore, we believe that our work will bring about new applications in the field of NLP and other disciplines.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 15H05318 and JST CREST Grant Number J130002054, Japan.

References

Lada A. Adamic and Natalie Glance. 2005. The political blogosphere and the 2004 U.S. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery (LinkKDD 2005)*. pages 36–43. https://doi.org/10.1145/1134271.1134277.

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E. Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2011)*. pages 1–9.

Eytan Bakshy, Solomon Messing, and Lada A. Adamic. 2015. Exposure to ideologically diverse news and opinion on face-

book. *Science* 348(6239):1130–1132. https://doi.org/10.1126/science.aaa1160.

David Bamman and Noah A. Smith. 2015. Open extraction of fine-grained political statements. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*. pages 76–85. https://doi.org/10.18653/v1/D15-1008.

Robert M. Bell and Yehuda Koren. 2007. Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter* 9(2):75–79. https://doi.org/10.1145/1345448.1345465.

Antoine Boutet, Hyoungshick Kim, and Eiko Yoneki. 2013. What's in Twitter, I know what parties are popular and who you are supporting now! *Social Network Analysis and Mining (SNAM 2012)* 3(4):1379–1391. https://doi.org/10.1109/ASONAM.2012.32.

Wei-Sheng Chin, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. 2015. A fast parallel stochastic gradient method for matrix factorization in shared memory systems. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6(1):2. https://doi.org/10.1145/2668133.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*. pages 431–439. http://aclweb.org/anthology/W06-1651.

Raviv Cohen and Derek Ruths. 2013. Classifying political orientation on Twitter: It's not easy! In *Proc. of the Seventh International AAAI Conference on Weblogs and Social Media (ICWSM 2013)*. pages 91–99.

Michael D Conover, Bruno Gonçalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. 2011. Predicting the political alignment of twitter users. In *Privacy, 2011 IEEE Third International Conference on Security, Risk and Trust and 2011 IEEE Third Inernational Conference on Social Computing (PASSAT-SocialCom 2011)*. IEEE, pages 192–199.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*. pages 241–249. http://aclweb.org/anthology/C10-2028.

Lingjia Deng and Janyce Wiebe. 2015. MPQA 3.0: An entity/event-level sentiment corpus. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015)*. pages 1323–1328. https://doi.org/10.3115/v1/N15-1146.

Quang Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*. pages 294–303. http://aclweb.org/anthology/D11-1027.

Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. 2011. The Yahoo! Music dataset and KDD-Cup'11. In *Proceedings of the 2011 International Conference on KDD Cup 2011 (KDDCUP 2011)*. pages 3–18.

Jacob Eisenstein, Amr Ahmed, and Eric P Xing. 2011. Sparse additive generative models of text. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*.

Juliane Fluck, Sumit Madan, Tilia Renate Ellendorff, Theo Mevissen, Simon Clematide, Adrian van der Lek, and Fabio Rinaldi. 2015. Track 4 overview: Extraction of causal network information in biological expression language (BEL). In *Proceedings of the Fifth BioCreative Challenge Evaluation Workshop*. pages 333–346.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering - Volume 12*. pages 76–83. https://doi.org/10.3115/1119312.1119322.

Jeffrey Gottfried and Elisa Shearer. 2016. News use across social media platforms 2016. Technical report, Pew Research Center.

Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jong-Hoon Oh, and Jun'ichi Kazama. 2012. Excitatory or inhibitory: A new semantic orientation extracts contradiction and causality from the web. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*. Association for Computational Linguistics, pages 619–630. http://aclweb.org/anthology/D12-1057.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, and Jong-Hoon Oh. 2015. Generating event causality hypotheses through semantic relations. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*. pages 2396–2403.

Kathleen Hall Jamieson and Joseph N. Cappella. 2008. *Echo Chamber: Rush Limbaugh and the Conservative Media Establishment*. Oxford University Press.

Richard Johansson and Alessandro Moschitti. 2010. Syntactic and semantic structure for opinion expression detection. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL 2010)*. pages 67–76. http://aclweb.org/anthology/W10-2910.

Kristen Johnson and Dan Goldwasser. 2016. "All I know about politics is what I read in Twitter": Weakly supervised models for extracting politicians' stances from twitter. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*. pages 2966–2977. http://aclweb.org/anthology/C16-1279.

Fred N. Kerlinger. 1984. *Liberalism and Conservatism: The Nature and Structure of Social Attitudes*. Lawrence Erlbaum Associates.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37. https://doi.org/10.1109/MC.2009.263.

William S. Maddox and Stuart A. Lilie. 1984. *Beyond Liberal and Conservative: Reassessing the Political Spectrum*. Cato Inst.

Aibek Makazhanov and Davood Rafiei. 2013. Predicting political preference of twitter users. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*. pages 298–305. https://doi.org/10.1145/2492517.2492527.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. pages 31–41. https://doi.org/10.18653/v1/S16-1003.

Akiko Murakami and Rudy Raymond. 2010. Support or oppose?: classifying positions in online debates from reply activities and opinion expressions. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*. pages 869–875. http://aclweb.org/anthology/C10-2100.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media (ICWSM 2010)*. pages 122–129.

Eli Pariser. 2011. *The Filter Bubble: How the New Personalized Web Is Changing What We Read and How We Think*. Penguin Books.

Ashequl Qadir and Ellen Riloff. 2014. Learning emotion indicators from tweets: Hashtags, hashtag patterns, and phrases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. pages 1203–1209. https://doi.org/10.3115/v1/D14-1127.

Kira Radinsky and Sagie Davidovich. 2012. Learning to predict from textual data. *Journal of Artificial Intelligence Research (JAIR)* 45(1):641–684.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning causality for news events prediction. In *Proceedings of the 21st International Conference on World Wide Web (WWW 2012)*. pages 909–918. https://doi.org/10.1145/2187836.2187958.

Yanchuan Sim, Brice D. L. Acree, Justin H. Gross, and Noah A. Smith. 2013. Measuring ideological proportions in political speeches. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*. pages 91–101. http://aclweb.org/anthology/D13-1010.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009)*. pages 226–234. http://aclweb.org/anthology/P09-1026.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*. pages 327–335. http://aclweb.org/anthology/W06-1639.

Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Fourth International AAAI Conference on Weblogs and Social Media (ICWSM 2010)*. pages 178–185.

Cynthia Van Hee, Els Lefever, and Veronique Hoste. 2016. Monday mornings are my fave :) #not exploring the automatic recognition of irony in english tweets. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*. pages 2730–2739. http://aclweb.org/anthology/C16-1257.

Marilyn A. Walker, Pranav Anand, Rob Abbott, Jean E. Fox Tree, Craig Martell, and Joseph King. 2012. That is your evidence?: Classifying stance in online political debate. *Decision Support Systems* 53(4):719–729. https://doi.org/10.1016/j.dss.2012.05.032.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation* 39(2):165–210. https://doi.org/10.1007/s10579-005-7880-9.

Felix Ming Fai Wong, Chee Wei Tan, Soumya Sen, and Mung Chiang. 2016. Quantifying political leaning from tweets, retweets, and retweeters. *IEEE Transactions on Knowledge and Data Engineering* 28(8):2158–2172. https://doi.org/10.1109/TKDE.2016.2553667.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*. pages 1640–1649. http://aclweb.org/anthology/P13-1161.

Daniel Xiaodan Zhou, Paul Resnick, and Qiaozhu Mei. 2011. Classifying the political leaning of news articles and users from user votes. In *Fifth International AAAI Conference on Weblogs and Social Media (ICWSM 2011)*. pages 417–424.

Automatically Labeled Data Generation for Large Scale Event Extraction

Yubo Chen[1,2], **Shulin Liu**[1,2], **Xiang Zhang**[1], **Kang Liu**[1] and **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
{yubo.chen, shulin.liu, xiang.zhang, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Modern models of event extraction for tasks like ACE are based on supervised learning of events from small hand-labeled data. However, hand-labeled training data is expensive to produce, in low coverage of event types, and limited in size, which makes supervised methods hard to extract large scale of events for knowledge base population. To solve the data labeling problem, we propose to automatically label training data for event extraction via world knowledge and linguistic knowledge, which can detect key arguments and trigger words for each event type and employ them to label events in texts automatically. The experimental results show that the quality of our large scale automatically labeled data is competitive with elaborately human-labeled data. And our automatically labeled data can incorporate with human-labeled data, then improve the performance of models learned from these data.

1 Introduction

Event Extraction (EE), a challenging task in Information Extraction, aims at detecting and typing events (Event Detection), and extracting arguments with different roles (Argument Identification) from natural-language texts. For example, in the sentence shown in Figure 1, an EE system is expected to identify an *Attack* event triggered by *threw* and extract the corresponding five augments with different roles: *Yesterday* (Role=*Time*), *demonstrators* (Role=*Attacker*), *stones* (Role=*Instrument*), *soldiers* (Role=*Target*), and *Israeli* (Role=*Place*).

To this end, so far most methods (Nguyen et al., 2016; Chen et al., 2015; Li et al., 2014; Hong et al., 2011; Ji and Grishman, 2008) usually adopted supervised learning paradigm which relies on elaborate human-annotated data, such as ACE 2005[1], to train extractors. Although this paradigm was widely studied, existing approaches still suffer from high costs for manually labeling training data and low coverage of predefined event types. In ACE 2005, all 33 event types are manually predefined and the corresponding event information (including triggers, event types, arguments and their roles) are manually annotated only in 599 English documents since the annotation process is extremely expensive. As Figure 2 shown, nearly 60% of event types in ACE 2005 have less than 100 labeled samples and there are even three event types which have less than ten labeled samples. Moreover, those predefined 33 event types are in low coverage for Natural Language Processing (NLP) applications on large-scale data.

Figure 1: This sentence expresses an *Attack* event triggered by *threw* and containing five arguments.

Therefore, for extracting large scale events, especially in open domain scenarios, how to automatically and efficiently generate sufficient training data is an important problem. This paper aims to automatically generate training data for EE, which involves labeling triggers, event types, arguments and their roles. Figure 1 shows an example of labeled sentence. Recent improvements of Distant Supervision (DS) have been proven to be effective to label training data for Relation Extraction (RE), which aims to predict semantic re-

[1] http://projects.ldc.upenn.edu/ace/

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 409–419
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1038

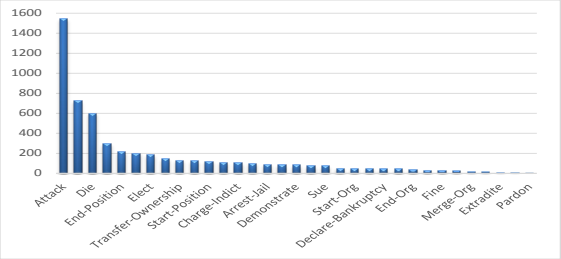


Figure 2: Statistics of ACE 2005 English Data.

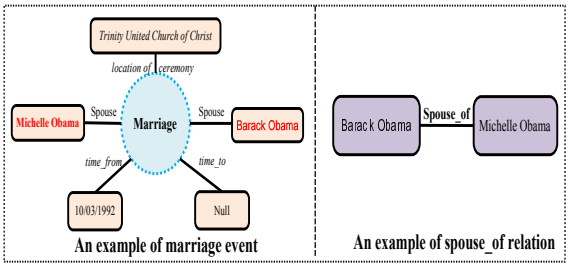


Figure 3: A comparison of events and relations.

lations between pairs of entities, formulated as $(entity_1, relation, entity_2)$. And DS for RE assumes that if two entities have a relationship in a known knowledge base, then all sentences that mention these two entities will express that relationship in some way (Mintz et al., 2009). However, when we use DS for RE to EE, we meet following challenges:

Triggers are not given out in existing knowledge bases. EE aims to detect an event instance of a specific type and extract their arguments and roles, formulated as $(event\ instance, event\ type; role_1, argument_1; role_2, argument_2; ...; role_n, argument_n)$, which can be regarded as a kind of multiple or complicated relational data. In Figure 3, the right part shows an example of $spouse\_of$ relation between $Barack\ Obama$ and $Michelle\ Obama$, where two rectangles represent two entities and the edge connecting them represents their relation. DS for RE uses two entities to automatically label training data; In comparison, the left part in Figure 3 shows a *marriage* event of $Barack\ Obama$ and $Michelle\ Obama$, where the dash circle represents the *marriage* event instance of $Barack\ Obama$ and $Michelle\ Obama$, rectangles represent arguments of the event instance, and each edge connecting an argument and the event instance expresses the role of the argument. For example, $Barack\ Obama$ plays a $Spouse$ role in this marriage event instance. It seems that we could use an event instance and an argument to automatically generate training data for argument identification just like DS for RE. However, an event instance is a virtual node in existing knowledge bases and mentioned implicitly in texts. For example, in Freebase, the aforementioned marriage event instance is represented as $m.02nqglv$ (see details in Section 2). Thus we cannot directly use an event instance and an argument, like $m.02nqglv$ and $Barack\ Obama$, to label back in sentences. In ACE event extraction program, an event instance is represented as a trigger word, which is the main word that most clearly represents an event occurrence in sentences, like *threw* in Figure 1. Following ACE, we can use trigger words to represent event instance, like *married* for *people.marriage* event instance. Unfortunately, triggers are not given out in existing knowledge bases.

To resolve the trigger missing problem mentioned above, we need to discover trigger words before employing distant supervision to automatically label event arguments. Following DS in RE, we could naturally assume that a sentence contains all arguments of an event in the knowledge base tend to express that event, and the verbs occur in these sentences tend to evoke this type of events. However, **arguments for a specific event instance are usually mentioned in multiple sentences**. Simply employing all arguments in the knowledge base to label back in sentences will generate few sentences as training samples. As shown in Table 1, only 0.02% of instances can find all argument mentions in one sentence.

Event Type	EI#	A#	S#
education.education	530,538	8	0
film.film_crew_gig	252,948	3	8
people.marriage	152,276	5	0
...	...	...	...
military.military_service	27,933	6	0
olympics.olympic_medal_honor	20,790	5	4
sum of the selected 21 events	3,870,492	100	798

Table 1: Statistics of events in Freebase. EI# denotes number of event instances in Freebase. A# denotes number of arguments for each event types, and S# indicates number of sentences contain all arguments of each event type in Wikipedia.

To solve above problems, we propose an approach to automatically generate labeled data for large scale EE by jointly using world knowledge (Freebase) and linguistic knowledge (FrameNet). At first, we put forward an approach to prioritize

arguments and select key or representative arguments (see details in Section 3.1) for each event type by using Freebase; Secondly, we merely use key arguments to label events and figure out trigger words; Thirdly, an external linguistic knowledge resource, FrameNet, is employed to filter noisy trigger words and expand more triggers; After that, we propose a Soft Distant Supervision (SDS) for EE to automatically label training data, which assumes that any sentence containing all key arguments in Freebase and a corresponding trigger word is likely to express that event in some way, and arguments occurring in that sentence are likely to play the corresponding roles in that event. Finally, we evaluate the quality of the automatically labeled training data by both manual and automatic evaluations. In addition, we employ a CNN-based EE approach with multi-instance learning for the automatically labeled data as a baseline for further research on this data. In summary, the contributions of this paper are as follows:

- To our knowledge, it is the first work to automatically label data for large scale EE via world knowledge and linguistic knowledge. All the labeled data in this paper have been released and can be downloaded freely[2].
- We propose an approach to figure out key arguments of an event by using Freebase, and use them to automatically detect events and corresponding trigger words. Moreover, we employ FrameNet to filter noisy triggers and expand more triggers.
- The experimental results show that the quality of our large scale automatically labeled data is competitive with elaborately human-annotated data. Also, our automatically labeled data can augment traditional human-annotated data, which could significantly improve the extraction performance.

2 Background

In this paper, we respectively use Freebase as our world knowledge containing event instance and FrameNet as the linguistic knowledge containing trigger information. The articles in Wikipedia are used as unstructured texts to be labeled. To understand our method easily, we first introduce them as follows:

Freebase is a semantic knowledge base (Bollacker et al., 2008), which makes use of mediators (also called compound value types, CVTs) to merge multiple values into a single value. As shown in Figure 3, *people.marriage* is one type of CVTs. There are many instances of *people.marriage* and the marriage of *Barack Obama* and *Michelle Obama* is numbered as *m.02nqglv*. *Spouse*, *from*, *to* and *location of ceremony* are roles of the *people.marriage* CVTs. *Barack Obama*, *Michelle Obama*, *10/3/1992* and *Trinity United Church of Christ* are the values of the instances. In this paper, we regard these CVTs as events, type of CVTs as event type, CVT instances as event instances, values in CVTs as arguments in events and roles of CVTs as the roles of arguments play in the event, respectively. According to the statistics of the Freebase released on 23^{th} April, 2015, there are around 1885 CVTs and around 14 million CVTs instances. After filtering out useless and meaningless CVTs, such as CVTs about user profiles and website information, we select 21 types of CVTs with around 3.8 million instances for experiments, which mainly involves events about *education*, *military*, *sports* and so on.

FrameNet[3] is a linguistic resource storing information about lexical and predicate argument semantics (Baker et al., 1998). FrameNet contains more than $1,000$ frames and $10,000$ Lexical Units (LUs). Each frame of FrameNet can be taken as a semantic frame of a type of events (Liu et al., 2016). Each frame has a set of lemmas with part of speech tags that can evoke the frame, which are called LUs. For example, *appoint.v* is a LU of *Appointing* frame in FrameNet, which can be mapped to *people.appointment* events in Freebase. And a LUs of the frame plays a similar role as the trigger of an event. Thus we use FrameNet to detect triggers in our automatically data labeling process.

Wikipedia[4] that we used was released on January, 2016. All 6.3 million articles in it are used in our experiments. We use Wikipedia because it is relatively up-to-date, and much of the information in Freebase is derived from Wikipedia.

3 Method of Generating Training Data

Figure 4 describes the architecture of automatically labeling data, which primarily involves the following four components: (i) Key argument de-

[2]https://github.com/acl2017submission/event-data

[3]http://framenet.icsi.berkeley.edu

[4]https://www.wikipedia.org/

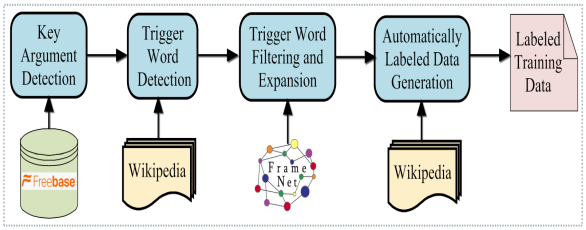


Figure 4: The architecture of automatically labeling training data for large scale event extraction.

tection, which prioritizes arguments of each event type and selects key arguments for each type of event; (ii) Trigger word detection, which uses key arguments to label sentences that may express events preliminarily, and then detect triggers; (iii) Trigger word filtering and expansion, which uses FrameNet to filter noisy triggers and expand triggers; (iv) Automatically labeled data generation, which uses a SDS to label events in sentences.

3.1 Key Argument Detection

This section illustrates how to detect key arguments for each event type via Freebase. Intuitively, arguments of a type of event play different roles. Some arguments play indispensable roles in an event, and serve as vital clues when distinguishing different events. For example, compared with arguments like *time*, *location* and so on, *spouses* are key arguments in a *marriage* event. We call these arguments as key arguments. We propose to use *Key Rate* (KR) to estimate the importance of an argument to a type of event, which is decided by two factors: *Role Saliency* and *Event Relevance*.

Role Saliency (RS) reflects the saliency of an argument to represent a specific event instance of a given event type. If we tend to use an argument to distinguish one event instance form other instances of a given event type, this argument will play a salient role in the given event type. We define RS as follows:

$$RS_{ij} = \frac{Count(A_i, ET_j)}{Count(ET_j)} \quad (1)$$

where RS_{ij} is the role saliency of i-th argument to j-th event type, $Count(A_i\ , ET_j)$ is the number of $Arguemnt_i$ occurring in all instances of $eventType_j$ in Freebase and $Count(ET_j)$ is the number of instances of $eventType_j$ in Freebase.

Event Relevance (ER) reflects the ability in which an argument can be used to discriminate different event types. If an argument occurs in every event type, the argument will have a low event relevance. We propose to compute ER as follows:

$$ER_i = \log \frac{Sum(ET)}{1 + Count(ETCi)} \quad (2)$$

where ER_i is the event relevance of i-th argument, $Sum\ (ET)$ is the number of all event types in knowledge base and $Count(ETC_i)$ is the number of event types containing i-th argument. Finally, KR is computed as follows:

$$KR_{ij} = RS_{ij} * ER_i \quad (3)$$

We compute KR for all arguments of each event type, and sort them according to KR. Then we choose top K arguments as key arguments.

3.2 Trigger Word Detection

After detecting key arguments for every event types, we use these key arguments to label sentences that may express events in Wikipedia. At first, we use Standford CoreNLP tool[5] to converts the raw Wikipedia texts into a sequence of sentences, attaches NLP annotations (POS tag, NER tag). Finally, we select sentences that contains all key arguments of an event instance in Freebase as sentences expressing corresponding events. Then we use these labeled sentences to detect triggers.

In a sentence, a verb tend to express an occurrence of an event. For example, in ACE 2005 English data, there are 60% of events triggered by verbs. As shown in Figure 1, *threw* is a trigger of *Attack* event. Intuitively, if a verb occurs more times than other verbs in the labeled sentences of one event type , the verb tends to trigger this type of event; and if a verb occurs in sentences of every event types, like *is*, the verb will have a low probability to trigger events. Thus we propose *Trigger Candidate Frequency* (TCF) and *Trigger Event Type Frequency* (TETF) to evaluate above two aspects. Finally we employ *Trigger Rate* (TR), which is the product of TCF and TETF to estimate the probability of a verb to be a trigger, which is formulated as follows:

$$TR_{ij} = TCF_{ij} * TETF_i \quad (4)$$

$$TCF_{ij} = \frac{Count(V_i, ETS_j)}{Count(ETS_j)} \quad (5)$$

[5]http://stanfordnlp.github.io/CoreNLP/

$$TETF_i = \log \frac{Sum(ET)}{1 + Count(ETI_i)} \quad (6)$$

where TR_{ij} is the trigger rate of i-th verb to j-th event type, $Count(V_i, ETS_j)$ is the number of sentences, which express j-th type of event and contain i-th verb, $Count(ETS_j)$ is the number of sentences expressing j-th event type, $Count(ETI_i)$ is the number of event types, which have the labeled sentences containing i-th verb. Finally, we choose verbs with high TR values as the trigger words for each event type.

3.3 Trigger Word Filtering and Expansion

We can obtain an initial verbal trigger lexicon by above trigger word detection. However, this initial trigger lexicon is noisy and merely contains verbal triggers. The nominal triggers like *marriage* are missing. Because the number of nouns in one sentence is usually larger than that of verbs, it is hard to use TR to find nominal triggers. Thus, we propose to use linguistic resource FrameNet to filter noisy verbal triggers and expand nominal triggers. As the success of word embedding in capturing semantics of words (Turian et al., 2010), we employ word embedding to map the events in Freebase to frames in FrameNet. Specifically, we use the average word embedding of all words in i-th Freebase event type name e_i and word embedding of k-th lexical units of j-th frame $e_{j,k}$ to compute the semantic similarity. Finally, we select the frame contains max similarity of e_i and $e_{j,k}$ as the mapped frame, which can be formulated as follows:

$$frame(i) = \arg\max_j(similarity(e_i, e_{j,k})) \quad (7)$$

Then, we filter the verb, which is in initial verbal trigger word lexicon and not in the mapping frame. And we use nouns with high confidence in the mapped frame to expand trigger lexicon.

3.4 Automatically labeled data generation

Finally, we propose a Soft Distant Supervision and use it to automatically generate training data, which assumes that any sentence containing all key arguments in Freebase and a corresponding trigger word is likely to express that event in some way, and arguments occurring in that sentence are likely to play the corresponding roles in that event.

4 Method of Event Extraction

In this paper, event extraction is formulated as a two-stage, multi-class classification task. The first stage is called *Event Classification*, which aims to predict whether the key argument candidates participate in a Freebase event. If the key arguments participate a Freebase event, the second stage is conducted, which aims to assign arguments to the event and identify their corresponding roles. We call this stage as *argument classification*. We employ two similar Dynamic Multi-pooling Convolutional Neural Networks with Multi-instance Learning (DMCNNs-MIL) for above two stages. The Dynamic Multi-pooling Convolutional Neural Networks (DMCNNs) is the best reported CNN-based model for event extraction (Chen et al., 2015) by using human-annotated training data. However, our automatically labeled data face a noise problem, which is a intrinsic problem of using DS to construct training data (Hoffmann et al., 2011; Surdeanu et al., 2012). In order to alleviate the wrong label problem, we use Multi-instance Learning (MIL) for two DMCNNs. Because the second stage is more complicated and limited in space, we take the MIL used in arguments classification as an example and describes as follows:

We define all of the parameters for the stage of argument classification to be trained in DMCNNs as θ. Suppose that there are T bags $\{M_1, M_2, ..., M_T\}$ and that the i-th bag contains q_i instances (sentences) $M_i = \{m_i^1, m_i^2, ..., m_i^{q_i}\}$, the objective of multi-instance learning is to predict the labels of the unseen bags. In stage of argument classification, we take sentences containing the same argument candidate and triggers with a same event type as a bag and all instances in a bag are considered independently. Given an input instance m_i^j, the network with the parameter θ outputs a vector O, where the r-th component O_r corresponds to the score associated with argument role r. To obtain the conditional probability $p(r|m_i^j, \theta)$, we apply a softmax operation over all argument role types:

$$p(r|m_i^j, \theta) = \frac{e^{o_r}}{\sum_{k=1}^{n} e^{o_k}} \quad (8)$$

where, n is the number of roles. And the objective of multi-instance learning is to discriminate bags rather than instances. Thus, we define the objective function on the bags. Given all (T) training bags (M_i, y_i), we can define the objective function

Event Type	Freebase Size	Sentences (KA)	Sentences (KA+T)	Examples of argument roles sorted by KR	Examples of triggers
people.marriage	152,276	56,837	26,349	spouse, spouse, from, to, location	marriage, marry, wed, wedding, couple,..., wife
music.group_membership	239,813	90,617	20,742	group, member, start, role, end	musician, singer, sing, sang, sung, concert,..., play
education.education	530,538	26,966	11,849	student, institution, degree,..., minor	educate, education, graduate, learn, study,..., student
organization.leadership	43,610	5,429	3,416	organization, person, title,..., to	CEO, charge, administer, govern, rule, boss,..., chair
olympics.olympic_medal_honor	20,790	4,056	2,605	medalist, olympics, event,..., country	win, winner, tie, victor, gold, silver,..., bronze
...	...	...	...	...	...
sum of 21 selected events	3,870,492	421,602	72,611	argument1, argument2 ,..., argumentN	trigger1, trigger2, trigger3, ... , triggerN

Table 2: The statistics of five largest automatically labeled events in selected 21 Freebase events, with their size of instances in Freebase, sentences labeled with key argument (KA) and KA + Triggers(T), examples of arguments roles sorted by KR and examples of triggers.

using cross-entropy at the bag level as follows:

$$J(\theta) = \sum_{i=1}^{T} \log p(y_i | m_i^j, \theta) \quad (9)$$

where j is constrained as follows:

$$j^* = \arg\max_j p(r | m_i^j, \theta) \quad 1 \leq j \leq q_i \quad (10)$$

To compute the network parameter θ, we maximize the log likelihood $J(\theta)$ through stochastic gradient descent over mini-batches with the Adadelta (Zeiler, 2012) update rule.

5 Experiments

In this section, we first manually evaluate our automatically labeled data. Then, we conduct automatic evaluations for our labeled data based on ACE corpus and analyze effects of different approaches to automatically label training data. Finally, we shows the performance of DMCNNs-MIL on our automatically labeled data.

5.1 Our Automatically Labeled Data

By using the proposed methods, a large set of labeled data could be generated automatically. Table 2 shows the statistics of the five largest automatically labeled events among selected 21 Freebase events. Two hyper parameters, the number of key arguments and the value of TR in our automatically data labeling, are set as 2 and 0.8, by grid search respectively. When we merely use two key arguments to label data, we will obtain $421,602$ labeled sentences. However, these sentences miss labeling triggers. Thus, we leverage these rough labeled data and FrameNet to find triggers and use SDS to generate labeled data. Finally, $72,611$ labeled sentences are generated automatically. Compared with nearly $6,000$ human annotated labeled sentence in ACE, our method can automatically generate large scale labeled training data.

5.2 Manual Evaluations of Labeled Data

##001 He is the uncle of [**Amal Clooney**], [**wife**] of the actor [**George Clooney**].
Trigger: wife **Event Type:** Marriage **MannalAnotate[Y/N]:**
Argument: Amal Clooney **Role:**Spouse **MannalAnotate[Y/N]:**
Argument: George Clooney **Role:**Spouse **MannalAnotate[Y/N]:**

##002 She was [**married**] to the cinematographer [**Theo Nischwitz**] and was sometimes credited as [**Gertrud Hinz-Nischwitz**].
Trigger: married **Event Type:** Marriage **MannalAnotate[Y/N]:**
Argument: Theo Nischwitz **Role:**Spouse **MannalAnotate[Y/N]:**
Argument: Gertrud Hinz-Nischwitz **Role:**Spouse **MannalAnotate[Y/N]:**

Figure 5: Examples of manual evaluations.

We firstly manually evaluate the precision of our automatically generated labeled data. We randomly select 500 samples from our automatically labeled data. Each selected sample is a sentence with a highlighted trigger, labeled arguments and corresponding event type and argument roles. Figure 5 gives some samples. Annotators are asked to assign one of two labels to each sample. "Y": the word highlighted in the given sentence indeed triggers an event of the corresponding type or the word indeed plays the corresponding role in that event. Otherwise "N" is labeled. It is very easy to annotate a sample for annotators, thus the annotated results are expected to be of high quality. Each sample is independently annotated by three annotators[6] (including one of the authors and two of our colleagues who are familiar with event extraction task) and the final decision is made by voting.

Stage	Average Precision
Trigger Labeling	88.9
Argument Labeling	85.4

Table 3: Manual Evaluation Results

We repeat above evaluation process on the final $72,611$ labeled data three times and the average precision is shown in Table 3. Our automatically generated data can achieve a precision of 88.9 and 85.4 for trigger labeling and argument labeling re-

[6]The inter-agreement rate is 87.5%

Methods	Trigger Identification(%)			Trigger Identification + Classification(%)			Argument Identification(%)			Argument Role(%)		
	P	R	F	P	R	F	P	R	F	P	R	F
Li's structure trained with ACE	76.9	65.0	70.4	73.7	62.3	67.5	69.8	47.9	56.8	64.7	44.4	52.7
Chen's DMCNN trained with ACE	80.4	67.7	73.5	75.6	63.6	69.1	68.8	51.9	59.1	62.2	46.9	53.5
Nguyen's JRNN trained with ACE	68.5	75.7	71.9	66.0	73.0	69.3	61.4	64.2	62.8	54.2	56.7	55.4
DMCNN trained with ED Only	77.6	67.7	72.3	72.9	63.7	68.0	64.9	51.7	57.6	58.7	46.7	52.0
DMCNN trained with ACE+ED	79.7	69.6	**74.3**	75.7	66.0	**70.5**	71.4	56.9	**63.3**	62.8	50.1	**55.7**

Table 4: Overall performance on ACE blind test data

spectively, which demonstrates that our automatically labeled data is of high quality.

5.3 Automatic Evaluations of Labeled Data

To prove the effectiveness of the proposed approach automatically, we add automatically generated labeled data into ACE dataset to expand the training sets and see whether the performance of the event extractor trained on such expanded training sets is improved. In our automatically labeled data, there are some event types that can correspond to those in ACE dataset. For example, our *people.marriage* events can be mapped to *life.marry* events in ACE2005 dataset. We mapped these types of events manually and we add them into ACE training corpus in two ways. (1) we delete the human annotated ACE data for these mapped event types in ACE dataset and add our automatically labeled data to remainder ACE training data. We call this Expanded Data (ED) as *ED Only*. (2) We directly add our automatically labeled data of mapped event types to ACE training data and we call this training data as *ACE+ED*. Then we use such data to train the same event extraction model (DMCNN) and evaluate them on the ACE testing data set. Following (Nguyen et al., 2016; Chen et al., 2015; Li et al., 2013), we used the same test set with 40 newswire articles and the same development set with 30 documents and the rest 529 documents are used for ACE training set. And we use the same evaluation metric P, R, F as ACE task defined. We select three baselines trained with ACE data. (1) *Li's structure*, which is the best reported structured-based system (Li et al., 2013). (2) *Chen's DMCNN*, which is the best reported CNN-based system (Chen et al., 2015). (3) *Nguyen's JRNN*, which is the state-of-the-arts system (Nguyen et al., 2016).

The results are shown in Table 4. Compared with all models, *DMCNN trained with ACE+ED* achieves the highest performance. This demonstrates that our automatically generated labeled data could expand human annotated training data effectively. Moreover, compared with *Chen's DMCNN trained with ACE*, *DMCNN trained with ED Only* achieves a competitive performance. This demonstrates that our large scale automatically labeled data is competitive with elaborately human-annotated data.

5.4 Discussion

Impact of Key Rate

In this section, we prove the effectiveness of KR to find key arguments and explore the impact of different numbers of key arguments to automatically generate data. We specifically select two methods as baselines for comparison with our KR method: ER and RS, which use the event relevance and role salience to sort arguments of each type of events respectively. Then we choose the same number of key arguments in all methods and use these key arguments to label data. After that we evaluate these methods by using above automatic evaluations based on ACE data. Results are shown in Table 5. *ACE+KR* achieve the best performance in both stages. This demonstrates the effectiveness of our KR methods.

Feature	**Trigger**	**Argument**
	F_1	F_1
ACE	69.1	53.5
ACE + RS	70.1	55.3
ACE + ER	69.5	54.2
ACE + KR	**70.5**	**55.7**

Table 5: Effects of ER, RS and KR

To explore the impact of different numbers of key arguments, we sort all arguments of each type of events according to KR value and select top k arguments as the key arguments. Examples are shown in Table 2. Then we automatically evaluate the performance by using automatic evaluations proposed above. Figure 6 shows the results, when we set $k = 2$, the method achieves a best

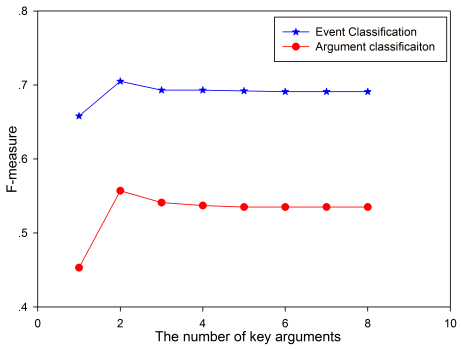


Figure 6: Effects of the number of key arguments

performance in both stages. Then, the F1 value reduces as k grows. The reason is that the heuristics for data labeling are stricter as k grows. As a result, less training data is generated. For example, if $k = 2$, we will get $25,797$ sentences labeled as *people.marriage* events and we will get 534 labeled sentences, if $k = 3$. However, when we set $k = 1$, although more labeled data are generated, the precision could not be guaranteed.

Impact of Trigger Rate and FrameNet

In this section, we prove the effectiveness of TR and FrameNet to find triggers. We specifically select two methods as baselines: TCF and TETF. TCF, TETF and TR respectively use the trigger candidate frequency, trigger event type frequency and trigger rate to sort trigger candidates of each type of events. Then we generate initial trigger lexicon by using all trigger candidates with high TCF value, TETF value or TR value. We set these hyper parameters as $0.8, 0.9$ and 0.8, respectively, which are determined by grid search from $(0.5, 0.6, 0.7, 0.8, 0.9, 1.0)$. FrameNet was used to filter noisy verbal triggers and expand nominal triggers. Trigger examples generated by *TR+Framenet* are shown in Table 2. Then we evaluate the performance of these methods by using above automatic evaluations. Results are shown in Table 6, Compared with *ACE+TCF* and *ACE+TETF*, *ACE+TR* gains a higher performance in both stages. It demonstrates the effectiveness of our TR methods. When we use FrameNet to generate triggers, compared with *ACE+TR*, we get a 1.0 improvement on trigger classification and a 1.7 improvement on argument classification. Such improvements are higher than improvements gained by other methods (TCF, IEF, TR), which demonstrates the effectiveness of the usage of FrameNet.

Feature	Trigger	Argument
	F_1	F_1
ACE	69.1	53.5
ACE + TCF	69.3	53.8
ACE + TETF	69.2	53.7
ACE + TR	69.5	54.0
ACE + TR + FrameNet	**70.5**	**55.7**

Table 6: Effects of TCF, TETF,TR and FrameNet

5.5 Performance of DMCNN-MIL

Following previous work (Mintz et al., 2009) in distant supervised RE, we evaluate our method in two ways: held-out and manual evaluation.

Held-out Evaluation

In the held-out evaluation, we hold out part of the Freebase event data during training, and compare newly discovered event instances against this held-out data. We use the following criteria to judge the correctness of each predicted event automatically: (1) An event is correct if its key arguments and event type match those of an event instance in Freebase; (2) An argument is correctly classified if its event type and argument role match those of any of the argument instance in the corresponding Freebase event. Figure 7 and Figure 8 show the precision-recall (P-R) curves for each method in the two stages of event extraction respectively. We can see that multi-instance learning is effective to alleviate the noise problem in our distant supervised event extraction.

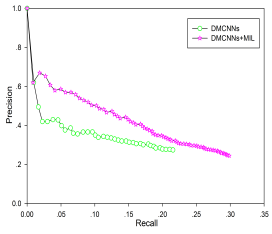


Figure 7: P-R curves for event classification.

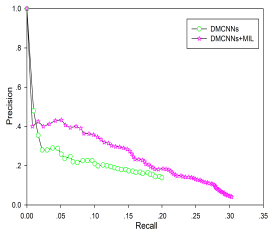


Figure 8: P-R curves for argument classification.

Human Evaluation

Because the incomplete nature of Freebase, held-out evaluation suffers from false negatives problem. We also perform a manual evaluation to eliminate these problems. In the manual evaluation, we manually check the newly discovered event instances that are not in Freebase. Because the number of these event instances in the test data is unknown, we cannot calculate the recall in this case.

Instead, we calculate the precision of the top n extracted event instances. The human evaluation results are presented in Table 7. We can see that DMCNNs-MIL achieves the best performance.

Methods	Event Classificaiton			
	Top 100	Top 300	Top 500	Average
DMCNNs	58.7	54.3	52.9	55.3
DMCNNs+MIL	**70.6**	**67.2**	**64.3**	**67.4**
Methods	Argument Classificaiton			
	Top 100	Top 300	Top 500	Average
DMCNNs	43.5	40.6	36.7	40.3
DMCNNs+MIL	**50.8**	**45.6**	**43.5**	**46.6**

Table 7: Precision for top 100, 300, and 500 events

6 Related Work

Most of previous event extraction work focused on supervised learning paradigm and trained event extractors on human-annotated data which yield relatively high performance. (Ahn, 2006; Ji and Grishman, 2008; Hong et al., 2011; McClosky et al., 2011; Li et al., 2013, 2014; Chen et al., 2015; Nguyen and Grishman, 2015; Nguyen et al., 2016). However, these supervised methods depend on the quality of the training data and labeled training data is expensive to produce. Unsupervised methods can extract large numbers of events without using labeled data (Chambers and Jurafsky, 2011; Cheung et al., 2013; Huang et al., 2016). But extracted events may not be easy to be mapped to events for a particular knowledge base.

Distant supervision have been used in relation extraction for automatically labeling training data (Mintz et al., 2009; Hinton et al., 2012; Krause et al., 2012; Krishnamurthy and Mitchell, 2012; Berant et al., 2013; Surdeanu et al., 2012; Zeng et al., 2015). But DS for RE cannot directly use for EE. For the reasons that an event is more complicated than a relation and the task of EE is more difficult than RE. The best reported supervised RE and EE system got a F1-score of 88.0% (Wang et al., 2016) and 55.4% (Nguyen et al., 2016) respectively. Reschke et al. (2014) extended the distant supervision approach to fill slots in plane crash. However, the method can only extract arguments of one plane crash type and need flight number strings as input. In other words, the approach cannot extract whole event with different types automatically.

7 Conclusion and Future Work

In this paper, we present an approach to automatically label training data for EE. The experimental results show the quality of our large scale automatically labeled data is competitive with elaborately human-annotated data. Also, we provide a DMCNN-MIL model for this data as a baseline for further research. In the future, we will use the proposed automatically data labeling method to more event types and explore more models to extract events by using automatically labeled data.

Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 61533018) and the National Basic Research Program of China (No. 2014CB340503). And this research work was also supported by Google through focused research awards program.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning About Time and Events*. pages 1–8. http://dl.acm.org/citation.cfm?id=1629235.1629236.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 86–90. http://aclweb.org/anthology/C98-1013.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1533–1544. http://aclweb.org/anthology/D13-1160.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. pages 1247–1250. http://doi.acm.org/10.1145/1376616.1376746.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 976–986. http://aclweb.org/anthology/P11-1098.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meet-*

ing of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. Association for Computational Linguistics, pages 167–176. https://doi.org/10.3115/v1/P15-1017.

Kit Jackie Chi Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 837–846. http://aclweb.org/anthology/N13-1104.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* https://arxiv.org/pdf/1207.0580.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and S. Daniel Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 541–550. http://aclweb.org/anthology/P11-1055.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1127–1136. http://aclweb.org/anthology/P11-1113.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 258–268. http://www.aclweb.org/anthology/P16-1025.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, pages 254–262. http://aclweb.org/anthology/P08-1030.

Sebastian Krause, Hong Li, Hans Uszkoreit, and Feiyu Xu. 2012. Large-scale learning of relation-extraction rules with distant supervision from the web. In *Proceedings of International Semantic Web Conference*, Springer, pages 263–278. http://link.springer.com/chapter/10.1007/978-3-642-35176-1_17.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 754–765. http://aclweb.org/anthology/D12-1069.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1846–1851. https://doi.org/10.3115/v1/D14-1198.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 73–82. http://aclweb.org/anthology/P13-1008.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 2134–2143. http://www.aclweb.org/anthology/P16-1201.

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1626–1635. http://aclweb.org/anthology/P11-1163.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, pages 1003–1011. http://aclweb.org/anthology/P09-1113.

Huu Thien Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, pages 365–371. https://doi.org/10.3115/v1/P15-2060.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 300–309. http://www.aclweb.org/anthology/N16-1034.

Kevin Reschke, Martin Jankowiak, Mihai Surdeanu, Christopher D Manning, and Daniel Jurafsky.

2014. Event extraction using distant supervision. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*. pages 4527–4531. http://www.lrec-conf.org/proceedings/lrec2014/pdf/1127_Paper.pdf.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and D. Christopher Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 455–465. http://aclweb.org/anthology/D12-1042.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 384–394. http://aclweb.org/anthology/P10-1040.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via a multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1298–1307. http://www.aclweb.org/anthology/P16-1123.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701* https://arxiv.org/pdf/1212.5701.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via a piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1753–1762. https://doi.org/10.18653/v1/D15-1203.

Time Expression Analysis and Recognition Using Syntactic Token Types and General Heuristic Rules

Xiaoshi Zhong, Aixin Sun, and Erik Cambria
School of Computer Science and Engineering
Nanyang Technological University, Singapore
{xszhong,axsun,cambria}@ntu.edu.sg

Abstract

Extracting time expressions from free text is a fundamental task for many applications. We analyze time expressions from four different datasets and find that only a small group of words are used to express time information and that the words in time expressions demonstrate similar syntactic behaviour. Based on the findings, we propose a type-based approach named SynTime[1] for time expression recognition. Specifically, we define three main syntactic token types, namely *time token*, *modifier*, and *numeral*, to group time-related token regular expressions. On the types we design general heuristic rules to recognize time expressions. In recognition, SynTime first identifies time tokens from raw text, then searches their surroundings for modifiers and numerals to form time segments, and finally merges the time segments to time expressions. As a light-weight rule-based tagger, SynTime runs in real time, and can be easily expanded by simply adding keywords for the text from different domains and different text types. Experiments on benchmark datasets and tweets data show that SynTime outperforms state-of-the-art methods.

1 Introduction

Time expression plays an important role in information retrieval and many applications in natural language processing (Alonso et al., 2011; Campos et al., 2014). Recognizing time expressions from free text has attracted considerable attention since last decade (Verhagen et al., 2007, 2010; UzZaman et al., 2013).

[1] Source: https://github.com/zhongxiaoshi/syntime

We analyze time expressions in four datasets: TimeBank (Pustejovsky et al., 2003b), Gigaword (Parker et al., 2011), WikiWars (Mazur and Dale, 2010), and Tweets. From the analysis we make four findings about time expressions. First, most time expressions are very short, with 80% of time expressions containing no more than three tokens. Second, at least 91.8% of time expressions contain at least one time token. Third, the vocabulary used to express time information is very small, with a small group of keywords. Finally, words in time expressions demonstrate similar syntactic behaviour. All the findings relate to the principle of least effort (Zipf, 1949). That is, people tend to act under the least effort in order to minimize the cost of energy at both individual level and collective level to language usage (Zipf, 1949). Time expression is part of language and acts as an interface of communication. Short expressions, occurrence, small vocabulary, and similar syntactic behaviour all reduce the cost of energy required to communicate.

According to the findings we propose a type-based approach named SynTime ('Syn' stands for syntactic) to recognize time expressions. Specifically, we define three main token types, namely *time token*, *modifier*, and *numeral*, to group time-related token regular expressions. Time tokens are the words that explicitly express time information, such as time units (e.g., 'year'). Modifiers modify time tokens; they appear before or after time tokens, e.g., 'several' and 'ago' in 'several years ago.' Numerals are ordinals and numbers. From free text SynTime first identifies time tokens, then recognizes modifiers and numerals.

Naturally, SynTime is a rule-based tagger. The key difference between SynTime and other rule-based taggers lies in the way of defining token types and the way of designing rules. The definition of token type in SynTime is inspired by part-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 420–429
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1039

of-speech in which "linguists group some words of language into classes (sets) which show similar syntactic behaviour." (Manning and Schutze, 1999) SynTime defines token types for tokens according to their syntactic behaviour. Other rule-based taggers define types for tokens based on their semantic meaning. For example, SUTime defines 5 semantic modifier types, such as frequency modifiers;[2] while SynTime defines 5 syntactic modifier types, such as modifiers that appear before time tokens. (See Section 4.1 for details.) Accordingly, other rule-based taggers design deterministic rules based on their meanings of tokens themselves. SynTime instead designs general rules on the token types rather than on the tokens themselves. For example, our general rules do not work on tokens 'February' nor '1989' but on their token types 'MONTH' and 'YEAR.' That is why we call SynTime a type-based approach. More importantly, other rule-based taggers design rules in a fixed method, including fixed length and fixed position. In contrast, SynTime designs general rules in a heuristic way, based on the idea of boundary expansion. The general heuristic rules are quite light-weight that it makes SynTime much more flexible and expansible, and leads SynTime to run in real time.

The heuristic rules are designed on token types and are independent of specific tokens, SynTime therefore is independent of specific domains, specific text types, and even specific languages that consist of specific tokens. In this paper, we test SynTime on specific domains and specific text types in English. (The test for other languages needs only to construct a collection of token regular expressions in the target language under our defined token types.) Specifically, we evaluate SynTime against three state-of-the-art methods (i.e., HeidelTime, SUTime, and UWTime) on three datasets: TimeBank, WikiWars, and Tweets.[3] TimeBank and Tweets are comprehensive datasets while WikiWars is a specific domain dataset about war; TimeBank and WikiWars are the datasets in formal text while Tweets dataset is in informal text. Experiments show that SynTime achieves comparable results on WikiWars dataset, and significantly outperforms the three state-of-the-art baselines on TimeBank and Tweets datasets. More importantly, SynTime achieves the best recalls on all three datasets and exceptionally good results on Tweets dataset. To sum up, we make the following contributions.

- We analyze time expressions from four datasets and make four findings. The findings provide evidence in terms of time expression for the principle of least effort (Zipf, 1949).
- We propose a time tagger named SynTime to recognize time expressions using syntactic token types and general heuristic rules. SynTime is independent of specific tokens, and therefore independent of specific domains, specific text types, and specific languages.
- We conduct experiments on three datasets, and the results demonstrate the effectiveness of SynTime against state-of-the-art baselines.

2 Related Work

Many research works on time expression identification are reported in TempEval exercises (Verhagen et al., 2007, 2010; UzZaman et al., 2013). The task is divided into two subtasks: recognition and normalization.

Rule-based Time Expression Recognition. Rule-based time taggers like GUTime, HeidelTime, and SUTime, predefine time-related words and rules (Verhagen et al., 2005; Strötgen and Gertz, 2010; Chang and Manning, 2012). HeidelTime (Strötgen and Gertz, 2010) hand-crafts rules with time resources like weekdays and months, and leverages language clues like part-of-speech to identify time expression. SUTime (Chang and Manning, 2012) designs deterministic rules using a cascade finite automata (Hobbs et al., 1997) on regular expressions over tokens (Chang and Manning, 2014). It first identifies individual words, then expands them to chunks, and finally to time expressions. Rule-based taggers achieve very good results in TempEval exercises.

SynTime is also a rule-based tagger while its key difference from other rule-based taggers is that between the rules and the tokens it introduces a layer of token type; its rules work on token types and are independent of specific tokens. Moreover, SynTime designs rules in a heuristic way.

Machine Learning based Method. Machine learning based methods extract features from the text and apply statistical models on the features for recognizing time expressions. Example features

[2] https://github.com/stanfordnlp/CoreNLP/tree/master/src/edu/stanford/nlp/time/rules

[3] Gigaword dataset is not used in our experiments because the labels in the dataset are not the ground truth labels but instead are automatically generated by other taggers.

include character features, word features, syntactic features, semantic features, and gazetteer features (Llorens et al., 2010; Filannino et al., 2013; Bethard, 2013). The statistical models include Markov logic network, logistic regression, support vector machines, maximum entropy, and conditional random fields (Llorens et al., 2010; UzZaman and Allen, 2010; Filannino et al., 2013; Bethard, 2013). Some models obtain good performance, and even achieve the highest F_1 of 82.71% on strict match in TempEval-3 (Bethard, 2013).

Outside TempEval exercises, Angeli et al. leverage compositional grammar and employ a EM-style approach to learn a latent parser for time expression recognition (Angeli et al., 2012). In the method named UWTime, Lee et al. handcraft a combinatory categorial grammar (CCG) (Steedman, 1996) to define a set of lexicon with rules and use L1-regularization to learn linguistic context (Lee et al., 2014). The two methods explicitly use linguistic information. In (Lee et al., 2014), especially, CCG could capture rich structure information of language, similar to the rule-based methods. Tabassum et al. focus on resolving the dates in tweets, and use distant supervision to recognize time expressions (Tabassum et al., 2016). They use five time types and assign one of them to each word, which is similar to SynTime in the way of defining types over tokens. However, they focus only on the type of date, while SynTime recoginizes all the time expressions and does not involve learning and runs in real time.

Time Expression Normalization. Methods in TempEval exercises design rules for time expression normalization (Verhagen et al., 2005; Strötgen and Gertz, 2010; Llorens et al., 2010; UzZaman and Allen, 2010; Filannino et al., 2013; Bethard, 2013). Because the rule systems have high similarity, Llorens et al. suggest to construct a large knowledge base as a public resource for the task (Llorens et al., 2012). Some researchers treat the normalization process as a learning task and use machine learning methods (Lee et al., 2014; Tabassum et al., 2016). Lee et al. (Lee et al., 2014) use AdaGrad algorithm (Duchi et al., 2011) and Tabassum et al. (Tabassum et al., 2016) use a log-linear algorithm to normalize time expressions.

SynTime focuses only on the recognition task. The normalization could be achieved by using methods similar to the existing rule systems, because they are highly similar (Llorens et al., 2012).

Table 1: Statistics of the datasets (A tweet here is a document.)

Dataset	#Docs	#Words	#TIMEX
TimeBank	183	61,418	1,243
Gigaword	2,452	666,309	12,739
WikiWars	22	119,468	2,671
Tweets	942	18,199	1,127

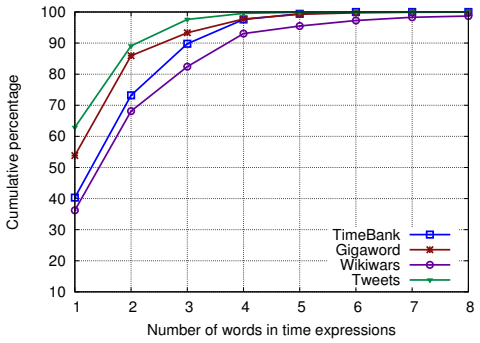


Figure 1: Length distribution of time expressions

3 Time Expression Analysis

3.1 Dataset

We conduct an analysis on four datasets: TimeBank, Gigaword, WikiWars, and Tweets. TimeBank (Pustejovsky et al., 2003b) is a benchmark dataset in TempEval series (Verhagen et al., 2007, 2010; UzZaman et al., 2013), consisting of 183 news articles. Gigaword (Parker et al., 2011) is a large *automatically* labelled dataset with 2,452 news articles and used in TempEval-3. WikiWars dataset is derived from Wikipedia articles about wars (Mazur and Dale, 2010). Tweets is our manually annotated dataset with 942 tweets of which each contains at least one time expression. Table 1 summarizes the datasets.

3.2 Finding

From the four datasets, we analyze their time expressions and make four findings. We will see that despite the four datasets vary in corpus sizes, in text types, and in domains, their time expressions demonstrate similar characteristics.

Finding 1 *Time expressions are very short. More than 80% of time expressions contain no more than three words and more than 90% contain no more than four words.*

Figure 1 plots the length distribution of time expressions. Although the texts are collected from different sources (i.e., news articles, Wikipedia articles, and tweets) and vary in sizes, the length

Table 2: The percentage of time expressions that contain at least one time token, and the average length of time expressions

Dataset	Percent	Average Length
TimeBank	94.61	2.00
Gigaword	96.44	1.70
WikiWars	91.81	2.38
Tweets	96.01	1.51

Table 3: Number of distinct words and number of distinct time tokens in time expressions

Dataset	#Words	#Time Tokens
TimeBank	130	64
Gigaword	214	80
WikiWars	224	74
Tweets	107	64

of time expressions follow a similar distribution. In particular, the one-word time expressions range from 36.23% in WikiWars to 62.91% in Tweets. In informal communication people tend to use words in minimum length to express time information. The third column in Table 2 reports the average length of time expressions. On average, time expressions contain about two words.

Finding 2 *More than 91% of time expressions contain at least one time token.*

The second column in Table 2 reports the percentage of time expressions that contain at least one time token. We find that at least 91.81% of time expressions contain time token(s). (Some time expressions have no time token but depend on other time expressions; in '2 to 8 days,' for example, '2' depends on '8 days.') This suggests that time tokens account for time expressions. Therefore, to recognize time expressions, it is essential to recognize their time tokens.

Finding 3 *Only a small group of time-related keywords are used to express time information.*

From the time expressions in all four datasets, we find that the group of keywords used to express time information is small.

Table 3 reports the number of distinct words and of distinct time tokens. The words/tokens are manually normalized before counting and their variants are ignored. For example, 'year' and '5yrs' are counted as one token 'year.' Numerals in the counting are ignored. Despite the four datasets vary in sizes, domains, and text types, the numbers of their distinct time tokens are comparable.

Across the four datasets, the number of distinct words is 350, about half of the simply summing of 675; the number of distinct time tokens is 123, less than half of the simply summing 282. Among the 123 distinct time tokens, 45 appear in all the four datasets, and 101 appear in at least two datasets. This indicates that time tokens, which account for time expressions, are highly overlapped across the four datasets. In other words, time expressions highly overlap at their time tokens.

Finding 4 *POS information could not distinguish time expressions from common words, but within time expressions, POS tags can help distinguish their constituents.*

For each dataset we list the top 10 POS tags that appear in time expressions, and their percentages over the whole text. Among the 40 tags (10 $\times$ 4 datasets), 37 have percentage lower than 20%; other 3 are CD. This indicates that POS could not provide enough information to distinguish time expressions from common words. However, the most common POS tags in time expressions are NN*, JJ, RB, CD, and DT. Within time expressions, the time tokens usually have NN* and RB, the modifiers have JJ and RB, and the numerals have CD. This finding indicates that for the time expressions, their similar constituents behave in similar syntactic way. When seeing this, we realize that this is exactly how linguists define part-of-speech for language.[4] The definition of POS for language inspires us to define a syntactic type system for the time expression, part of language.

The four findings all relate to the principle of least effort (Zipf, 1949). That is, people tend to act with least effort so as to minimize the cost of energy at both individual and collective levels to the language usage (Zipf, 1949). Time expression is part of language and acts as an interface of communication. Short expressions, occurrence, small vocabulary, and similar syntactic behaviour all reduce the cost of energy required to communicate.

To summarize: on average, a time expression contains two tokens of which one is time token and the other is modifier/numeral, and the size of time tokens is small. To recognize a time expression, therefore, we first recognize the time token, then recognize the modifier/numeral.

[4] "linguists group some words of language into classes (sets) which show similar syntactic behaviour." (Manning and Schutze, 1999)

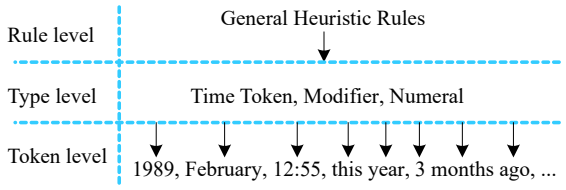


Figure 2: Layout of SynTime. The layout consists of three levels: token level, type level, and rule level. Token types group the constituent tokens of time expressions. Heuristic rules work on token types, and are independent of specific tokens.

4 SynTime: Syntactic Token Types and General Heuristic Rules

SynTime defines a syntactic type system for the tokens of time expressions, and designs heuristic rules working on the token types. Figure 2 shows the layout of SynTime, consisting of three levels: Token level, type level, and rule level. Token types at the type level group the tokens of time expressions. Heuristic rules lie at the rule level, working on token types rather than on tokens themselves. That is why the heuristic rules are general. For example, the heuristic rules do not work on tokens '1989' nor 'February,' but on their token types 'YEAR' and 'MONTH.' The heuristic rules are only relevant to token types, and are independent of specific tokens. For this reason, our token types and heuristic rules are independent of specific domains, specific text types, and even specific languages that consist of specific tokens. In this paper, we test SynTime on specific domain (i.e., war domain) and specific text types (i.e., formal text and informal text) in English. The test for other languages simply needs to construct a set of token regular expressions in the target language under our defined token types.

Figure 3 shows the overview of SynTime in practice. Shown on the left-hand side, SynTime is initialized with regular expressions over tokens. After initialization, SynTime can be directly applied on text. On the other hand, SynTime can be easily expanded by simply adding the time-related token regular expressions from training text under each defined token type. The expansion enables SynTime to recognize time expressions in text from different domains and different text types.

Shown on the right-hand side of Figure 3, SynTime recognizes time expression through three main steps. In the first step, SynTime identifies

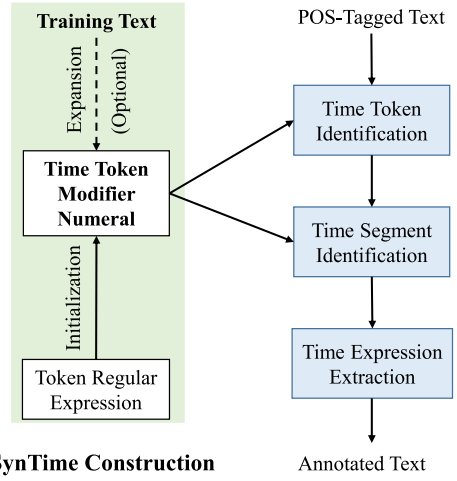


Figure 3: Overview of SynTime. Left-hand side shows the construction of SynTime, with initialization using token regular expressions, and optional expansion using training text. Right-hand side shows the main steps of SynTime recognizing time expressions.

time tokens from the POS-tagged raw text. Then around the time tokens SynTime searches for modifiers and numerals to form time segments. In the last step, SynTime transforms the time segments to time expressions.

4.1 SynTime Construction

We define a syntactic type system for time expression, specifically, 15 token types for time tokens, 5 token types for modifiers, and 1 token type for numeral. Token types to tokens is like POS tags to words; for example, 'February' has a POS tag of NNP and a token type of MONTH.

Time Token. We define 15 token types for the time tokens and use their names similar to Joda-Time classes:[5] DECADE (-), YEAR (-), SEASON (5), MONTH (12), WEEK (7), DATE (-), TIME (-), DAY_TIME (27), TIMELINE (12), HOLIDAY (20), PERIOD (9), DURATION (-), TIME_UNIT (15), TIME_ZONE (6), and ERA (2). Number in '()' indicates the number of distinct tokens in this token type. '-' indicates that this token type involves changing digits and cannot be counted.

Modifier. We define 3 token types for the modifiers according to their possible positions relative to time tokens. Modifiers that appear before time tokens are PREFIX (48); modifiers after time tokens are SUFFIX (2). LINKAGE (4) link two time

[5] http://www.joda.org/joda-time/

tokens. Besides, we define 2 special modifier types, COMMA (1) for comma ',' and IN_ARTICLE (2) for indefinite articles 'a' and 'an.'

TimeML (Pustejovsky et al., 2003a) and TimeBank (Pustejovsky et al., 2003b) do not treat most prepositions like 'on' as a part of time expressions. Thus SynTime does not collect those prepositions.

Numeral. Number in time expressions can be a time token e.g., '10' in 'October 10, 2016,' or a modifier e.g., '10' in '10 days.' We define NUMERAL (-) for the ordinals and numbers.

SynTime Initialization. The token regular expressions for initializing SynTime are collected from SUTime,[6] a state-of-the-art rule-based tagger that achieved the highest recall in TempEval-3 (Chang and Manning, 2012, 2013). Specifically, we collect from SUTime only the tokens and the regular expressions over tokens, and discard its other rules of recognizing full time expressions.

4.2 Time Expression Recognition

On the token types, SynTime designs a small set of heuristic rules to recognize time expressions. The recognition process includes three main steps: (1) time token identification, (2) time segment identification, and (3) time expression extraction.

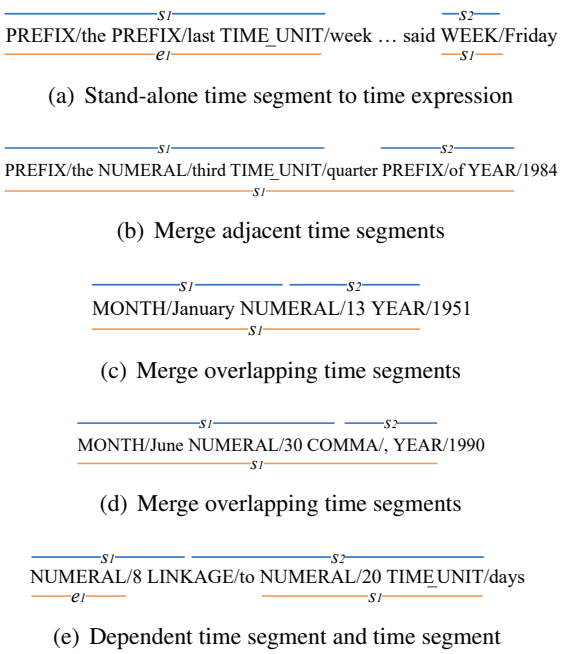
4.2.1 Time Token Identification

Identifying time tokens is simple, through matching of string and regular expressions. Some words might cause ambiguity. For example, 'May' could be a modal verb, or the fifth month of year. To filter out the ambiguous words, we use POS information. In implementation, we use Stanford POS Tagger;[7] and the POS tags for matching the instances of token types in SynTime are based on our Finding 4 in Section 3.2.

Besides time tokens are identified, in this step, individual token is assigned with one token type of either modifier or numeral if it is matched with token regular expressions. In the next two steps, SynTime works on those token types.

4.2.2 Time Segment Identification

The task of time segment identification is to search the surrounding of each time token identified in previous step for modifiers and numerals, then gather the time token with its modifiers and numerals to form a time segment. The searching is



(a) Stand-alone time segment to time expression

(b) Merge adjacent time segments

(c) Merge overlapping time segments

(d) Merge overlapping time segments

(e) Dependent time segment and time segment

Figure 4: Example time segments and time expressions. The above labels are from time segment identification; the below labels are for time expression extraction.

under simple heuristic rules in which the key idea is to expand the time token's boundaries.

At first, each time token is a time segment. If it is either a PERIOD or DURATION, then no need to further search. Otherwise, search its left and its right for modifiers and numerals. For the left searching, if encounter a PREFIX or NUMERAL or IN_ARTICLE, then continue searching. For the right searching, if encounter a SUFFIX or NUMERAL, then continue searching. Both the left and the right searching stop when reaching a COMMA or LINKAGE or a non-modifier/numeral word. The left searching does not exceed the previous time token; the right searching does not exceed the next time token. A time segment consists of exactly one time token, and zero or some modifiers/numerals.

A special kind of time segments do not contain any time token; they depend on other time segments next to them. For example, in '8 to 20 days,' 'to 20 days' is a time segment, and '8 to' forms a dependent time segment. (See Figure 4(e).)

4.2.3 Time Expression Extraction

The task of time expression extraction is to extract time expressions from the identified time segments in which the core step is to determine whether to merge two adjacent or overlapping time segments into a new time segment.

[6] https://github.com/stanfordnlp/CoreNLP/tree/master/src/edu/stanford/nlp/time/rules

[7] http://nlp.stanford.edu/software/tagger.shtml

We scan the time segments in a sentence from beginning to the end. A stand-alone time segment is a time expression. (See Figure 4(a).) The focus is to deal with two or more time segments that are adjacent or overlapping. If two time segments s_1 and s_2 are adjacent, merge them to form a new time segment s_1. (See Figure 4(b).) Consider that s_1 and s_2 overlap at a shared boundary. According to our time segment identification, the shared boundary could be a modifier or a numeral. If the word at the shared boundary is neither a COMMA nor a LINKAGE, then merge s_1 and s_2. (See Figure 4(c).) If the word is a LINKAGE, then extract s_1 as a time expression and continue scanning. When the shared boundary is a COMMA, merge s_1 and s_2 only if the COMMA's previous token and its next token satisfy the three conditions: (1) the previous token is a time token or a NUMERAL; (2) the next token is a time token; and (3) the token types of the previous token and of the next token are not the same. (See Figure 4(d).)

Although Figure 4 shows the examples as token types together with the tokens, we should note that the heuristic rules only work on the token types. After the extraction step, time expressions are exported as a sequence of tokens from the sequence of token types.

4.3 SynTime Expansion

SynTime could be expanded by simply adding new words under each defined token type without changing any rule. The expansion requires the words to be added to be annotated manually. We apply the initial SynTime on the time expressions from training text and list the words that are not covered. Whether the uncovered words are added to SynTime is manually determined. The rule for determination is that the added words can not cause ambiguity and should be generic. WikiWars dataset contains a few examples like this: 'The time Arnold reached Quebec City.' Words in this example are extremely descriptive, and we do not collect them. In tweets, on the other hand, people may use abbreviations and informal variants; for example, '2day' and 'tday' are popular spellings of 'today.' Such kind of abbreviations and informal variants will be collected.

According to our findings, not many words are used to express time information, the manual addition of keywords thus will not cost much. In addition, we find that even in tweets people tend to use formal words. In the Twitter word clusters trained from 56 million English tweets,[8] the most often used words are the formal words, and their frequencies are much greater than the informal words'. The cluster of 'today,'[9] for example, its most often use is the formal one, 'today,' which appears 1,220,829 times; while its second most often use '2day' appears only 34,827 times. The low rate of informal words (e.g., about 3% in 'today' cluster) suggests that even in informal environment the manual keyword addition costs little.

5 Experiments

We evaluate SynTime against three state-of-the-art baselines (i.e., HeidelTime, SUTime, and UWTime) on three datasets (i.e., TimeBank, WikiWars, and Tweets). WikiWars is a specific domain dataset about war; TimeBank and WikiWars are the datasets in formal text while Tweets dataset is in informal text. For SynTime we report the results of its two versions: SynTime-I and SynTime-E. SynTime-I is the initial version, and SynTime-E is the expanded version of SynTime-I.

5.1 Experiment Setting

Datasets. We use three datasets of which TimeBank and WikiWars are benchmark datasets whose details are shown in Section 3.1; Tweets is our manually labeled dataset that are collected from Twitter. For Tweets dataset, we randomly sample 4000 tweets and use SUTime to tag them. 942 tweets of which each contains at least one time expression. From the remaining 3,058 tweets, we randomly sample 500 and manually annotate them, and find that only 15 tweets contain time expressions. We therefore roughly consider that SUTime misses about 3% time expressions in tweets. Two annotators then manually annotate the 942 tweets with discussion to final agreement according to the standards of TimeML and TimeBank. We finally get 1,127 manually labeled time expressions. For the 942 tweets, we randomly sample 200 tweets as test set, and the rest 742 as training set, because a baseline UWTime requires training.

Baseline Methods. We compare SynTime with methods: HeidelTime (Strötgen and Gertz, 2010), SUTime (Chang and Manning, 2012), and UW-

[8] http://www.cs.cmu.edu/~ark/TweetNLP/cluster_viewer.html

[9] http://www.cs.cmu.edu/~ark/TweetNLP/paths/01111110010.html

Table 4: Overall performance. The best results are in bold face and the second best are underlined. Some results are borrowed from their original papers and the papers are indicated by the references.

Dataset	Method	Strict Match			Relaxed Match		
		Pr.	*Re.*	F_1	*Pr.*	*Re.*	F_1
TimeBank	HeidelTime(Strotgen et al., 2013)	83.85	78.99	81.34	93.08	87.68	90.30
	SUTime(Chang and Manning, 2013)	78.72	80.43	79.57	89.36	91.30	90.32
	UWTime(Lee et al., 2014)	86.10	80.40	83.10	**94.60**	88.40	91.40
	SynTime-I	91.43	92.75	92.09	94.29	**95.65**	**94.96**
	SynTime-E	**91.49**	**93.48**	**92.47**	93.62	**95.65**	94.62
WikiWars	HeidelTime(Lee et al., 2014)	85.20	79.30	82.10	92.60	86.20	89.30
	SUTime	78.61	76.69	76.64	95.74	89.57	92.55
	UWTime(Lee et al., 2014)	**87.70**	78.80	**83.00**	**97.60**	87.60	92.30
	SynTime-I	80.00	80.22	80.11	92.16	92.41	92.29
	SynTime-E	79.18	**83.47**	81.27	90.49	**95.39**	**92.88**
Tweets	HeidelTime	**89.58**	72.88	80.37	95.83	77.97	85.98
	SUTime	76.03	77.97	76.99	88.43	90.68	89.54
	UWTime	88.54	72.03	79.44	**96.88**	78.81	86.92
	SynTime-I	89.52	94.07	91.74	93.55	98.31	95.87
	SynTime-E	89.20	**94.49**	**91.77**	93.20	**98.78**	**95.88**

Time (Lee et al., 2014). HeidelTime and SUTime both are rule-based methods, and UWTime is a learning method. When training UWTime on Tweets, we try two settings: (1) train with only Tweets training set; (2) train with TimeBank and Tweets training set. The second setting achieves slightly better result and we report that result.

Evaluation Metrics. We follow TempEval-3 and use their evaluation toolkit[10] to report $Precision$, $Recall$, and F_1 in terms of *strict match* and *relaxed match* (UzZaman et al., 2013).

5.2 Experiment Result

Table 4 reports the overall performance. Among the 18 measures, SynTime-I and SynTime-E achieve 12 best results and 13 second best results. Except the strict match on WikiWars dataset, both SynTime-I and SynTime-E achieve F_1 above 91%. For the relaxed match on all three datasets, SynTime-I and SynTime-E achieve recalls above 92%. The high recalls are consistent with our finding that at least 91.81% of time expressions contain time token(s). (See Table 2.) This indicates that SynTime covers most of time tokens. On Tweets dataset, SynTime-I and SynTime-E achieve exceptionally good performance. Their F_1 reach 91.74% with 11.37% improvement in strict match and 95.87% with 6.33% improvement in relaxed match. The reasons are that in informal environment people tend to use time expressions in minimum length, (62.91% of one-word time expressions in Tweets; see Figure 1.) the size of time keywords is small, (only 60 distinct time tokens; see Table 3.) and even in tweets people tend to use formal words. (See Section 4.3 for our finding from Twitter word clusters.) For precision, SynTime achieves comparable results in strict match and performs slightly poorer in relaxed match.

5.2.1 SynTime-I vs. Baseline Methods

On TimeBank dataset, SynTime-I achieves F_1 of 92.09% in strict match and of 94.96% in relaxed match. On Tweets, SynTime-I achieves 91.74% and 95.87%, respectively. It outperforms all the baseline methods. The reason is that for the rule-based time taggers, their rules are designed in a fixed way, lacking flexibility. For example, SUTime could recognize '1 year' but not 'year 1.' For the machine learning based methods, some of the features they used actually hurt the modelling. Time expressions involve quite many changing numbers which in themselves affect the pattern recognition. For example, it is difficult to build connection between 'May 22, 1986' and 'February 01, 1989' at the level of word or of character. One suggestion is to consider a type-based learning method that could use type information. For example, the above two time expressions refer to the same pattern of 'MONTH NUMERAL COMMA

[10] http://www.cs.rochester.edu/~naushad/tempeval3/tools.zip

Table 5: Number of time tokens and modifiers for expansion

Dataset	#Time Tokens	#Modifiers
TimeBank	3	5
WikiWars	16	21
Tweets	3	2

YEAR' at the level of token type. POS is a kind of type information. But according to our analysis, POS could not distinguish time expressions from common words. Features need carefully designing. On WikiWars, SynTime-I achieves competitive results in both matches. Time expressions in WikiWars include lots of prepositions and quite a few descriptive time expressions. SynTime could not fully recognize such kinds of time expressions because it follows TimeML and TimeBank.

5.2.2 SynTime-E vs. SynTime-I

Table 5 lists the number of time tokens and modifiers added to SynTime-I to get SynTime-E.

On TimeBank and Tweets datasets, only a few tokens are added, the corresponding results are affected slightly. This confirms that the size of time words is small, and that SynTime-I covers most of time words. On WikiWars dataset, relatively more tokens are added, SynTime-E performs much better than SynTime-I, especially in recall. It improves the recall by 3.25% in strict match and by 2.98% in relaxed match. This indicates that with more words added from specific domains (e.g., WikiWars dataset about war), SynTime can significantly improve the performance.

5.3 Limitations

SynTime assumes that words are tokenized and POS tagged correctly. In reality, however, the tokenized and tagged words are not that perfect, due to the limitation of used tools. For example, Stanford POS Tagger assigns VBD to the word 'sat' in 'friday or sat' while whose tag should be NNP. The incorrect tokens and POS tags affect the result.

6 Conclusion and future work

We conduct an analysis on time expressions from four datasets, and find that time expressions in general are very short and expressed by a small vocabulary, and words in time expressions demonstrate similar syntactic behavior. Our findings provide evidence in terms of time expression for the principle of least effort (Zipf, 1949). Inspired by part-of-speech, based on the findings, we define a syntactic type system for the time expression, and propose a type-based time expression tagger, named by SynTime. SynTime defines syntactic token types for tokens and on the token types it designs general heuristic rules based on the idea of boundary expansion. Experiments on three datasets show that SynTime outperforms the state-of-the-art baselines, including rule-based time taggers and machine learning based time tagger. Because our heuristic rules are quite simple, SynTime is light-weight and runs in real time.

Our token types and heuristic rules are independent of specific tokens, SynTime therefore is independent of specific domains, specific text types, and even specific languages that consist of specific tokens. In this paper, we test SynTime on specific domains and specific text types in English. The test for other languages needs only to construct a collection of token regular expressions in the target language under our defined token types.

Time expression is part of language and follows the principle of least effort. Since language usage relates to human habits (Zipf, 1949; Chomsky, 1986; Pinker, 1995), we might expect that humans would share some common habits, and therefore expect that other parts of language would more or less follow the same principle. In the future we will try our analytical method on other parts of language.

Acknowledgments

The authors would like to thank the three anonymous reviewers for their insightful comments and constructive suggestions. This research is mainly supported by the Singapore Ministry of Education Research Fund MOE2014-T2-2-066.

References

Omar Alonso, Jannik Strotgen, Ricardo Baeza-Yates, and Michael Gertz. 2011. Temporal information retrieval: Challenges and opportunities. In *Proceedings of 1st International Temporal Web Analytics Workshop*. pages 1–8.

Gabor Angeli, Christopher D. Manning, and Daniel Jurafsky. 2012. Parsing time: Learning to interpret time expressions. In *Proceedings of 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 446–455.

Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Proceedings of the 7th International Workshop on Semantic Evaluation*. pages 10–14.

Ricardo Campos, Gael Dias, Alipio M. Jorge, and Adam Jatowt. 2014. Survey of temporal information retrieval and related applications. *ACM Computing Surveys* 47(2):15.

Angel X. Chang and Christopher D. Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *Proceedings of 8th International Conference on Language Resources and Evaluation*. pages 3735–3740.

Angel X. Chang and Christopher D. Manning. 2013. Sutime: Evaluation in tempeval-3. In *Proceedings of second Joint Conference on Lexical and Computational Semantics (SEM)*. pages 78–82.

Angel X. Chang and Christopher D. Manning. 2014. Tokensregex: Defining cascaded regular expressions over tokens. Technical report, Department of Computer Science, Stanford University.

Noam Chomsky. 1986. *Knowledge of Language: Its Nature, Origin, and Use*. New York: Prager.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12:2121–2159.

Michele Filannino, Gavin Brown, and Goran Nenadic. 2013. Mantime: Temporal expression identification and normalization in the tempeval-3 challenge. In *Proceedings of the 7th International Workshop on Semantic Evaluation*.

Jerry R. Hobbs, Douglas E. Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1997. Fastus: A cascaded finite-state transducer for extracting information from natrual-language text. In *Finite State Devices for Natural Language Processing*. pages 383–406.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*. pages 1437–1447.

Hector Llorens, Leon Derczynski, Robert Gaizauskas, and Estela Saquete. 2012. Timen: An open temporal expression normalisation resource. In *Proceedings of 8th International Conference on Language Resources and Evaluation*. pages 3044–3051.

Hector Llorens, Estela Saquete, and Borja Navarro. 2010. Tipsem (english and spanish): Evaluating crfs and semantic roles in tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. pages 284–291.

Christopher Manning and Hinrich Schutze. 1999. *Foundations of Statistical Natural Language Processing*. Cambride: MIT Press.

Pawel Mazur and Robert Dale. 2010. Wikiwars: A new corpus for research on temporal expressions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 913–922.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. Engilish gigaword fifth edition.

Steven Pinker. 1995. *The language instinct: The new science of language and mind*, volume 7529. Penguin.

James Pustejovsky, Jose Castano, Robert Ingria, Roser Sauri, Robert Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir Radev. 2003a. Timeml: Robust specification of event and temporal expressions in text. *New Directions in Question Answering* 3:28–34.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Beth Sundheim, Dragomir Radev, David Day, Lisa Ferro, and Marcia Lazo. 2003b. The timebank corpus. *Corpus Linguistics* 2003:647–656.

Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press.

Jannik Strötgen and Michael Gertz. 2010. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval'10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 321–324.

Jannik Strotgen, Julian Zell, and Michael Gertz. 2013. Heideltime: Tuning english and developing spanish resources. In *Proceedings of second Joint Conference on Lexical and Computational Semantics (SEM)*. pages 15–19.

Jeniya Tabassum, Alan Ritter, and Wei Xu. 2016. Tweetime: A minimally supervised method for recognizing and normalizing time expressions in twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 307–318.

Naushad UzZaman and James F. Allen. 2010. Trips and trios system for tempeval-2: Extracting temporal information from text. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. pages 276–283.

Naushad UzZaman, Hector Llorens, Leon Derczynski, Marc Verhagen, James Allen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation*. pages 1–9.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluation*. pages 75–80.

Marc Verhagen, Inderjeet Mani, Roser Sauri, Robert Knippen, Seok Bae Jang, Jessica Littman, Anna Rumshisky, John Phillips, Inderjeet Mani, Roser Sauri, Robert Knippen, Seok Bae Jang, Jessica Littman, Anna Rumshisky, John Phillips, and James Pustejovsky. 2005. Automating temporal annotation with tarqi. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions.*. pages 81–84.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. pages 57–62.

George Zipf. 1949. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley Press, Inc.

Learning with Noise: Enhance Distantly Supervised Relation Extraction with Dynamic Transition Matrix

Bingfeng Luo[1], **Yansong Feng**[*1], **Zheng Wang**[2], **Zhanxing Zhu**[3],
Songfang Huang[4], **Rui Yan**[1] and **Dongyan Zhao**[1]
[1]ICST, Peking University, China
[2]School of Computing and Communications, Lancaster University, UK
[3]Peking University, China
[4]IBM China Research Lab, China
{bf_luo,fengyansong,zhanxing.zhu,ruiyan,zhaody}@pku.edu.cn
z.wang@lancaster.ac.uk
huangsf@cn.ibm.com

Abstract

Distant supervision significantly reduces human efforts in building training data for many classification tasks. While promising, this technique often introduces noise to the generated training data, which can severely affect the model performance. In this paper, we take a deep look at the application of distant supervision in relation extraction. We show that the dynamic transition matrix can effectively characterize the noise in the training data built by distant supervision. The transition matrix can be effectively trained using a novel curriculum learning based method without any direct supervision about the noise. We thoroughly evaluate our approach under a wide range of extraction scenarios. Experimental results show that our approach consistently improves the extraction results and outperforms the state-of-the-art in various evaluation scenarios.

1 Introduction

Distant supervision (DS) is rapidly emerging as a viable means for supporting various classification tasks – from relation extraction (Mintz et al., 2009) and sentiment classification (Go et al., 2009) to cross-lingual semantic analysis (Fang and Cohn, 2016). By using knowledge learned from seed examples to label data, DS automatically prepares large scale training data for these tasks.

While promising, DS does not guarantee perfect results and often introduces noise to the generated data. In the context of relation extraction, DS works by considering sentences containing both the subject and object of a <*subj*, rel, *obj*> triple as its supports. However, the generated data are not always perfect. For instance, DS could match the knowledge base (KB) triple, <*Donald Trump*, born-in, *New York*> in *false positive* contexts like *Donald Trump worked in New York City*. Prior works (Takamatsu et al., 2012; Ritter et al., 2013) show that DS often mistakenly labels real positive instances as negative (*false negative*) or versa vice (*false positive*), and there could be confusions among positive labels as well. These noises can severely affect training and lead to poorly-performing models.

Tackling the noisy data problem of DS is non-trivial, since there usually lacks of explicit supervision to capture the noise. Previous works have tried to remove sentences containing unreliable syntactic patterns (Takamatsu et al., 2012), design new models to capture certain types of noise or aggregate multiple predictions under the *at-least-one assumption* that at least one of the aligned sentences supports the triple in KB (Riedel et al., 2010; Surdeanu et al., 2012; Ritter et al., 2013; Min et al., 2013). These approaches represent a substantial leap forward towards making DS more practical. however, are either tightly couple to certain types of noise, or have to rely on manual rules to filter noise, thus unable to scale. Recent breakthrough in neural networks provides a new way to reduce the influence of incorrectly labeled data by aggregating multiple training instances attentively for relation classification, without explicitly characterizing the inherent noise (Lin et al., 2016; Zeng et al., 2015). Although promising, modeling noise within neural network architectures is still in its early stage and much remains to be done.

In this paper, we aim to enhance DS noise modeling by providing the capability to explicitly characterize the noise in the DS-style training data

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 430–439
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1040

within neural networks architectures. We show that while noise is inevitable, it is possible to characterize the noise pattern in a unified framework along with its original classification objective. Our key insight is that the DS-style training data typically contain useful clues about the noise pattern. For example, we can infer that since some people work in their birthplaces, DS could wrongly label a training sentence describing a working place as a born-in relation. Our novel approach to noisy modeling is to use a dynamically-generated transition matrix for each training instance to (1) characterize the possibility that the DS labeled relation is confused and (2) indicate its noise pattern. To tackle the challenge of no direct guidance over the noise pattern, we employ a curriculum learning based training method to gradually model the noise pattern over time, and utilize trace regularization to control the behavior of the transition matrix during training. Our approach is flexible – while it does not make any assumptions about the data quality, the algorithm can make effective use of the data-quality prior knowledge to guide the learning procedure when such clues are available.

We apply our method to the relation extraction task and evaluate under various scenarios on two benchmark datasets. Experimental results show that our approach consistently improves both extraction settings, outperforming the state-of-the-art models in different settings.

Our work offers an effective way for tackling the noisy data problem of DS, making DS more practical at scale. Our main contributions are to (1) design a *dynamic* transition matrix structure to characterize the noise introduced by DS, and (2) design a curriculum learning based framework to adaptively guide the training procedure to learn with noise.

2 Problem Definition

The task of distantly supervised relation extraction is to extract knowledge triples, <*subj*, rel, *obj*>, from free text with the training data constructed by aligning existing KB triples with a large corpus. Specifically, given a triple in KB, DS works by first retrieving all the sentences containing both *subj* and *obj* of the triple, and then constructing the training data by considering these sentences as support to the existence of the triple. This task can be conducted in both the sentence and the bag levels. The former takes a sentence s containing both $subj$ and obj as input, and outputs the relation expressed by the sentence between $subj$ and obj. The latter setting alleviates the noisy data problem by using the ***at-least-one assumption*** that at least one of the retrieved sentences containing both $subj$ and obj supports the <*subj*, rel, *obj*> triple. It takes a bag of sentences S as input where each sentence $s \in S$ contains both $subj$ and obj, and outputs the relation between $subj$ and obj expressed by this bag.

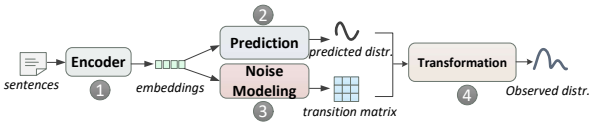


Figure 1: Overview of our approach

3 Our approach

In order to deal with the noisy training data obtained through DS, our approach follows four steps as depicted in Figure 1. First, each input sentence is fed to a sentence encoder to generate an embedding vector. Our model then takes the sentence embeddings as input and produce a predicted relation distribution, $\mathbf{p}$, for the input sentence (or the input sentence bag). At the same time, our model dynamically produces a transition matrix, $\mathbf{T}$, which is used to characterize the noise pattern of sentence (or the bag). Finally, the predicted distribution is multiplied by the transition matrix to produce the observed relation distribution, $\mathbf{o}$, which is used to match the noisy relation labels assigned by DS while the predicted relation distribution $\mathbf{p}$ serves as output of our model during testing. One of the key challenges of our approach is on determining the element values of the transition matrix, which will be described in Section 4.

3.1 Sentence-level Modeling

Sentence Embedding and Prediction In this work, we use a piecewise convolutional neural network (Zeng et al., 2015) for sentence encoding, but other sentence embedding models can also be used. We feed the sentence embedding to a full connection layer, and use *softmax* to generate the predicted relation distribution, $\mathbf{p}$.

Noise Modeling First, each sentence embedding $\mathbf{x}$, generated b sentence encoder, is passed to a full connection layer as a non-linearity to obtain the sentence embedding $\mathbf{x}_n$ used specifically for noise modeling. We then use *softmax* to calculate the

transition matrix $\mathbf{T}$, for each sentence:

$$T_{ij} = \frac{exp(\mathbf{w}_{ij}^T \mathbf{x}_n + b)}{\sum_{j=1}^{|\mathbb{C}|} exp(\mathbf{w}_{ij}^T \mathbf{x}_n + b)} \quad (1)$$

where T_{ij} is the conditional probability for the input sentence to be labeled as relation j by DS, given i as the true relation, b is a scalar bias, $|\mathbb{C}|$ is the number of relations, $\mathbf{w}_{ij}$ is the weight vector characterizing the confusion between i and j.

Here, we dynamically produce a transition matrix, $\mathbf{T}$, specifically for each sentence, but with the parameters ($\mathbf{w}_{ij}$) shared across the dataset. By doing so, we are able to adaptively characterize the noise pattern for each sentence, with a few parameters only. In contrast, one could also produce a global transition matrix for all sentences, with much less computation, where one need not to compute $\mathbf{T}$ on the fly (see Section 6.1).

Observed Distribution When we characterize the noise in a sentence with a transition matrix $\mathbf{T}$, if its true relation is i, we can assume that i might be erroneously labeled as relation j by DS with probability T_{ij}. We can therefore capture the observed relation distribution, $\mathbf{o}$, by multiplying $\mathbf{T}$ and the predicted relation distribution, $\mathbf{p}$:

$$\mathbf{o} = \mathbf{T}^T \cdot \mathbf{p} \quad (2)$$

where $\mathbf{o}$ is then normalized to ensure $\sum_i o_i = 1$.

Rather than using the predicted distribution $\mathbf{p}$ to directly match the relation labeled by DS (Zeng et al., 2015; Lin et al., 2016), here we utilize $\mathbf{o}$ to match the noisy labels during training and still use $\mathbf{p}$ as output during testing, which actually captures the procedure of how the noisy label is produced and thus protects $\mathbf{p}$ from the noise.

3.2 Bag Level Modeling

Bag Embedding and Prediction One of the key challenges for bag level model is how to aggregate the embeddings of individual sentences into the bag level. In this work, we experiment two methods, namely average and attention aggregation (Lin et al., 2016). The former calculates the bag embedding, $\mathbf{s}$, by averaging the embeddings of each sentence, and then feed it to a *softmax* classifier for relation classification.

The attention aggregation calculates an attention value, a_{ij}, for each sentence i in the bag with respect to each relation j, and aggregates to the bag level as $\mathbf{s}_j$, by the following equations[1]:

$$\mathbf{s}_j = \sum_i^n a_{ij}\mathbf{x}_i;\ a_{ij} = \frac{exp(\mathbf{x}_i^T \mathbf{r}_j)}{\sum_{i'}^n exp(\mathbf{x}_{i'}^T \mathbf{r}_j)} \quad (3)$$

where $\mathbf{x}_i$ is the embedding of sentence i, n the number of sentences in the bag, and $\mathbf{r}_j$ is the randomly initialized embedding for relation j. In similar spirit to (Lin et al., 2016), the resulting bag embedding $\mathbf{s}_j$ is fed to a *softmax* classifier to predict the probability of relation j for the given bag.

Noise Modeling Since the transition matrix addresses the transition probability with respect to each true relation, the attention mechanism appears to be a natural fit for calculating the transition matrix in bag level. Similar to attention aggregation above, we calculate the bag embedding with respect to each relation using Equation 3, but with a separate set of relation embeddings $\mathbf{r'}_j$. We then calculate the transition matrix, $\mathbf{T}$, by:

$$T_{ij} = \frac{exp(\mathbf{s}_i^T \mathbf{r'}_j + b_i)}{\sum_{j=1}^{|\mathbb{C}|} exp(\mathbf{s}_i^T \mathbf{r'}_j + b_i)} \quad (4)$$

where $\mathbf{s}_i$ is the bag embedding regarding relation i, and $\mathbf{r'}_j$ is the embedding for relation j.

4 Curriculum Learning based Training

One of the key challenges of this work is on how to train and produce the transition matrix to model the noise in the training data without any direct guidance and human involvement. A straightforward solution is to directly align the observed distribution, $\mathbf{o}$, with respect to the noisy labels by minimizing the sum of the two terms: $CrossEntropy(\mathbf{o}) + Regularization$. However, doing so does not guarantee that the prediction distribution, $\mathbf{p}$, will match the true relation distribution. The problem is at the beginning of the training, we have no prior knowledge about the noise pattern, thus, both $\mathbf{T}$ and $\mathbf{p}$ are less reliable, making the training procedure be likely to trap into some poor local optimum. Therefore, we require a technique to guide our model to gradually adapt to the noisy training data, e.g., learning something simple first, and then trying to deal with noises.

[1] While (Lin et al., 2016) use bilinear function to calculate a_{ij}, we simply use dot product since we find these two functions perform similarly in our experiments.

Fortunately, this is exactly what curriculum learning can do. The idea of curriculum learning (Bengio et al., 2009) is simple: starting with the easiest aspect of a task, and leveling up the difficulty gradually, which fits well to our problem. We thus employ a curriculum learning framework to guide our model to gradually learn how to characterize the noise. Another advantage is to avoid falling into poor local optimum.

With curriculum learning, our approach provides the flexibility to combine prior knowledge of noise, e.g., splitting a dataset into reliable and less reliable subsets, to improve the effectiveness of the transition matrix and better model the noise.

4.1 Trace Regularization

Before proceeding to training details, we first discuss how we characterize the noise level of the data by controlling the trace of its transition matrix. Intuitively, if the noise is small, the transition matrix $\mathbf{T}$ will tend to become an identity matrix, i.e., given a set of annotated training sentences, the observed relations and their true relations are almost identical. Since each row of $\mathbf{T}$ sums to 1, the similarity between the transition matrix and the identity matrix can be represented by its trace, $trace(\mathbf{T})$. The larger the $trace(\mathbf{T})$ is, the larger the diagonal elements are, and the more similar the transition matrix $\mathbf{T}$ is to the identity matrix, indicating a lower level of noise. Therefore, we can characterize the noise pattern by controlling the expected value of $trace(\mathbf{T})$ in the form of regularization. For example, we will expect a larger $trace(\mathbf{T})$ for reliable data, but a smaller $trace(\mathbf{T})$ for less reliable data. Another advantage of employing trace regularization is that it could help reduce the model complexity and avoid overfitting.

4.2 Training

To tackle the challenge of no direct guidance over the noise patterns, we implement a curriculum learning based training method to first train the model without considerations for noise. In other words, we first focus on the loss from the prediction distribution $\mathbf{p}$, and then take the noise modeling into account gradually along the training process, i.e., gradually increasing the importance of the loss from the observed distribution $\mathbf{o}$ while decreasing the importance of $\mathbf{p}$. In this way, the prediction branch is roughly trained before the model managing to characterize the noise, thus avoids being stuck into poor local optimum. We thus design to minimize the following loss function:

$$L = \sum_{i=1}^{N} -((1-\alpha)log(o_{iy_i}) + \alpha log(p_{iy_i})) - \beta trace(\mathbf{T}^i) \quad (5)$$

where $0<\alpha\leq 1$ and $\beta>0$ are two weighting parameters, y_i is the relation assigned by DS for the i-th instance, N the total number of training instances, o_{iy_i} is the probability that the observed relation for the i-th instance is y_i, and p_{iy_i} is the probability to predict relation y_i for the i-th instance.

Initially, we set α=1, and train our model completely by minimizing the loss from the prediction distribution $\mathbf{p}$. That is, we do not expect to model the noise, but focus on the prediction branch at this time. As the training progresses, the prediction branch gradually learns the basic prediction ability. We then decrease α and β by $0<\rho<1$ ($\alpha^*=\rho\alpha$ and $\beta^*=\rho\beta$) every τ epochs, i.e., learning more about the noise from the observed distribution $\mathbf{o}$ and allowing a relatively smaller $trace(\mathbf{T})$ to accommodate more noise. The motivation behind is to put more and more effort on learning the noise pattern as the training proceeds, with the essence of curriculum learning. This gradually learning paradigm significantly distinguishes from prior work on noise modeling for DS seen to date. Moreover, as such a method does not rely on any extra assumptions, it can serve as our default training method for $\mathbf{T}$.

With Prior Knowledge of Data Quality On the other hand, if we happen to have prior knowledge about which part of the training data is more reliable and which is less reliable, we can utilize this knowledge as guidance to design the curriculum. Specifically, we can build a curriculum by first training the prediction branch on the reliable data for several epochs, and then adding the less reliable data to train the full model. In this way, the prediction branch is roughly trained before exposed to more noisy data, thus is less likely to fall into poor local optimum.

Furthermore, we can take better control of the training procedure with trace regularization, e.g., encouraging larger $trace(\mathbf{T})$ for reliable subset and smaller $trace(\mathbf{T})$ for less relaibale ones. Specifically, we propose to minimize:

$$L = \sum_{m=1}^{M}\sum_{i=1}^{N_m} -log(o_{mi,y_{mi}}) - \beta_m trace(\mathbf{T}^{mi}) \quad (6)$$

where β_m is the regularization weight for the m-th data subset, M is the total number of subsets, N_m the number of instances in m-th subset, and $\mathbf{T}^{mi}$, y_{mi} and $o_{mi,y_{mi}}$ are the transition matrix, the relation labeled by DS and the observed probability of this relation for the i-th training instance in the m-th subset, respectively. Note that different from Equation 5, this loss function does not need to initiate training by minimizing the loss regarding the prediction distribution $\mathbf{p}$, since one can easily start by learning from the most reliable split first.

We also use trace regularization for the most reliable subset, since there are still some noise annotations inevitably appearing in this split. Specifically, we expect its $trace(\mathbf{T})$ to be large (using a positive β) so that the elements of $\mathbf{T}$ will be centralized to the diagonal and $\mathbf{T}$ will be more similar to the identity matrix. As for the less reliable subset, we expect the $trace(\mathbf{T})$ to be small (using a negative β) so that the elements of the transition matrix will be diffusive and $\mathbf{T}$ will be less similar to the identity matrix. In other words, the transition matrix is encouraged to characterize the noise.

Note that this loss function only works for sentence level models. For bag level models, since reliable and less reliable sentences are all aggregated into a sentence bag, we can not determine which bag is reliable and which is not. However, bag level models can still build a curriculum by changing the content of a bag, e.g., keeping reliable sentences in the bag first, then gradually adding less reliable ones, and training with Equation 5, which could benefit from the prior knowledge of data quality as well.

5 Evaluation Methodology

Our experiments aim to answer two main questions: (1) is it possible to model the noise in the training data generated through DS, even when there is no prior knowledge to guide us? and (2) whether the prior knowledge of data quality can help our approach better handle the noise.

We apply our approach to both sentence level and bag level extraction models, and evaluate in the situations where we do not have prior knowledge of the data quality as well as where such prior knowledge is available.

5.1 Datasets

We evaluate our approach on two datasets.

TIMERE We build TIMERE by using DS to align time-related Wikidata (Vrandečić and Krötzsch, 2014) KB triples to Wikipedia text. It contains 278,141 sentences with 12 types of relations between an entity mention and a time expression. We choose to use time-related relations because time expressions speak for themselves in terms of reliability. That is, given a KB triple $<e$, rel, $t>$ and its aligned sentences, the finer-grained the time expression t appears in the sentence, the more likely the sentence supports the existence of this triple. For example, a sentence containing both *Alphabet* and *October-2-2015* is very likely to express the inception-time of *Alphabet*, while a sentence containing both *Alphabet* and *2015* could instead talk about many events, e.g., releasing financial report of 2015, hiring a new CEO, etc. Using this heuristics, we can split the dataset into 3 subsets according to different granularities of the time expressions involved, indicating different levels of reliability. Our criteria for determining the reliability are as follows. Instances with full date expressions, i.e., Year-Month-Day, can be seen as the most reliable data, while those with partial date expressions, e.g., Month-Year and Year-Only, are considered as less reliable. Negative data are constructed heuristically that any *entity-time* pairs in a sentence without corresponding triples in Wikidata are treated as negative data. During training, we can access 184,579 negative and 77,777 positive sentences, including 22,214 reliable, 2,094 and 53,469 less reliable ones. The validation set and test set are randomly sampled from the reliable (full-date) data for relatively fair evaluations and contains 2,776, 2,771 positive sentences and 5,143, 5,095 negative sentences, respectively.

ENTITYRE is a widely-used entity relation extraction dataset, built by aligning triples in Freebase to the New York Times (NYT) corpus (Riedel et al., 2010). It contains 52 relations, 136,947 positive and 385,664 negative sentences for training, and 6,444 positive and 166,004 negative sentences for testing. Unlike TIMERE, this dataset does not contain any prior knowledge about the data quality. Since the sentence level annotations in ENTITYRE are too noisy to serve as gold standard, we only evaluate bag-level models on ENTITYRE, a standard practice in previous works (Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016).

5.2 Experimental Setup

Hyper-parameters We use 200 convolution kernels with widow size 3. During training, we use stochastic gradient descend (SGD) with batch size 20. The learning rates for sentence-level and bag-level models are 0.1 and 0.01, respectively.

Sentence level experiments are performed on TIMERE, using 100-d word embeddings pre-trained using GloVe (Pennington et al., 2014) on Wikipedia and Gigaword (Parker et al., 2011), and 20-d vectors for distance embeddings. Each of the three subsets of TIMERE is added after the previous phase has run for 15 epochs. The trace regularization weights are $\beta_1 = 0.01$, $\beta_2 = -0.01$ and $\beta_3 = -0.1$, respectively, from the reliable to the most unreliable, with the ratio of β_3 and β_2 fixed to 10 or 5 when tuning.

Bag level experiments are performed on both TIMERE and ENTITYRE. For TIMERE, we use the same parameters as above. For ENTITYRE, we use 50-d word embeddings pre-trained on the NYT corpus using word2vec (Mikolov et al., 2013), and 5-d vectors for distance embedding. For both datasets, α and β in Eq. 5 are initialized to 1 and 0.1, respectively. We tried various decay rates, {0.95, 0.9, 0.8}, and steps, {3, 5, 8}. We found that using a decay rate of 0.9 with step of 5 gives best performance in most cases.

Evaluation Metric The performance is reported using the precision-recall (PR) curve, which is a standard evaluation metric in relation extraction. Specifically, the extraction results are first ranked decreasingly by their confidence scores, then the precision and recall are calculated by setting the threshold to be the score of each extraction result one by one.

Naming Conventions We evaluate our approach under a wide range of settings for sentence level (sent_) and bag level (bag_) models: (1) _mix: trained on all three subsets of TIMERE mixed together; (2) _reliable: trained using the reliable subset of TIMERE only; (3) _PR: trained with prior knowledge of annotation quality, i.e., starting from the reliable data and then adding the unreliable data; (4) _TM: trained with dynamic transition matrix; (5) _GTM: trained with a global transition matrix. In bag level, we also investigate the performance of average aggregation (_avg) and attention aggregation (_att).

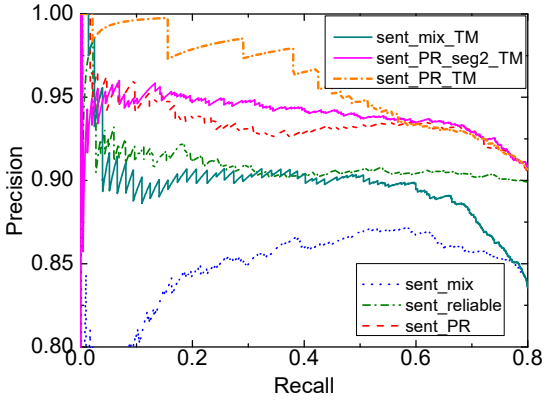


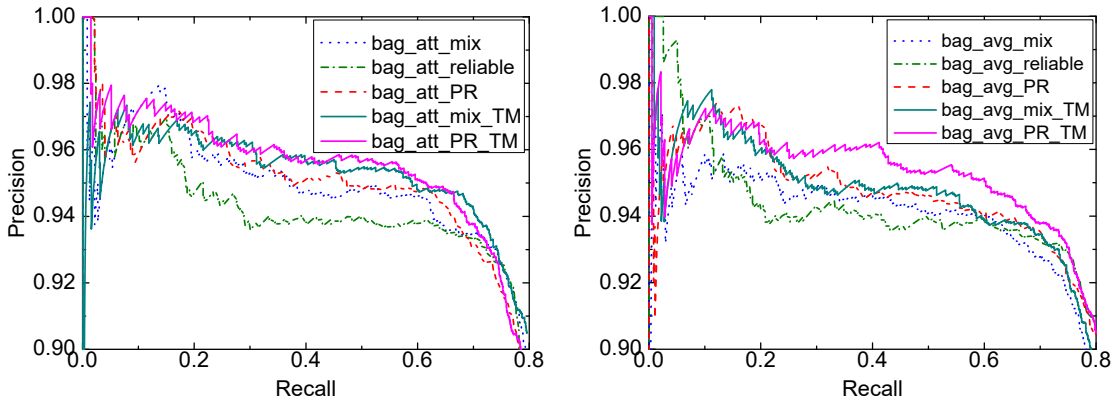
Figure 2: Sentence Level Results on TIMERE

6 Experimental Results

6.1 Performance on TIMERE

Sentence Level Models The results of sentence level models on TIMERE are shown in Figure 2. We can see that mixing all subsets together (sent_mix) gives the worst performance, significantly worse than using the reliable subset only (sent_reliable). This suggests the noisy nature of the training data obtained through DS and properly dealing with the noise is the key for DS for a wider range of applications. When getting help from our dynamic transition matrix, the model (sent_mix_TM) significantly improves sent_mix, delivering the same level of performance as sent_reliable in most cases. This suggests that our transition matrix can help to mitigate the bad influence of noisy training instances.

Now let us consider the PR scenario where one can build a curriculum by first training on the reliable subset, then gradually moving to both reliable and less reliable data. We can see that, this simple curriculum learning based model (sent_PR) further outperforms sent_reliable significantly, indicating that the curriculum learning framework not only reduces the effect of noise, but also helps the model learn from noisy data. When applying the transition matrix approach into this curriculum learning framework using one reliable subset and one unreliable subset generated by mixing our two less reliable subsets, our model (sent_PR_seg2_TM) further improves sent_PR by utilizing the dynamic transition matrix to model the noise. It is not surprising that when we use all three subsets separately, our model (sent_PR_TM) significantly outperforms all other models by a large margin.



(a) Attention Aggregation

(b) Average Aggregation

Figure 3: Bag Level Results on TIMERE

Bag Level Models In this setting, we first look at the performance of the bag level models with attention aggregation. The results are shown in Figure 3(a). Consider the comparison between the model trained on the reliable subset only (`bag_att_reliable`) and the one trained on the mixed dataset (`bag_att_mix`). In contrast to the sentence level, `bag_att_mix` outperforms `bag_att_reliable` by a large margin, because `bag_att_mix` has taken the *at-least-one assumption* into consideration through the attention aggregation mechanism (Eq. 3), which can be seen as a denoising step within the bag. This may also be the reason that when we introduce either our dynamic transition matrix (`bag_att_mix_TM`) or the curriculum of using prior knowledge of data quality (`bag_att_PR`) into the bag level models, the improvement regarding `bag_att_mix` is not as significant as in the sentence level.

However, when we apply our dynamic transition matrix into the curriculum built upon prior knowledge of data quality (`bag_att_PR_TM`), the performance gets further improved. This happens especially in the high precision part compared to `bag_att_PR`. We also note that the bag level's *at-least-one assumption* does not always hold, and there are still false negative and false positive problems. Therefore, using our transition matrix approach with or without prior knowledge of data quality, i.e., `bag_att_mix_TM` and `bag_att_PR_TM`, both improve the performance, and `bag_att_PR_TM` performs slightly better.

The results of bag level models with average aggregation are shown in Figure 3(b), where the relative ranking of various settings is similar to those with attention aggregation. A notable difference is that both `bag_avg_PR` and `bag_avg_mix_TM` improve `bag_avg_mix` by a larger margin compared to that in the attention aggregation setting. The reason may be that the average aggregation mechanism is not as good as the attention aggregation in denoising within the bag, which leaves more space for our transition matrix approach or curriculum learning with prior knowledge to improve. Also note that `bag_avg_reliable` performs best in the very-low-recall region but worst in general. This is because that it ranks higher the sentences expressing either `birth-date` or `death-date`, the simplest but the most common relations in the dataset, but fails to learn other relations with limited or noisy training instances, given its relatively simple aggregation strategy.

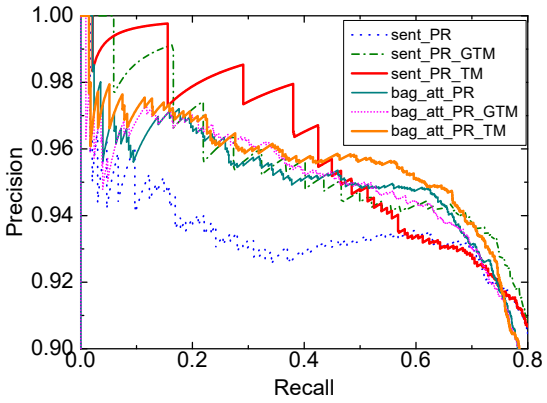


Figure 4: Global TM v.s. Dynamic TM

Global v.s. Dynamic Transition Matrix We also compare our dynamic transition matrix method with the global transition matrix method, which maintains only one transition matrix for all training instances. Specifically, instead of dynam-

ically generating a transition matrix for each datum, we first initialize an identity matrix $\mathbf{T}' \in \mathbb{R}^{|\mathbb{C}|\times|\mathbb{C}|}$, where $|\mathbb{C}|$ is the number of relations (including `no-relation`). Then the global transition matrix $\mathbf{T}$ is built by applying *softmax* to each row of $\mathbf{T}'$ so that $\sum_j \mathbf{T}_{ij} = 1$:

$$T_{ij} = \frac{e^{T'_{ij}}}{\sum_{j=1}^{|\mathbb{C}|} e^{T'_{ij}}} \quad (7)$$

where T_{ij} and T'_{ij} are the elements in the i^{th} row and j^{th} column of $\mathbf{T}$ and $\mathbf{T}'$. The element values of matrix $\mathbf{T}'$ are also updated via backpropagation during training. As shown in Figure 4, using one global transition matrix (`_GTM`) is also beneficial and improves both the sentence level (`sent_PR`) and bag level (`bag_att_PR`) models. However, since the global transition matrix only captures the global noise pattern, it fails to characterize individuals with subtle differences, resulting in a performance drop compared to the dynamic one (`_TM`).

Case Study We find our transition matrix method tends to obtain more significant improvement on noisier relations. For example, *time_of_spacecraft_landing* is noisier than *time_of_spacecraft_launch* since compared to the launching of a spacecraft, there are fewer sentences containing the landing time of a spacecraft that talks directly about the landing. Instead, many of these sentences tend to talk about the activities of the crew. Our `sent_PR_TM` model improves the F1 of *time_of_spacecraft_landing* and *time_of_spacecraft_launch* over `sent_PR` by 9.09% and 2.78%, respectively. The transition matrix makes more significant improvement on *time_of_spacecraft_landing* since there are more noisy sentences for our method to handle, which results in more significant improvement on the quality of the training data.

6.2 Performance on ENTITYRE

We evaluate our bag level models on ENTITYRE. As shown in Figure 5, it is not surprising that the basic model with attention aggregation (`att`) significantly outperforms the average one (`avg`), where `att` in our bag embedding is similar in spirit to (Lin et al., 2016), which has reported the-state-of-the-art performance on ENTITYRE. When injected with our transition matrix approach, both `att_TM` and `avg_TM` clearly outperform their basic versions.

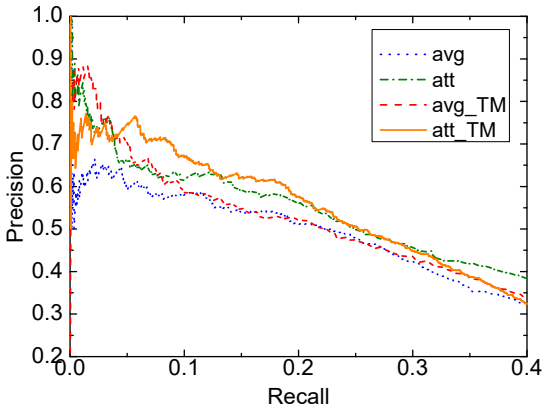


Figure 5: Results on ENTITYRE

Method	P@R_10	P@R_20	P@R_30
Mintz	39.88	28.55	16.81
MultiR	60.94	36.41	-
MIML	60.75	33.82	-
avg	58.04	51.25	42.45
avg_TM	58.56	52.35	43.59
att	61.51	56.36	**45.63**
att_TM	**67.24**	**57.61**	44.90

Table 1: Comparison with feature-based methods. P@R_10/20/30 refers to the precision when recall equals 10%, 20% and 30%.

Similar to the situations in TIMERE, since `att` has taken the *at-least-one assumption* into account through its attention-based bag embedding mechanism, thus the improvement made by `att_TM` is not as large as by `avg_TM`.

We also include the comparison with three feature-based methods: `Mintz` (Mintz et al., 2009) is a multiclass logistic regression model; `MultiR` (Hoffmann et al., 2011) is a probabilistic graphical model that can handle overlapping relations; `MIML` (Surdeanu et al., 2012) is also a probabilistic graphical model but operates in the multi-instance multi-label paradigm. As shown in Table 1, although traditional feature-based methods have reasonable results in the low recall region, their performances drop quickly as the recall goes up, and `MultiR` and `MIML` did not even reach the 30% recall. This indicates that, while human-designed featurs can effectively capture certain relation patterns, their coverage is relatively low. On the other hand, neural network models have more stable performance across different recalls, and `att_TM` performs generally better than other models, indicating again the effectiveness of our transition matrix method.

7 Related Work

In addition to relation extraction, distant supervision (DS) is shown to be effective in generating training data for various NLP tasks, e.g., tweet sentiment classification (Go et al., 2009), tweet named entity classifying (Ritter et al., 2011), etc. However, these early applications of DS do not well address the issue of data noise.

In relation extraction (RE), recent works have been proposed to reduce the influence of wrongly labeled data. The work presented by (Takamatsu et al., 2012) removes potential noisy sentences by identifying bad syntactic patterns at the preprocessing stage. (Xu et al., 2013) use pseudo-relevance feedback to find possible false negative data. (Riedel et al., 2010) make the *at-least-one assumption* and propose to alleviate the noise problem by considering RE as a multi-instance classification problem. Following this assumption, people further improves the original paradigm using probabilistic graphic models (Hoffmann et al., 2011; Surdeanu et al., 2012), and neural network methods (Zeng et al., 2015). Recently, (Lin et al., 2016) propose to use attention mechanism to reduce the noise within a sentence bag. Instead of characterizing the noise, these approaches only aim to alleviate the effect of noise.

The *at-least-one assumption* is often too strong in practice, and there are still chances that the sentence bag may be false positive or false negative. Thus it is important to model the noise pattern to guide the learning procedure. (Ritter et al., 2013) and (Min et al., 2013) try to employ a set of latent variables to represent the true relation. Our approach differs from them in two aspects. We target noise modeling in neutral networks while they target probabilistic graphic models. We further advance their models by providing the capability to model the fine-grained transition from the true relation to the observed, and the flexibility to combine indirect guidance.

Outside of NLP, various methods have been proposed in computer vision to model the data noise using neural networks. (Sukhbaatar et al., 2015) utilize a global transition matrix with weight decay to transform the true label distribution to the observed. (Reed et al., 2014) use a hidden layer to represent the true label distribution but try to force it to predict both the noisy label and the input. (Chen and Gupta, 2015; Xiao et al., 2015) first estimate the transition matrix on a clean dataset and apply to the noisy data. Our model shares similar spirit with (Misra et al., 2016) in that we all dynamically generate a transition matrix for each training instance, but, instead of using vanilla SGD, we train our model with a novel curriculum learning training framework with trace regularization to control the behavior of transition matrix. In NLP, the only work in neural-network-based noise modeling is to use one single global transition matrix to model the noise introduced by cross-lingual projection of training data (Fang and Cohn, 2016). Our work advances them through generating a transition matrix dynamically for each instance, to avoid using one single component to characterize both reliable and unreliable data.

8 Conclusions

In this paper, we investigate the noise problem inherent in the DS-style training data. We argue that the data speak for themselves by providing useful clues to reveal their noise patterns. We thus propose a novel transition matrix based method to dynamically characterize the noise underlying such training data in a unified framework along the original prediction objective. One of our key innovations is to exploit a curriculum learning based training method to gradually learn to model the underlying noise pattern without direct guidance, and to provide the flexibility to exploit any prior knowledge of the data quality to further improve the effectiveness of the transition matrix. We evaluate our approach in two learning settings of the distantly supervised relation extraction. The experimental results show that the proposed method can better characterize the underlying noise and consistently outperform start-of-the-art extraction models under various scenarios.

Acknowledgement

This work is supported by the National High Technology R&D Program of China (2015AA015403); the National Natural Science Foundation of China (61672057, 61672058); KLSTSPI Key Lab. of Intelligent Press Media Technology; the UK Engineering and Physical Sciences Research Council under grants EP/M01567X/1 (SANDeRs) and EP/M015793/1 (DIVIDEND); and the Royal Society International Collaboration Grant (IE161012).

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*. ACM, pages 41–48.

Xinlei Chen and Abhinav Gupta. 2015. Webly supervised learning of convolutional networks. In *ICCV*. pages 1431–1439.

Meng Fang and Trevor Cohn. 2016. Learning when to trust distant supervision: An application to low-resource pos tagging using cross-lingual projection. In *CONLL*. pages 178–186.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford* 1(12).

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*. pages 541–550.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*. volume 1, pages 2124–2133.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. pages 3111–3119.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *HLT-NAACL*. pages 777–782.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*. pages 1003–1011.

Ishan Misra, C Lawrence Zitnick, Margaret Mitchell, and Ross Girshick. 2016. Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels. In *CVPR*. pages 2930–2939.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, linguistic data consortium. Technical report, Linguistic Data Consortium, Philadelphia.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2014. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596* .

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 148–163.

Alan Ritter, Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *EMNLP*. Association for Computational Linguistics, pages 1524–1534.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *TACL* 1:367–378.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. 2015. Training convolutional networks with noisy labels. In *ICLR*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*. pages 455–465.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *ACL*. pages 721–729.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10):78–85.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from massive noisy labeled data for image classification. In *CVPR*. pages 2691–2699.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *ACL*. pages 665–670.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*. pages 1753–1762.

A Syntactic Neural Model for General-Purpose Code Generation

Pengcheng Yin
Language Technologies Institute
Carnegie Mellon University
pcyin@cs.cmu.edu

Graham Neubig
Language Technologies Institute
Carnegie Mellon University
gneubig@cs.cmu.edu

Abstract

We consider the problem of parsing natural language descriptions into source code written in a general-purpose programming language like Python. Existing data-driven methods treat this problem as a language generation task without considering the underlying syntax of the target programming language. Informed by previous work in semantic parsing, in this paper we propose a novel neural architecture powered by a grammar model to explicitly capture the target syntax as prior knowledge. Experiments find this an effective way to scale up to generation of complex programs from natural language descriptions, achieving state-of-the-art results that well outperform previous code generation and semantic parsing approaches.

1 Introduction

Every programmer has experienced the situation where they know what they want to do, but do not have the ability to turn it into a concrete implementation. For example, a Python programmer may want to "*sort my_list in descending order*," but not be able to come up with the proper syntax `sorted(my_list, reverse=True)` to realize his intention. To resolve this impasse, it is common for programmers to search the web in natural language (NL), find an answer, and modify it into the desired form (Brandt et al., 2009, 2010). However, this is time-consuming, and thus the software engineering literature is ripe with methods to directly generate code from NL descriptions, mostly with hand-engineered methods highly tailored to specific programming languages (Balzer, 1985; Little and Miller, 2009; Gvero and Kuncak, 2015).

In parallel, the NLP community has developed methods for data-driven semantic parsing, which attempt to map NL to structured logical forms executable by computers. These logical forms can be general-purpose meaning representations (Clark and Curran, 2007; Banarescu et al., 2013), formalisms for querying knowledge bases (Tang and Mooney, 2001; Zettlemoyer and Collins, 2005; Berant et al., 2013) and instructions for robots or personal assistants (Artzi and Zettlemoyer, 2013; Quirk et al., 2015; Misra et al., 2015), among others. While these methods have the advantage of being learnable from data, compared to the programming languages (PLs) in use by programmers, the *domain-specific* languages targeted by these works have a schema and syntax that is relatively simple.

Recently, Ling et al. (2016) have proposed a data-driven code generation method for high-level, *general-purpose* PLs like Python and Java. This work treats code generation as a sequence-to-sequence modeling problem, and introduce methods to generate words from character-level models, and copy variable names from input descriptions. However, unlike most work in semantic parsing, it does not consider the fact that code has to be well-defined programs in the target syntax.

In this work, we propose a data-driven syntax-based neural network model tailored for generation of general-purpose PLs like Python. In order to capture the strong underlying syntax of the PL, we define a model that transduces an NL statement into an Abstract Syntax Tree (AST; Fig. 1(a), § 2) for the target PL. ASTs can be deterministically generated for all well-formed programs using standard parsers provided by the PL, and thus give us a way to obtain syntax information with minimal engineering. Once we generate an AST, we can use deterministic generation tools to convert the AST into surface code. We hypothesize

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 440–450
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1041

Production Rule	Role	Explanation
Call ↦ expr[*func*] expr*[*args*] keyword*[*keywords*]	Function Call	▷ *func*: the function to be invoked ▷ *args*: arguments list ▷ *keywords*: keyword arguments list
If ↦ expr[*test*] stmt*[*body*] stmt*[*orelse*]	If Statement	▷ *test*: condition expression ▷ *body*: statements inside the If clause ▷ *orelse*: elif or else statements
For ↦ expr[*target*] expr*[*iter*] stmt*[*body*] stmt*[*orelse*]	For Loop	▷ *target*: iteration variable ▷ *iter*: enumerable to iterate over ▷ *body*: loop body ▷ *orelse*: else statements
FunctionDef ↦ identifier[*name*] arguments*[*args*] stmt*[*body*]	Function Def.	▷ *name*: function name ▷ *args*: function arguments ▷ *body*: function body

Table 1: Example production rules for common Python statements (Python Software Foundation, 2016)

that such a structured approach has two benefits.

First, we hypothesize that structure can be used to constrain our search space, ensuring generation of well-formed code. To this end, we propose a syntax-driven neural code generation model. The backbone of our approach is a *grammar model* (§ 3) which formalizes the generation story of a derivation AST into sequential application of *actions* that either apply production rules (§ 3.1), or emit terminal tokens (§ 3.2). The underlying syntax of the PL is therefore encoded in the grammar model *a priori* as the set of possible actions. Our approach frees the model from recovering the underlying grammar from limited training data, and instead enables the system to focus on learning the compositionality among existing grammar rules. Xiao et al. (2016) have noted that this imposition of structure on neural models is useful for semantic parsing, and we expect this to be even more important for general-purpose PLs where the syntax trees are larger and more complex.

Second, we hypothesize that structural information helps to model information flow within the neural network, which naturally reflects the recursive structure of PLs. To test this, we extend a standard recurrent neural network (RNN) decoder to allow for additional neural connections which reflect the recursive structure of an AST (§ 4.2). As an example, when expanding the node ⋆ in Fig. 1(a), we make use of the information from both its parent and left sibling (the dashed rectangle). This enables us to locally pass information of relevant code segments via neural network connections, resulting in more confident predictions.

Experiments (§ 5) on two Python code generation tasks show 11.7% and 9.3% absolute improvements in accuracy against the state-of-the-art system (Ling et al., 2016). Our model also gives competitive performance on a standard semantic parsing benchmark[1].

2 The Code Generation Problem

Given an NL description x, our task is to generate the code snippet c in a modern PL based on the intent of x. We attack this problem by first generating the underlying AST. We define a probabilistic grammar model of generating an AST y given x: $p(y|x)$. The best-possible AST $\hat{y}$ is then given by

$$\hat{y} = \arg\max_y p(y|x). \quad (1)$$

$\hat{y}$ is then deterministically converted to the corresponding surface code c.[2] While this paper uses examples from Python code, our method is PL-agnostic.

Before detailing our approach, we first present a brief introduction of the Python AST and its underlying grammar. The Python abstract grammar contains a set of production rules, and an AST is generated by applying several production rules composed of a head node and multiple child nodes. For instance, the first rule in Tab. 1 is used to generate the function call `sorted(·)` in Fig. 1(a). It consists of a head node of type `Call`, and three child nodes of type `expr`, `expr*` and `keyword*`, respectively. Labels of each node are noted within brackets. In an AST, non-terminal nodes sketch the general structure of the target code, while terminal nodes can be categorized into two types: *operation terminals* and *variable terminals*. Operation terminals correspond to basic arithmetic operations like `AddOp`.Variable terminal nodes store values for variables and constants of built-in data types[3]. For instance, all terminal nodes in Fig. 1(a) are variable terminal nodes.

3 Grammar Model

Before detailing our neural code generation method, we first introduce the grammar model at its core. Our probabilistic grammar model defines the generative story of a derivation AST. We fac-

[1] Implementation available at https://github.com/neulab/NL2code

[2] We use `astor` library to convert ASTs into Python code.

[3] `bool`, `float`, `int`, `str`.

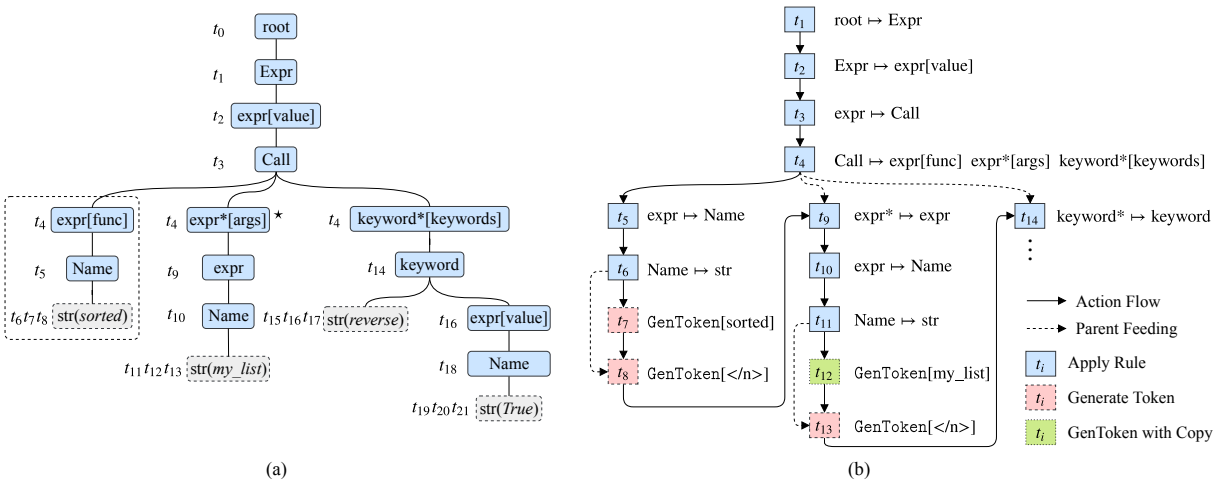


Figure 1: (a) the Abstract Syntax Tree (AST) for the given example code. Dashed nodes denote terminals. Nodes are labeled with time steps during which they are generated. (b) the action sequence (up to t_{14}) used to generate the AST in (a)

torize the generation process of an AST into sequential application of *actions* of two types:

- APPLYRULE$[r]$ applies a production rule r to the current derivation tree;
- GENTOKEN$[v]$ populates a variable terminal node by appending a terminal token v.

Fig. 1(b) shows the generation process of the target AST in Fig. 1(a). Each node in Fig. 1(b) indicates an action. Action nodes are connected by solid arrows which depict the chronological order of the action flow. The generation proceeds in depth-first, left-to-right order (dotted arrows represent parent feeding, explained in § 4.2.1).

Formally, under our grammar model, the probability of generating an AST y is factorized as:

$$p(y|x) = \prod_{t=1}^{T} p(a_t|x, a_{<t}), \qquad (2)$$

where a_t is the action taken at time step t, and $a_{<t}$ is the sequence of actions before t. We will explain how to compute the action probabilities $p(a_t|\cdot)$ in Eq. (2) in § 4. Put simply, the generation process begins from a `root` node at t_0, and proceeds by the model choosing APPLYRULE actions to generate the overall program structure from a closed set of grammar rules, then at leaves of the tree corresponding to variable terminals, the model switches to GENTOKEN actions to generate variables or constants from the open set. We describe this process in detail below.

3.1 APPLYRULE Actions

APPLYRULE actions generate program structure, expanding the current node (the *frontier node* at time step t: n_{f_t}) in a depth-first, left-to-right traversal of the tree. Given a fixed set of production rules, APPLYRULE chooses a rule r from the subset that has a head matching the type of n_{f_t}, and uses r to expand n_{f_t} by appending all child nodes specified by the selected production. As an example, in Fig. 1(b), the rule `Call` $\mapsto$ `expr...` expands the frontier node `Call` at time step t_4, and its three child nodes `expr`, `expr*` and `keyword*` are added to the derivation.

APPLYRULE actions grow the derivation AST by appending nodes. When a variable terminal node (e.g., `str`) is added to the derivation and becomes the frontier node, the grammar model then switches to GENTOKEN actions to populate the variable terminal with tokens.

Unary Closure Sometimes, generating an AST requires applying a chain of unary productions. For instance, it takes three time steps ($t_9 - t_{11}$) to generate the sub-structure `expr*` $\mapsto$ `expr` $\mapsto$ `Name` $\mapsto$ `str` in Fig. 1(a). This can be effectively reduced to one step of APPLYRULE action by taking the closure of the chain of unary productions and merging them into a single rule: `expr*` $\mapsto^*$ `str`. Unary closures reduce the number of actions needed, but would potentially increase the size of the grammar. In our experiments we tested our model both with and without unary closures (§ 5).

3.2 GENTOKEN Actions

Once we reach a frontier node n_{f_t} that corresponds to a variable type (e.g., `str`), GENTOKEN actions are used to fill this node with values. For general-purpose PLs like Python, variables and constants have values with one or multiple tokens. For in-

stance, a node that stores the name of a function (e.g., `sorted`) has a single token, while a node that denotes a string constant (e.g., `a='hello world'`) could have multiple tokens. Our model copes with both scenarios by firing GENTOKEN actions at one or more time steps. At each time step, GENTOKEN appends one terminal token to the current frontier variable node. A special `</n>` token is used to "close" the node. The grammar model then proceeds to the new frontier node.

Terminal tokens can be generated from a predefined vocabulary, or be directly copied from the input NL. This is motivated by the observation that the input description often contains out-of-vocabulary (OOV) variable names or literal values that are directly used in the target code. For instance, in our running example the variable name `my_list` can be directly copied from the the input at t_{12}. We give implementation details in § 4.2.2.

4 Estimating Action Probabilities

We estimate action probabilities in Eq. (2) using attentional neural encoder-decoder models with an information flow structured by the syntax trees.

4.1 Encoder

For an NL description x consisting of n words $\{w_i\}_{i=1}^{n}$, the encoder computes a context sensitive embedding $\mathbf{h_i}$ for each w_i using a bidirectional Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997), similar to the setting in (Bahdanau et al., 2014). See supplementary materials for detailed equations.

4.2 Decoder

The decoder uses an RNN to model the sequential generation process of an AST defined as Eq. (2). Each action step in the grammar model naturally grounds to a time step in the decoder RNN. Therefore, the action sequence in Fig. 1(b) can be interpreted as unrolling RNN time steps, with solid arrows indicating RNN connections. The RNN maintains an internal state to track the generation process (§ 4.2.1), which will then be used to compute action probabilities $p(a_t|x, a_{<t})$ (§ 4.2.2).

4.2.1 Tracking Generation States

Our implementation of the decoder resembles a vanilla LSTM, with additional neural connections (parent feeding, Fig. 1(b)) to reflect the topological structure of an AST. The decoder's internal hidden state at time step t, $\mathbf{s}_t$, is given by:

$$\mathbf{s}_t = f_{\text{LSTM}}([\mathbf{a}_{t-1} : \mathbf{c}_t : \mathbf{p}_t : \mathbf{n}_{f_t}], \mathbf{s}_{t-1}), \quad (3)$$

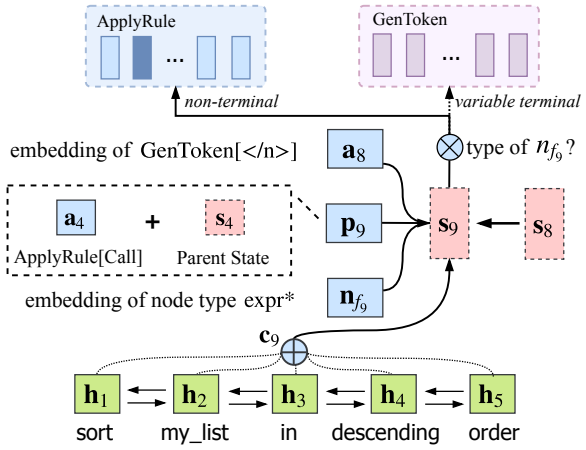


Figure 2: Illustration of a decoder time step ($t = 9$)

where $f_{\text{LSTM}}(\cdot)$ is the LSTM update function. $[:]$ denotes vector concatenation. $\mathbf{s}_t$ will then be used to compute action probabilities $p(a_t|x, a_{<t})$ in Eq. (2). Here, $\mathbf{a}_{t-1}$ is the embedding of the previous action. $\mathbf{c}_t$ is a context vector retrieved from input encodings $\{\mathbf{h}_i\}$ via soft attention. $\mathbf{p}_t$ is a vector that encodes the information of the parent action. $\mathbf{n}_{f_t}$ denotes the node type embedding of the current frontier node n_{f_t}[4]. Intuitively, feeding the decoder the information of n_{f_t} helps the model to keep track of the frontier node to expand.

Action Embedding $\mathbf{a}_t$ We maintain two action embedding matrices, $\mathbf{W}_R$ and $\mathbf{W}_G$. Each row in $\mathbf{W}_R$ ($\mathbf{W}_G$) corresponds to an embedding vector for an action APPLYRULE$[r]$ (GENTOKEN$[v]$).

Context Vector $\mathbf{c}_t$ The decoder RNN uses soft attention to retrieve a context vector $\mathbf{c}_t$ from the input encodings $\{\mathbf{h}_i\}$ pertain to the prediction of the current action. We follow Bahdanau et al. (2014) and use a Deep Neural Network (DNN) with a single hidden layer to compute attention weights.

Parent Feeding $\mathbf{p}_t$ Our decoder RNN uses additional neural connections to directly pass information from parent actions. For instance, when computing $\mathbf{s}_9$, the information from its parent action step t_4 will be used. Formally, we define the *parent action step* p_t as the time step at which the frontier node n_{f_t} is generated. As an example, for t_9, its parent action step p_9 is t_4, since n_{f_9} is the node $\star$, which is generated at t_4 by the APPLYRULE[`Call`$\mapsto \ldots$] action.

We model parent information $\mathbf{p}_t$ from two sources: (1) the hidden state of parent action $\mathbf{s}_{p_t}$, and (2) the embedding of parent action $\mathbf{a}_{p_t}$. $\mathbf{p}_t$ is the concatenation. The parent feeding schema en-

[4]We maintain an embedding for each node type.

ables the model to utilize the information of parent code segments to make more confident predictions. Similar approaches of injecting parent information were also explored in the SEQ2TREE model in Dong and Lapata (2016)[5].

4.2.2 Calculating Action Probabilities

In this section we explain how action probabilities $p(a_t|x, a_{<t})$ are computed based on $\mathbf{s}_t$.

APPLYRULE The probability of applying rule r as the current action a_t is given by a softmax[6]:

$$p(a_t = \text{APPLYRULE}[r]|x, a_{<t}) = \text{softmax}(\mathbf{W}_R \cdot g(\mathbf{s}_t))^\intercal \cdot \mathbf{e}(r) \quad (4)$$

where $g(\cdot)$ is a non-linearity $\tanh(\mathbf{W} \cdot \mathbf{s}_t + \mathbf{b})$, and $\mathbf{e}(r)$ the one-hot vector for rule r.

GENTOKEN As in § 3.2, a token v can be generated from a predefined vocabulary or copied from the input, defined as the marginal probability:

$$p(a_t = \text{GENTOKEN}[v]|x, a_{<t}) = p(\text{gen}|x, a_{<t})p(v|\text{gen}, x, a_{<t}) + p(\text{copy}|x, a_{<t})p(v|\text{copy}, x, a_{<t}).$$

The selection probabilities $p(\text{gen}|\cdot)$ and $p(\text{copy}|\cdot)$ are given by $\text{softmax}(\mathbf{W}_S \cdot \mathbf{s}_t)$. The probability of generating v from the vocabulary, $p(v|\text{gen}, x, a_{<t})$, is defined similarly as Eq. (4), except that we use the GENTOKEN embedding matrix $\mathbf{W}_G$, and we concatenate the context vector $\mathbf{c}_t$ with $\mathbf{s}_t$ as input. To model the copy probability, we follow recent advances in modeling copying mechanism in neural networks (Gu et al., 2016; Jia and Liang, 2016; Ling et al., 2016), and use a pointer network (Vinyals et al., 2015) to compute the probability of copying the i-th word from the input by attending to input representations $\{\mathbf{h}_i\}$:

$$p(w_i|\text{copy}, x, a_{<t}) = \frac{\exp(\omega(\mathbf{h}_i, \mathbf{s}_t, \mathbf{c}_t))}{\sum_{i'=1}^{n} \exp(\omega(\mathbf{h}_{i'}, \mathbf{s}_t, \mathbf{c}_t))},$$

where $\omega(\cdot)$ is a DNN with a single hidden layer. Specifically, if w_i is an OOV word (e.g., the variable name `my_list`), which is represented by a special `<unk>` token during encoding, we then directly copy the actual word w_i from the input description to the derivation.

4.3 Training and Inference

Given a dataset of pairs of NL descriptions x_i and code snippets c_i, we parse c_i into its AST y_i and decompose y_i into a sequence of oracle actions, which explains the generation story of y_i under the grammar model. The model is then optimized by maximizing the log-likelihood of the oracle action sequence. At inference time, given an NL description, we use beam search to approximate the best AST $\hat{y}$ in Eq. (1). See supplementary materials for the pseudo-code of the inference algorithm.

[5] SEQ2TREE generates tree-structured outputs by conditioning on the hidden states of parent non-terminals, while our parent feeding uses the states of parent actions.

[6] We do not show bias terms for all softmax equations.

Dataset	HS	DJANGO	IFTTT
Train	533	16,000	77,495
Development	66	1,000	5,171
Test	66	1,805	758
Avg. tokens in description	39.1	14.3	7.4
Avg. characters in code	360.3	41.1	62.2
Avg. size of AST (# nodes)	136.6	17.2	7.0
Statistics of Grammar			
w/o unary closure			
# productions	100	222	1009
# node types	61	96	828
terminal vocabulary size	1361	6733	0
Avg. # actions per example	173.4	20.3	5.0
w/ unary closure			
# productions	100	237	–
# node types	57	92	–
Avg. # actions per example	141.7	16.4	–

Table 2: Statistics of datasets and associated grammars

5 Experimental Evaluation

5.1 Datasets and Metrics

HEARTHSTONE (HS) dataset (Ling et al., 2016) is a collection of Python classes that implement cards for the card game HearthStone. Each card comes with a set of fields (e.g., name, cost, and description), which we concatenate to create the input sequence. This dataset is relatively difficult: input descriptions are short, while the target code is in complex class structures, with each AST having 137 nodes on average.

DJANGO dataset (Oda et al., 2015) is a collection of lines of code from the Django web framework, each with a manually annotated NL description. Compared with the HS dataset where card implementations are somewhat homogenous, examples in DJANGO are more diverse, spanning a wide variety of real-world use cases like string manipulation, IO operations, and exception handling.

IFTTT dataset (Quirk et al., 2015) is a domain-specific benchmark that provides an interesting side comparison. Different from HS and DJANGO which are in a general-purpose PL, programs in IFTTT are written in a domain-specific language used by the IFTTT task automation

App. Users of the App write simple instructions (e.g., `If Instagram.AnyNewPhotoByYou Then Dropbox.AddFileFromURL`) with NL descriptions (e.g., "*Autosave your Instagram photos to Dropbox*"). Each statement inside the `If` or `Then` clause consists of a channel (e.g., `Dropbox`) and a function (e.g., `AddFileFromURL`)[7]. This simple structure results in much more concise ASTs (7 nodes on average). Because all examples are created by ordinary Apps users, the dataset is highly noisy, with input NL very loosely connected to target ASTs. The authors thus provide a high-quality filtered test set, where each example is verified by at least three annotators. We use this set for evaluation. Also note IFTTT's grammar has more productions (Tab. 2), but this does not imply that its grammar is more complex. This is because for HS and DJANGO terminal tokens are generated by GENTOKEN actions, but for IFTTT, all the code is generated directly by APPLYRULE actions.

Metrics As is standard in semantic parsing, we measure **accuracy**, the fraction of correctly generated examples. However, because generating an exact match for complex code structures is non-trivial, we follow Ling et al. (2016), and use token-level **BLEU-4** with as a secondary metric, defined as the averaged BLEU scores over all examples.[8]

5.2 Setup

Preprocessing All input descriptions are tokenized using NLTK. We perform simple canonicalization for DJANGO, such as replacing quoted strings in the inputs with place holders. See supplementary materials for details. We extract unary closures whose frequency is larger than a threshold k ($k = 30$ for HS and 50 for DJANGO).

Configuration The size of all embeddings is 128, except for node type embeddings, which is 64. The dimensions of RNN states and hidden layers are 256 and 50, respectively. Since our datasets are relatively small for a data-hungry neural model, we impose strong regularization using recurrent dropouts (Gal and Ghahramani, 2016) for all recurrent networks, together with standard dropout layers added to the inputs and outputs of the decoder RNN. We validate the dropout probability from $\{0, 0.2, 0.3, 0.4\}$. For decoding, we use a beam size of 15.

	HS		DJANGO	
	ACC	BLEU	ACC	BLEU
Retrieval System†	0.0	62.5	14.7	18.6
Phrasal Statistical MT†	0.0	34.1	31.5	47.6
Hierarchical Statistical MT†	0.0	43.2	9.5	35.9
NMT	1.5	60.4	45.1	63.4
SEQ2TREE	1.5	53.4	28.9	44.6
SEQ2TREE–UNK	13.6	62.8	39.4	58.2
LPN†	4.5	65.6	62.3	77.6
Our system	16.2	**75.8**	**71.6**	**84.5**
Ablation Study				
– frontier embed.	**16.7**	**75.8**	70.7	83.8
– parent feed.	10.6	75.7	71.5	84.3
– copy terminals	3.0	65.7	32.3	61.7
+ unary closure	–		70.3	83.3
– unary closure	10.1	74.8	–	

Table 3: Results on two Python code generation tasks. †Results previously reported in Ling et al. (2016).

5.3 Results

Evaluation results for Python code generation tasks are listed in Tab. 3. Numbers for our systems are averaged over three runs. We compare primarily with two approaches: (1) Latent Predictor Network (LPN), a state-of-the-art sequence-to-sequence code generation model (Ling et al., 2016), and (2) SEQ2TREE, a neural semantic parsing model (Dong and Lapata, 2016). SEQ2TREE generates trees one node at a time, and the target grammar is not explicitly modeled a priori, but *implicitly* learned from data. We test both the original SEQ2TREE model released by the authors and our revised one (SEQ2TREE–UNK) that uses unknown word replacement to handle rare words (Luong et al., 2015). For completeness, we also compare with a strong neural machine translation (NMT) system (Neubig, 2015) using a standard encoder-decoder architecture with attention and unknown word replacement[9], and include numbers from other baselines used in Ling et al. (2016). On the HS dataset, which has relatively large ASTs, we use unary closure for our model and SEQ2TREE, and for DJANGO we do not.

[7] Like Beltagy and Quirk (2016), we strip function parameters since they are mostly specific to users.

[8] These two metrics are not ideal: accuracy only measures exact match and thus lacks the ability to give credit to semantically correct code that is different from the reference, while it is not clear whether BLEU provides an appropriate proxy for measuring semantics in the code generation task. A more intriguing metric would be directly measuring semantic/functional code equivalence, for which we present a pilot study at the end of this section (cf. Error Analysis). We leave exploring more sophisticated metrics (e.g. based on static code analysis) as future work.

[9] For NMT, we also attempted to find the best-scoring syntactically correct predictions in the size-5 beam, but this did not yield a significant improvement over the NMT results in Tab. 3.

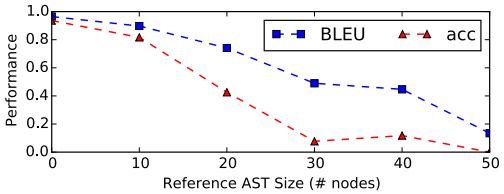


Figure 3: Performance w.r.t reference AST size on DJANGO

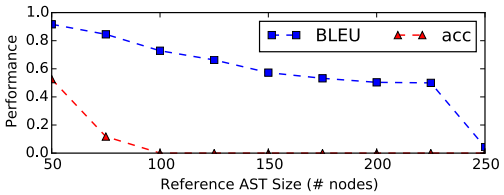


Figure 4: Performance w.r.t reference AST size on HS

	CHANNEL	FULL TREE
Classical Methods		
posclass (Quirk et al., 2015)	81.4	71.0
LR (Beltagy and Quirk, 2016)	88.8	**82.5**
Neural Network Methods		
NMT	87.7	77.7
NN (Beltagy and Quirk, 2016)	88.0	74.3
SEQ2TREE (Dong and Lapata, 2016)	89.7	78.4
Doubly-Recurrent NN (Alvarez-Melis and Jaakkola, 2017)	**90.1**	78.2
Our system	90.0	82.0
– parent feed.	89.9	81.1
– frontier embed.	**90.1**	78.7

Table 4: Results on the noise-filtered IFTTT test set of ">3 agree with gold annotations" (averaged over three runs), our model performs competitively among neural models.

System Comparison As in Tab. 3, our model registers 11.7% and 9.3% absolute improvements over LPN in accuracy on HS and DJANGO. This boost in performance strongly indicates the importance of modeling grammar in code generation. For the baselines, we find LPN outperforms NMT and SEQ2TREE in most cases. We also note that SEQ2TREE achieves a decent accuracy of 13.6% on HS, which is due to the effect of unknown word replacement, since we only achieved 1.5% without it. A closer comparison with SEQ2TREE is insightful for understanding the advantage of our syntax-driven approach, since both SEQ2TREE and our system output ASTs: (1) SEQ2TREE predicts one node each time step, and requires additional "dummy" nodes to mark the boundary of a subtree. The sheer number of nodes in target ASTs makes the prediction process error-prone. In contrast, the APPLYRULE actions of our grammar model allows for generating multiple nodes at a single time step. Empirically, we found that in HS, SEQ2TREE takes more than 300 time steps on average to generate a target AST, while our model takes only 170 steps. (2) SEQ2TREE does not directly use productions in the grammar, which possibly leads to grammatically incorrect ASTs and thus empty code outputs. We observe that the ratio of grammatically incorrect ASTs predicted by SEQ2TREE on HS and DJANGO are 21.2% and 10.9%, respectively, while our system guarantees grammaticality.

Ablation Study We also ablated our best-performing models to analyze the contribution of each component. "–frontier embed." removes the frontier node embedding $\mathbf{n}_{f_t}$ from the decoder RNN inputs (Eq. (3)). This yields worse results on DJANGO while gives slight improvements in accuracy on HS. This is probably because that the grammar of HS has fewer node types, and thus the RNN is able to keep track of n_{f_t} without depending on its embedding. Next, "–parent feed." removes the parent feeding mechanism. The accuracy drops significantly on HS, with a marginal deterioration on DJANGO. This result is interesting because it suggests that parent feeding is more important when the ASTs are larger, which will be the case when handling more complicated code generation tasks like HS. Finally, removing the pointer network ("–copy terminals") in GENTOKEN actions gives poor results, indicating that it is important to directly copy variable names and values from the input.

The results with and without unary closure demonstrate that, interestingly, it is effective on HS but not on DJANGO. We conjecture that this is because on HS it significantly reduces the number of actions from 173 to 142 (c.f., Tab. 2), with the number of productions in the grammar remaining unchanged. In contrast, DJANGO has a broader domain, and thus unary closure results in more productions in the grammar (237 for DJANGO vs. 100 for HS), increasing sparsity.

Performance by the size of AST We further investigate our model's performance w.r.t. the size of the gold-standard ASTs in Figs. 3 and 4. Not surprisingly, the performance drops when the size of the reference ASTs increases. Additionally, on the HS dataset, the BLEU score still remains at around 50 even when the size of ASTs grows to 200, indicating that our proposed syntax-driven approach is robust for long code segments.

Domain Specific Code Generation Although this is not the focus of our work, evaluation on IFTTT brings us closer to a standard semantic parsing set-

input *<name> Brawl </name> <cost> 5 </cost> <desc> Destroy all minions except one (chosen randomly) </desc> <rarity> Epic </rarity> ...*

pred.
```
class Brawl(SpellCard):
  def __init__(self):
    super().__init__('Brawl', 5, CHARACTER_CLASS.
        WARRIOR, CARD_RARITY.EPIC)
  def use(self, player, game):
    super().use(player, game)
    targets = copy.copy(game.other_player.minions)
    targets.extend(player.minions)
    for minion in targets:
      minion.die(self)
```
A

ref.
```
minions = copy.copy(player.minions)
minions.extend(game.other_player.minions)
if len(minions) > 1:
  survivor = game.random_choice(minions)
  for minion in minions:
    if minion is not survivor: minion.die(self)
```
B

input *join app_config.path and string 'locale' into a file path, substitute it for localedir.*

pred.
```
localedir = os.path.join(
    app_config.path, 'locale')
```
✓

input *self.plural is an lambda function with an argument n, which returns result of boolean expression n not equal to integer 1*

pred. `self.plural = lambda n: len(n)` ✗

ref. `self.plural = lambda n: int(n!=1)`

Table 5: Predicted examples from HS (1st) and DJANGO. Copied contents (copy probability > 0.9) are highlighted.

ting, which helps to investigate similarities and differences between generation of more complicated general-purpose code and and more limited-domain simpler code. Tab. 4 shows the results, following the evaluation protocol in (Beltagy and Quirk, 2016) for accuracies at both channel and full parse tree (channel + function) levels. Our full model performs on par with existing neural network-based methods, while outperforming other neural models in full tree accuracy (82.0%). This score is close to the best classical method (LR), which is based on a logistic regression model with rich hand-engineered features (e.g., brown clusters and paraphrase). Also note that the performance between NMT and other neural models is much closer compared with the results in Tab. 3. This suggests that general-purpose code generation is more challenging than the simpler IFTTT setting, and therefore modeling structural information is more helpful.

Case Studies We present output examples in Tab. 5. On HS, we observe that most of the time our model gives correct predictions by filling learned code templates from training data with arguments (e.g., cost) copied from input. This is in line with the findings in Ling et al. (2016). However, we do find interesting examples indicating that the model learns to generalize beyond trivial copying. For instance, the first example is one that our model predicted wrong — it generated code block A instead of the gold B (it also missed a function definition not shown here). However, we find that the block A actually conveys part of the input intent by destroying all, not some, of the minions. Since we are unable to find code block A in the training data, it is clear that the model has learned to generalize to some extent from multiple training card examples with similar semantics or structure.

The next two examples are from DJANGO. The first one shows that the model learns the usage of common API calls (e.g., `os.path.join`), and how to populate the arguments by copying from inputs. The second example illustrates the difficulty of generating code with complex nested structures like lambda functions, a scenario worth further investigation in future studies. More examples are attached in supplementary materials.

Error Analysis To understand the sources of errors and how good our evaluation metric (exact match) is, we randomly sampled and labeled 100 and 50 failed examples (with accuracy=0) from DJANGO and HS, respectively. We found that around 2% of these examples in the two datasets are actually semantically equivalent. These examples include: (1) using different parameter names when defining a function; (2) omitting (or adding) default values of parameters in function calls. While the rarity of such examples suggests that our exact match metric is reasonable, more advanced evaluation metrics based on statistical code analysis are definitely intriguing future work.

For DJANGO, we found that 30% of failed cases were due to errors where the pointer network failed to appropriately copy a variable name into the correct position. 25% were because the generated code only partially implemented the required functionality. 10% and 5% of errors were due to malformed English inputs and pre-processing errors, respectively. The remaining 30% of examples were errors stemming from multiple sources, or errors that could not be easily categorized into the above. For HS, we found that all failed card examples were due to partial implementation errors, such as the one shown in Table 5.

6 Related Work

Code Generation and Analysis Most works on code generation focus on generating code for domain specific languages (DSLs) (Kushman and

Barzilay, 2013; Raza et al., 2015; Manshadi et al., 2013), with neural network-based approaches recently explored (Liu et al., 2016; Parisotto et al., 2016; Balog et al., 2016). For general-purpose code generation, besides the general framework of Ling et al. (2016), existing methods often use language and task-specific rules and strategies (Lei et al., 2013; Raghothaman et al., 2016). A similar line is to use NL queries for code retrieval (Wei et al., 2015; Allamanis et al., 2015). The reverse task of generating NL summaries from source code has also been explored (Oda et al., 2015; Iyer et al., 2016). Finally, our work falls into the broad field of probabilistic modeling of source code (Maddison and Tarlow, 2014; Nguyen et al., 2013). Our approach of factoring an AST using probabilistic models is closely related to Allamanis et al. (2015), which uses a factorized model to measure the semantic relatedness between NL and ASTs for code retrieval, while our model tackles the more challenging generation task.

Semantic Parsing Our work is related to the general topic of semantic parsing, which aims to transform NL descriptions into executable logical forms. The target logical forms can be viewed as DSLs. The parsing process is often guided by grammatical formalisms like combinatory categorical grammars (Kwiatkowski et al., 2013; Artzi et al., 2015), dependency-based syntax (Liang et al., 2011; Pasupat and Liang, 2015) or task-specific formalisms (Clarke et al., 2010; Yih et al., 2015; Krishnamurthy et al., 2016; Mei et al., 2016). Recently, there are efforts in designing neural network-based semantic parsers (Misra and Artzi, 2016; Dong and Lapata, 2016; Neelakantan et al., 2016; Yin et al., 2016). Several approaches have be proposed to utilize grammar knowledge in a neural parser, such as augmenting the training data by generating examples guided by the grammar (Kociský et al., 2016; Jia and Liang, 2016). Liang et al. (2016) used a neural decoder which constrains the space of next valid tokens in the query language for question answering. Finally, the structured prediction approach proposed by Xiao et al. (2016) is closely related to our model in using the underlying grammar as prior knowledge to constrain the generation process of derivation trees, while our method is based on a unified grammar model which jointly captures production rule application and terminal symbol generation, and scales to general purpose code generation tasks.

7 Conclusion

This paper proposes a syntax-driven neural code generation approach that generates an abstract syntax tree by sequentially applying actions from a grammar model. Experiments on both code generation and semantic parsing tasks demonstrate the effectiveness of our proposed approach.

Acknowledgment

We are grateful to Wang Ling for his generous help with LPN and setting up the benchmark. We thank I. Beltagy for providing the IFTTT dataset. We also thank Li Dong for helping with SEQ2TREE and insightful discussions.

References

Miltiadis Allamanis, Daniel Tarlow, Andrew D. Gordon, and Yi Wei. 2015. Bimodal modelling of source code and natural language. In *Proceedings of ICML*. volume 37.

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured decoding with doubly recurrent neural networks. In *Proceedings of ICLR*.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of EMNLP*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transaction of ACL* 1(1).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2016. Deepcoder: Learning to write programs. *CoRR* abs/1611.01989.

Robert Balzer. 1985. A 15 year perspective on automatic programming. *IEEE Trans. Software Eng.* 11(11).

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of LAW-ID@ACL*.

I. Beltagy and Chris Quirk. 2016. Improved semantic parsers for if-then statements. In *Proceedings of ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*.

Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R. Klemmer. 2010. Example-centric programming: integrating web search into the development environment. In *Proceedings of CHI*.

Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proceedings of CHI*.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33(4).

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of CoNLL*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NIPS*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*.

Tihomir Gvero and Viktor Kuncak. 2015. Interactive synthesis using free-form queries. In *Proceedings of ICSE*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8).

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of ACL*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*.

Tomás Kociský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of EMNLP*.

Jayant Krishnamurthy, Oyvind Tafjord, and Aniruddha Kembhavi. 2016. Semantic parsing to probabilistic programs for situated question answering. In *Proceedings of EMNLP*.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of NAACL*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the EMNLP*.

Tao Lei, Fan Long, Regina Barzilay, and Martin C. Rinard. 2013. From natural language specifications to program input parsers. In *Proceedings of ACL*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *CoRR* abs/1611.00020.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL*.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of ACL*.

Greg Little and Robert C. Miller. 2009. Keyword programming in java. *Autom. Softw. Eng.* 16(1).

Chang Liu, Xinyun Chen, Eui Chul Richard Shin, Mingcheng Chen, and Dawn Xiaodong Song. 2016. Latent attention for if-then program synthesis. In *Proceedings of NIPS*.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*.

Chris J. Maddison and Daniel Tarlow. 2014. Structured generative models of natural source code. In *Proceedings of ICML*.

Mehdi Hafezi Manshadi, Daniel Gildea, and James F. Allen. 2013. Integrating programming by example and natural language programming. In *Proceedings of AAAI*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of AAAI*.

Dipendra K. Misra and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proceedings of EMNLP*.

Dipendra Kumar Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of ACL*.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *Proceedings of ICLR*.

Graham Neubig. 2015. lamtram: A toolkit for language and translation modeling using neural networks. http://www.github.com/neubig/lamtram.

Tung Thanh Nguyen, Anh Tuan Nguyen, Hoan Anh Nguyen, and Tien N. Nguyen. 2013. A statistical semantic language model for source code. In *Proceedings of ACM SIGSOFT*.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (T). In *Proceedings of ASE*.

Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. 2016. Neuro-symbolic program synthesis. *CoRR* abs/1611.01855.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of ACL*.

Python Software Foundation. 2016. Python abstract grammar. https://docs.python.org/2/library/ast.html.

Chris Quirk, Raymond J. Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of ACL*.

Mukund Raghothaman, Yi Wei, and Youssef Hamadi. 2016. SWIM: synthesizing what i mean: code search and idiomatic snippet synthesis. In *Proceedings of ICSE*.

Mohammad Raza, Sumit Gulwani, and Natasa Milic-Frayling. 2015. Compositional program synthesis from natural language and examples. In *Proceedings of IJCAI*.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of ECML*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of NIPS*.

Yi Wei, Nirupama Chandrasekaran, Sumit Gulwani, and Youssef Hamadi. 2015. Building bing developer assistant. Technical report. https://www.microsoft.com/en-us/research/publication/building-bing-developer-assistant/.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of ACL*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural enquirer: Learning to query tables in natural language. In *Proceedings of IJCAI*.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.

Learning bilingual word embeddings with (almost) no bilingual data

Mikel Artetxe **Gorka Labaka** **Eneko Agirre**
IXA NLP group
University of the Basque Country (UPV/EHU)
{mikel.artetxe,gorka.labaka,e.agirre}@ehu.eus

Abstract

Most methods to learn bilingual word embeddings rely on large parallel corpora, which is difficult to obtain for most language pairs. This has motivated an active research line to relax this requirement, with methods that use document-aligned corpora or bilingual dictionaries of a few thousand words instead. In this work, we further reduce the need of bilingual resources using a very simple self-learning approach that can be combined with any dictionary-based mapping technique. Our method exploits the structural similarity of embedding spaces, and works with as little bilingual evidence as a 25 word dictionary or even an automatically generated list of numerals, obtaining results comparable to those of systems that use richer resources.

1 Introduction

Multilingual word embeddings have attracted a lot of attention in recent times. In addition to having a direct application in inherently crosslingual tasks like machine translation (Zou et al., 2013) and crosslingual entity linking (Tsai and Roth, 2016), they provide an excellent mechanism for transfer learning, where a model trained in a resource-rich language is transferred to a less-resourced one, as shown with part-of-speech tagging (Zhang et al., 2016), parsing (Xiao and Guo, 2014) and document classification (Klementiev et al., 2012).

Most methods to learn these multilingual word embeddings make use of large parallel corpora (Gouws et al., 2015; Luong et al., 2015), but there have been several proposals to relax this requirement, given its scarcity in most language pairs. A possible relaxation is to use document-aligned or label-aligned comparable corpora (Søgaard et al., 2015; Vulić and Moens, 2016; Mogadala and Rettinger, 2016), but large amounts of such corpora are not always available for some language pairs.

An alternative approach that we follow here is to independently train the embeddings for each language on monolingual corpora, and then learn a linear transformation to map the embeddings from one space into the other by minimizing the distances in a bilingual dictionary, usually in the range of a few thousand entries (Mikolov et al., 2013a; Artetxe et al., 2016). However, dictionaries of that size are not readily available for many language pairs, specially those involving less-resourced languages.

In this work, we reduce the need of large bilingual dictionaries to much smaller seed dictionaries. Our method can work with as little as 25 word pairs, which are straightforward to obtain assuming some basic knowledge of the languages involved. The method can also work with trivially generated seed dictionaries of numerals (i.e. 1-1, 2-2, 3-3, 4-4...) making it possible to learn bilingual word embeddings without any real bilingual data. In either case, we obtain very competitive results, comparable to other state-of-the-art methods that make use of much richer bilingual resources.

The proposed method is an extension of existing mapping techniques, where the dictionary is used to learn the embedding mapping and the embedding mapping is used to induce a new dictionary iteratively in a self-learning fashion (see Figure 1). In spite of its simplicity, our analysis of the implicit optimization objective reveals that the method is exploiting the structural similarity of independently trained embeddings.

We analyze previous work in Section 2. Section 3 describes the self-learning framework, while Section 4 presents the experiments. Section 5 analyzes the underlying optimization objective, and Section 6 presents an error analysis.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 451–462
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1042

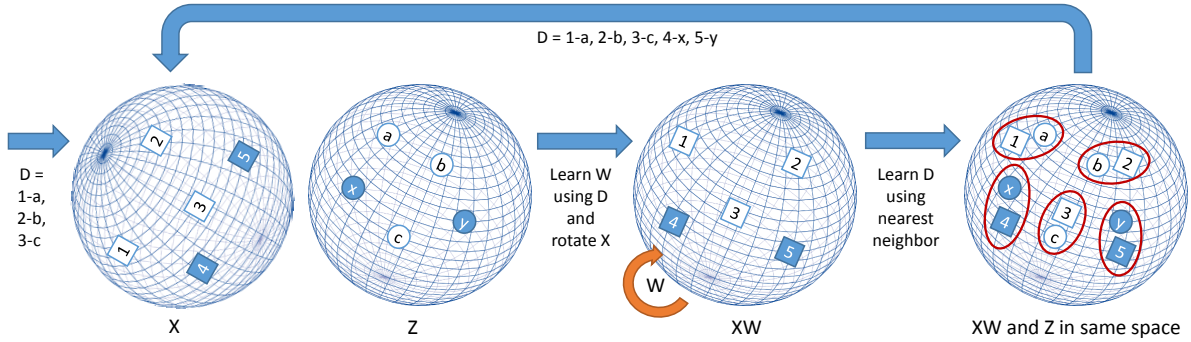


Figure 1: A general schema of the proposed self-learning framework. Previous works learn a mapping W based on the seed dictionary D, which is then used to learn the full dictionary. In our proposal we use the new dictionary to learn a new mapping, iterating until convergence.

2 Related work

We will first focus on bilingual embedding mappings, which are the basis of our proposals, and then on other unsupervised and weakly supervised methods to learn bilingual word embeddings.

2.1 Bilingual embedding mappings

Methods to induce bilingual mappings work by independently learning the embeddings in each language using monolingual corpora, and then learning a transformation from one embedding space into the other based on a bilingual dictionary.

The first of such methods is due to Mikolov et al. (2013a), who learn the linear transformation that minimizes the sum of squared Euclidean distances for the dictionary entries. The same optimization objective is used by Zhang et al. (2016), who constrain the transformation matrix to be orthogonal. Xing et al. (2015) incorporate length normalization in the training of word embeddings and maximize the cosine similarity instead, enforcing the orthogonality constraint to preserve the length normalization after the mapping. Finally, Lazaridou et al. (2015) use max-margin optimization with intruder negative sampling.

Instead of learning a single linear transformation from the source language into the target language, Faruqui and Dyer (2014) use canonical correlation analysis to map both languages to a shared vector space. Lu et al. (2015) extend this work and apply deep canonical correlation analysis to learn non-linear transformations.

Artetxe et al. (2016) propose a general framework that clarifies the relation between Mikolov et al. (2013a), Xing et al. (2015), Faruqui and Dyer (2014) and Zhang et al. (2016) as variants of the same core optimization objective, and show that a new variant is able to surpass them all. While most of the previous methods use gradient descent, Artetxe et al. (2016) propose an efficient analytical implementation for those same methods, recently extended by Smith et al. (2017) to incorporate dimensionality reduction.

A prominent application of bilingual embedding mappings, with a direct application in machine translation (Zhao et al., 2015), is bilingual lexicon extraction, which is also the main evaluation method. More specifically, the learned mapping is used to induce the translation of source language words that were missing in the original dictionary, usually by taking their nearest neighbor word in the target language according to cosine similarity, although Dinu et al. (2015) and Smith et al. (2017) propose alternative retrieval methods to address the hubness problem.

2.2 Unsupervised and weakly supervised bilingual embeddings

As mentioned before, our method works with as little as 25 word pairs, while the methods discussed previously use thousands of pairs. The only exception in this regard is the work by Zhang et al. (2016), who only use 10 word pairs with good results on transfer learning for part-of-speech tagging. Our experiments will show that, although their method captures coarse-grained relations, it fails on finer-grained tasks like bilingual lexicon induction.

Bootstrapping methods similar to ours have been previously proposed for traditional count-based vector space models (Peirsman and Padó, 2010; Vulić and Moens, 2013). However, while previous techniques incrementally build a high-

Algorithm 1 Traditional framework

Input: X (source embeddings)
Input: Z (target embeddings)
Input: D (seed dictionary)
1: $W \leftarrow$ LEARN_MAPPING(X, Z, D)
2: $D \leftarrow$ LEARN_DICTIONARY(X, Z, W)
3: EVALUATE_DICTIONARY(D)

dimensional model where each axis encodes the co-occurrences with a specific word and its equivalent in the other language, our method works with low-dimensional pre-trained word embeddings, which are more widely used nowadays.

A practical aspect for reducing the need of bilingual supervision is on the design of the seed dictionary. This is analyzed in depth by Vulić and Korhonen (2016), who propose using document-aligned corpora to extract the training dictionary. A more common approach is to rely on shared words and cognates (Peirsman and Padó, 2010; Smith et al., 2017), eliminating the need of bilingual data in practice. Our use of shared numerals exploits the same underlying idea, but relies on even less bilingual evidence and should thus generalize better to distant language pairs.

Miceli Barone (2016) and Cao et al. (2016) go one step further and attempt to learn bilingual embeddings without any bilingual evidence. The former uses adversarial autoencoders (Makhzani et al., 2016), combining an encoder that maps the source language embeddings into the target language, a decoder that reconstructs the original embeddings, and a discriminator that distinguishes mapped embeddings from real target language embeddings, whereas the latter adds a regularization term to the training of word embeddings that pushes the mean and variance of each dimension in different languages close to each other. Although promising, the reported performance in both cases is poor in comparison to other methods.

Finally, the induction of bilingual knowledge from monolingual corpora is closely related to the decipherment scenario, for which models that incorporate word embeddings have also been proposed (Dou et al., 2015). However, decipherment is only concerned with translating text from one language to another and relies on complex statistical models that are designed specifically for that purpose, while our approach is more general and learns task-independent multilingual embeddings.

Algorithm 2 Proposed self-learning framework

Input: X (source embeddings)
Input: Z (target embeddings)
Input: D (seed dictionary)
1: **repeat**
2: $W \leftarrow$ LEARN_MAPPING(X, Z, D)
3: $D \leftarrow$ LEARN_DICTIONARY(X, Z, W)
4: **until** convergence criterion
5: EVALUATE_DICTIONARY(D)

3 Proposed self-learning framework

As discussed in Section 2.1, a common evaluation task (and practical application) of bilingual embedding mappings is to induce bilingual lexicons, that is, to obtain the translation of source words that were missing in the training dictionary, which are then compared to a gold standard test dictionary for evaluation. This way, one can say that the seed (train) dictionary is used to learn a mapping, which is then used to induce a better dictionary (at least in the sense that it is larger). Algorithm 1 summarizes this framework.

Following this observation, we propose to use the output dictionary in Algorithm 1 as the input of the same system in a self-learning fashion which, assuming that the output dictionary was indeed better than the original one, should serve to learn a better mapping and, consequently, an even better dictionary the second time. The process can then be repeated iteratively to obtain a hopefully better mapping and dictionary each time until some convergence criterion is met. Algorithm 2 summarizes this alternative framework that we propose.

Our method can be combined with any embedding mapping and dictionary induction technique (see Section 2.1). However, efficiency turns out to be critical for a variety of reasons. First of all, by enclosing the learning logic in a loop, the total training time is increased by the number of iterations. Even more importantly, our framework requires to explicitly build the entire dictionary at each iteration, whereas previous work tends to induce the translation of individual words on-demand later at runtime. Moreover, from the second iteration onwards, it is this induced, full dictionary that has to be used to learn the embedding mapping, and not the considerably smaller seed dictionary as it is typically done. In the following two subsections, we respectively describe the embedding mapping method and the dictionary in-

duction method that we adopt in our work with these efficiency requirements in mind.

3.1 Embedding mapping

As discussed in Section 2.1, most previous methods to learn embedding mappings use variants of gradient descent. Among the more efficient exact alternatives, we decide to adopt the one by Artetxe et al. (2016) for its simplicity and good results as reported in their paper. We next present their method, adapting the formalization to explicitly incorporate the dictionary as required by our self-learning algorithm.

Let X and Z denote the word embedding matrices in two languages so that X_{i*} corresponds to the ith source language word embedding and Z_{j*} corresponds to the jth target language embedding. While Artetxe et al. (2016) assume these two matrices are aligned according to the dictionary, we drop this assumption and represent the dictionary explicitly as a binary matrix D, so that $D_{ij} = 1$ if the ith source language word is aligned with the jth target language word. The goal is then to find the optimal mapping matrix W^* so that the sum of squared Euclidean distances between the mapped source embeddings $X_{i*}W$ and target embeddings Z_{j*} for the dictionary entries D_{ij} is minimized:

$$W^* = \arg\min_W \sum_i \sum_j D_{ij} ||X_{i*}W - Z_{j*}||^2$$

Following Artetxe et al. (2016), we length normalize and mean center the embedding matrices X and Z in a preprocessing step, and constrain W to be an orthogonal matrix (i.e. $WW^T = W^TW = I$), which serves to enforce monolingual invariance, preventing a degradation in monolingual performance while yielding to better bilingual mappings. Under such orthogonality constraint, minimizing the squared Euclidean distance becomes equivalent to maximizing the dot product, so the above optimization objective can be reformulated as follows:

$$W^* = \arg\max_W \operatorname{Tr}\left(XWZ^TD^T\right)$$

where $\operatorname{Tr}(\cdot)$ denotes the trace operator (the sum of all the elements in the main diagonal). The optimal orthogonal solution for this problem is given by $W^* = UV^T$, where $X^TDZ = U\Sigma V^T$ is the singular value decomposition of X^TDZ. Since the dictionary matrix D is sparse, this can be efficiently computed in linear time with respect to the number of dictionary entries.

3.2 Dictionary induction

As discussed in Section 2.1, practically all previous work uses nearest neighbor retrieval for word translation induction based on embedding mappings. In nearest neighbor retrieval, each source language word is assigned the closest word in the target language. In our work, we use the dot product between the mapped source language embeddings and the target language embeddings as the similarity measure, which is roughly equivalent to cosine similarity given that we apply length normalization followed by mean centering as a preprocessing step (see Section 3.1). This way, following the notation in Section 3.1, we set $D_{ij} = 1$ if $j = \text{argmax}_k\,(X_{i*}W) \cdot Z_{k*}$ and $D_{ij} = 0$ otherwise[1].

While we find that independently computing the similarity measure between all word pairs is prohibitively slow, the computation of the entire similarity matrix XWZ^T can be easily vectorized using popular linear algebra libraries, obtaining big performance gains. However, the resulting similarity matrix is often too large to fit in memory when using large vocabularies. For that reason, instead of computing the entire similarity matrix XWZ^T in a single step, we iteratively compute submatrices of it using vectorized matrix multiplication, find their corresponding maxima each time, and then combine the results.

4 Experiments and results

In this section, we experimentally test the proposed method in bilingual lexicon induction and crosslingual word similarity. Subsection 4.1 describes the experimental settings, while Subsections 4.2 and 4.3 present the results obtained in each of the tasks. The code and resources necessary to reproduce our experiments are available at `https://github.com/artetxem/vecmap`.

4.1 Experimental settings

For easier comparison with related work, we evaluated our mappings on **bilingual lexicon induction** using the public **English-Italian** dataset by Dinu et al. (2015), which includes monolingual word embeddings in both languages together with a bilingual dictionary split in a training set and a

[1] Note that we induce the dictionary entries starting from the source language words. We experimented with other alternatives in development, with minor differences.

test set[2]. The embeddings were trained with the word2vec toolkit with CBOW and negative sampling (Mikolov et al., 2013b)[3], using a 2.8 billion word corpus for English (ukWaC + Wikipedia + BNC) and a 1.6 billion word corpus for Italian (itWaC). The training and test sets were derived from a dictionary built form Europarl word alignments and available at OPUS (Tiedemann, 2012), taking 1,500 random entries uniformly distributed in 5 frequency bins as the test set and the 5,000 most frequent of the remaining word pairs as the training set.

In addition to English-Italian, we selected two other languages from different language families with publicly available resources. We thus created analogous datasets for **English-German** and **English-Finnish**. In the case of German, the embeddings were trained on the 0.9 billion word corpus SdeWaC, which is part of the WaCky collection (Baroni et al., 2009) that was also used for English and Italian. Given that Finnish is not included in this collection, we used the 2.8 billion word Common Crawl corpus provided at WMT 2016[4] instead, which we tokenized using the Stanford Tokenizer (Manning et al., 2014). In addition to that, we created training and test sets for both pairs from their respective Europarl dictionaries from OPUS following the exact same procedure used for English-Italian, and the word embeddings were also trained using the same configuration as Dinu et al. (2015).

Given that the main focus of our work is on **small seed dictionaries**, we created random subsets of 2,500, 1,000, 500, 250, 100, 75, 50 and 25 entries from the original training dictionaries of 5,000 entries. This was done by shuffling once the training dictionaries and taking their first k entries, so it is guaranteed that each dictionary is a strict subset of the bigger dictionaries.

In addition to that, we explored using automatically generated dictionaries as a shortcut to practical unsupervised learning. For that purpose, we created **numeral dictionaries**, consisting of words matching the `[0-9]+` regular expression in both vocabularies (e.g. 1-1, 2-2, 3-3, 1992-1992 etc.). The resulting dictionary had 2772 entries for English-Italian, 2148 for English-German, and 2345 for English-Finnish. While more sophisticated approaches are possible (e.g. involving the edit distance of all words), we believe that this method is general enough that should work with practically any language pair, as Arabic numerals are often used even in languages with a different writing system (e.g. Chinese and Russian).

While bilingual lexicon induction is a standard evaluation task for seed dictionary based methods like ours, it is unsuitable for bilingual corpus based methods, as statistical word alignment already provides a reliable way to derive dictionaries from bilingual corpora and, in fact, this is how the test dictionary itself is built in our case. For that reason, we carried out some experiments in **crosslingual word similarity** as a way to test our method in a different task and allowing to compare it to systems that use richer bilingual data. There are no many crosslingual word similarity datasets, and we used the RG-65 and WordSim-353 crosslingual datasets for English-German and the WordSim-353 crosslingual dataset for English-Italian as published by Camacho-Collados et al. (2015) [5].

As for the **convergence criterion**, we decide to stop training when the improvement on the average dot product for the induced dictionary falls below a given threshold from one iteration to the next. After length normalization, the dot product ranges from -1 to 1, so we decide to set this threshold at 1e-6, which we find to be a very conservative value yet enough that training takes a reasonable amount of time. The curves in the next section confirm that this was a reasonable choice.

This convergence criterion is usually met in less than 100 iterations, each of them taking 5 minutes on a modest desktop computer (Intel Core i5-4670 CPU with 8GiB of RAM), including the induction of a dictionary of 200,000 words at each iteration.

4.2 Bilingual lexicon induction

For the experiments on bilingual lexicon induction, we compared our method with those proposed by Mikolov et al. (2013a), Xing et al. (2015), Zhang et al. (2016) and Artetxe et al. (2016), all of them implemented as part of the framework proposed by the latter. The results ob-

[2] http://clic.cimec.unitn.it/~georgiana.dinu/down/

[3] The context window was set to 5 words, the dimension of the embeddings to 300, the sub-sampling to 1e-05 and the number of negative samples to 10, and the vocabulary was restricted to the 200,000 most frequent words

[4] http://www.statmt.org/wmt16/translation-task.html

[5] http://lcl.uniroma1.it/similarity-datasets/

	English-Italian			English-German			English-Finnish		
	5,000	**25**	**num.**	**5,000**	**25**	**num.**	**5,000**	**25**	**num.**
Mikolov et al. (2013a)	34.93	0.00	0.00	35.00	0.00	0.07	25.91	0.00	0.00
Xing et al. (2015)	36.87	0.00	0.13	41.27	0.07	0.53	28.23	0.07	0.56
Zhang et al. (2016)	36.73	0.07	0.27	40.80	0.13	0.87	28.16	0.14	0.42
Artetxe et al. (2016)	39.27	0.07	0.40	**41.87**	0.13	0.73	**30.62**	0.21	0.77
Our method	**39.67**	**37.27**	**39.40**	40.87	**39.60**	**40.27**	28.72	**28.16**	**26.47**

Table 1: Accuracy (%) on bilingual lexicon induction for different seed dictionaries

tained with the 5,000 entry, 25 entry and the numerals dictionaries for all the 3 language pairs are given in Table 1.

The results for the 5,000 entry dictionaries show that our method is comparable or even better than the other systems. As another reference, the best published results using nearest-neighbor retrieval are due to Lazaridou et al. (2015), who report an accuracy of 40.20% for the full English-Italian dictionary, almost at pair with our system (39.67%).

In any case, the main focus of our work is on smaller dictionaries, and it is under this setting that our method really stands out. The 25 entry and numerals columns in Table 1 show the results for this setting, where all previous methods drop dramatically, falling below 1% accuracy in all cases. The method by Zhang et al. (2016) also obtains poor results with small dictionaries, which reinforces our hypothesis in Section 2.2 that their method can only capture coarse-grain bilingual relations for small dictionaries. In contrast, our proposed method obtains very competitive results for all dictionaries, with a difference of only 1-2 points between the full dictionary and both the 25 entry dictionary and the numerals dictionary in all three languages. Figure 2 shows the curve of the English-Italian accuracy for different seed dictionary sizes, confirming this trend.

Finally, it is worth mentioning that, even if all the three language pairs show the same general behavior, there are clear differences in their absolute accuracy numbers, which can be attributed to the linguistic proximity of the languages involved. In particular, the results for English-Finnish are about 10 points below the rest, which is explained by the fact that Finnish is a non-indoeuropean agglutinative language, making the task considerably more difficult for this language pair. In this regard, we believe that the good results with small dictionaries are a strong indication of the robustness of our method, showing that it is able to learn good bilingual mappings from very little bilingual evidence even for distant language pairs where the structural similarity of the embedding spaces is presumably weaker.

4.3 Crosslingual word similarity

In addition to the baseline systems in Section 4.2, in the crosslingual similarity experiments we also tested the method by Luong et al. (2015), which is the state-of-the-art for bilingual word embeddings based on parallel corpora (Upadhyay et al., 2016)[6]. As this method is an extension of word2vec, we used the same hyperparameters as for the monolingual embeddings when possible (see Section 4.1), and leave the default ones otherwise. We used Europarl as our parallel corpus to train this method as done by the authors, which consists of nearly 2 million parallel sentences.

As shown in the results in Table 2, our method obtains the best results in all cases, surpassing the rest of the dictionary-based methods by 1-3 points depending on the dataset. But, most importantly, it does not suffer from any significant degradation for using smaller dictionaries and, in fact, our method gets better results using the 25 entry dictionary or the numeral list as the only bilingual evidence than any of the baseline systems using much richer resources.

The relatively poor results of Luong et al. (2015) can be attributed to the fact that the dictionary based methods make use of much bigger monolingual corpora, while methods based on parallel corpora are restricted to smaller corpora. However, it is not clear how to introduce monolingual corpora on those methods. We did run some experiments with BilBOWA (Gouws et al., 2015), which supports training in monolingual corpora in addition to bilingual corpora, but obtained very poor results[7]. All in all, our experiments show

[6]We also tested English-German pre-trained embeddings from Klementiev et al. (2012) and Chandar A P et al. (2014). They both had coverage problems that made the results hard to compare, and, when considering the correlations for the word pairs in their vocabulary, their performance was poor.

[7]Upadhyay et al. (2016) report similar problems using

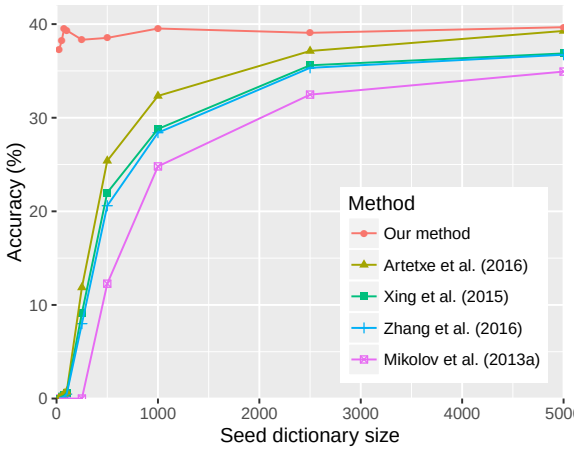


Figure 2: Accuracy on English-Italian bilingual lexicon induction for different seed dictionaries

that it is better to use large monolingual corpora in combination with very little bilingual data rather than a bilingual corpus of a standard size alone.

5 Global optimization objective

It might seem somehow surprising at first that, as seen in the previous section, our simple self-learning approach is able to learn high quality bilingual embeddings from small seed dictionaries instead of falling in degenerated solutions. In this section, we try to shed light on our approach, and give empirical evidence supporting our claim.

More concretely, we argue that, for the embedding mapping and dictionary induction methods described in Section 3, the proposed self-learning framework is implicitly solving the following global optimization problem[8]:

$$W^* = \arg\max_W \sum_i \max_j (X_{i*}W) \cdot Z_{j*}$$
$$\text{s.t.} \quad WW^T = W^TW = I$$

Contrary to the optimization objective for W in Section 3.1, the global optimization objective does not refer to any dictionary, and maximizes the similarity between each source language word and its closest target language word. Intuitively, a random solution would map source language embeddings to seemingly random locations in the target language space, and it would thus be unlikely that they have any target language word nearby, making the optimization value small. In contrast, a good solution would map source language words close to their translation equivalents in the target language space, and they would thus have their corresponding embeddings nearby, making the optimization value large. While it is certainly possible to build degenerated solutions that take high optimization values for small subsets of the vocabulary, we think that the structural similarity between independently trained embedding spaces in different languages is strong enough that optimizing this function yields to meaningful bilingual mappings when the size of the vocabulary is much larger than the dimensionality of the embeddings.

The reasoning for how the self-learning framework is optimizing this objective is as follows. At the end of each iteration, the dictionary D is updated to assign, for the current mapping W, each source language word to its closest target language word. This way, when we update W to maximize the average similarity of these dictionary entries at the beginning of the next iteration, it is guaranteed that the value of the optimization objective will improve (or at least remain the same). The reason is that the average similarity between each word and what were previously the closest words will be improved if possible, as this is what the updated W directly optimizes (see Section 3.1). In addition to that, it is also possible that, for some source words, some other target words get closer after the update. Thanks to this, our self-learning algorithm is guaranteed to converge to a local optimum of the above global objective, behaving like an alternating optimization algorithm for it.

It is interesting to note that the above reasoning is valid no matter what the the initial solution is, and, in fact, the global optimization objective does not depend on the seed dictionary nor any other

		IT	DE	
	Bi. data	**WS**	**RG**	**WS**
Luong et al. (2015)	Europarl	.331	.335	.356
Mikolov et al. (2013a)	5k dict	.627	.643	.528
Xing et al. (2015)	5k dict	.614	.700	.595
Zhang et al. (2016)	5k dict	.616	.704	.596
Artetxe et al. (2016)	5k dict	.617	.716	.597
Our method	5k dict	.624	.742	**.616**
	25 dict	.626	**.749**	.612
	num.	**.628**	.739	.604

Table 2: Spearman correlations on English-Italian and English-German crosslingual word similarity

BilBOWA.

[8]While we restrict our formal analysis to the embedding mapping and dictionary induction method that we use, the general reasoning should be valid for other choices as well.

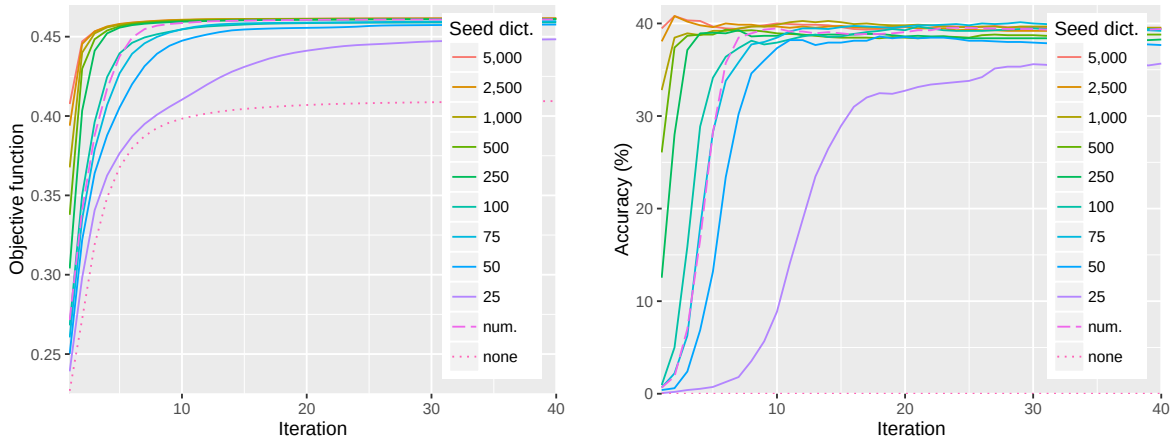


Figure 3: Learning curve on English-Italian according to the global objective function (left) and the accuracy on bilingual lexicon induction (right)

bilingual resource. For that reason, it should be possible to use a random initialization instead of a small seed dictionary. However, we empirically observe that this works poorly in practice, as our algorithm tends to get stuck in poor local optima when the initial solution is not good enough.

The general behavior of our method is reflected in Figure 3, which shows the learning curve for different seed dictionaries according to both the objective function and the accuracy on bilingual lexicon induction. As it can be seen, the objective function is improved from iteration to iteration and converges to a local optimum just as expected. At the same time, the learning curves show a strong correlation between the optimization objective and the accuracy, as it can be clearly observed that improving the former leads to an improvement of the latter, confirming our explanations. Regarding random initialization, the figure shows that the algorithm gets stuck in a poor local optimum of the objective function, which is the reason of the bad performance (0% accuracy) on bilingual lexicon induction, but the proposed optimization objective itself seems to be adequate.

Finally, we empirically observe that our algorithm learns similar mappings no matter what the seed dictionary was. We first repeated our experiments on English-Italian bilingual lexicon induction for 5 different dictionaries of 25 entries, obtaining an average accuracy of 38.15% and a standard deviation of only 0.75%. In addition to that, we observe that the overlap between the predictions made when starting with the full dictionary and the numerals dictionary is 76.00% (60.00% for the 25 entry dictionary). At the same time, 37.00% of the test cases are correctly solved by both instances, and it is only 5.07% of the test cases that one of them gets right and the other wrong (34.00% and 8.94% for the 25 entry dictionary). This suggests that our algorithm tends to converge to similar solutions even for disjoint seed dictionaries, which is in line with our view that we are implicitly optimizing an objective that is independent from the seed dictionary, yet a seed dictionary is necessary to build a good enough initial solution to avoid getting stuck in poor local optima. For that reason, it is likely that better methods to tackle this optimization problem would allow learning bilingual word embeddings without any bilingual evidence at all and, in this regard, we believe that our work opens exciting opportunities for future research.

6 Error analysis

So as to better understand the behavior of our system, we performed an error analysis of its output in English-Italian bilingual lexicon induction when starting with the 5,000 entry, the 25 entry and the numeral dictionaries in comparison with the baseline method of Artetxe et al. (2016) with the 5,000 entry dictionary. For that purpose, we took 100 random examples from the test set in the [1-5K] frequency bin, another 100 from the [5K-20K] frequency bin and 30 from the [100K-200K] frequency bin, and manually analyzed each of the errors made by all the 4 different variants.

Our analysis first reveals that, in all the cases, about a third of the translations taken as erroneous according to the gold standard are not so in real-

ity. This corresponds to both different morphological variants of the gold standard translations (e.g. *dichiarato/dichiarò*) and other valid translations that were missing in the gold standard (e.g. *climb* → *salita* instead of the gold standard *scalato*). This phenomenon is considerably more pronounced in the first frequency bins, which already have a much higher accuracy according to the gold standard.

As for the actual errors, we observe that nearly a third of them correspond to named entities for all the different variants. Interestingly, the vast majority of the proposed translations in these cases are also named entities (e.g. *Ryan* → *Jason*, *John* → *Paolo*), which are often highly related to the original ones (e.g. *Volvo* → *BMW*, *Olympus* → *Nikon*). While these are clear errors, it is understandable that these methods are unable to discriminate between named entities to this degree based solely on the distributional hypothesis, in particular when it comes to common proper names (e.g. *John*, *Andy*), and one could design alternative strategies to address this issue like taking the edit distance as an additional signal.

For the remaining errors, all systems tend to propose translations that have some degree of relationship with the correct ones, including near-synonyms (e.g. *guidelines* → *raccomandazioni*), antonyms (e.g. *sender* → *destinatario*) and words in the same semantic field (e.g. *nominalism* → *intuizionismo* / *innatismo*, which are all philosophical doctrines). However, there are also a few instances where the relationship is weak or unclear (e.g. *loch* → *giardini*, *sweep* → *serrare*). We also observe a few errors that are related to multiwords or collocations (e.g. *carrier* → *aereo*, presumably related to the multiword *air carrier* / *linea aerea*), as well as some rare word that is repeated across many translations (*Ferruzzi*), which could be attributed to the hubness problem (Dinu et al., 2015; Lazaridou et al., 2015).

All in all, our error analysis reveals that the baseline method of Artetxe et al. (2016) and the proposed algorithm tend to make the same kind of errors regardless of the seed dictionary used by the latter, which reinforces our interpretation in the previous section regarding an underlying optimization objective that is independent from any training dictionary. Moreover, it shows that the quality of the learned mappings is much better than what the raw accuracy numbers might suggest, encouraging the incorporation of these techniques in other applications.

7 Conclusions and future work

In this work, we propose a simple self-learning framework to learn bilingual word embedding mappings in combination with any embedding mapping and dictionary induction technique. Our experiments on bilingual lexicon induction and crosslingual word similarity show that our method is able to learn high quality bilingual embeddings from as little bilingual evidence as a 25 word dictionary or an automatically generated list of numerals, obtaining results that are competitive with state-of-the-art systems using much richer bilingual resources like larger dictionaries or parallel corpora. In spite of its simplicity, a more detailed analysis shows that our method is implicitly optimizing a meaningful objective function that is independent from any bilingual data which, with a better optimization method, might allow to learn bilingual word embeddings in a completely unsupervised manner.

In the future, we would like to delve deeper into this direction and fine-tune our method so it can reliably learn high quality bilingual word embeddings without any bilingual evidence at all. In addition to that, we would like to explore non-linear transformations (Lu et al., 2015) and alternative dictionary induction methods (Dinu et al., 2015; Smith et al., 2017). Finally, we would like to apply our model in the decipherment scenario (Dou et al., 2015).

Acknowledgements

We thank the anonymous reviewers for their insightful comments and Flavio Merenda for his help with the error analysis.

This research was partially supported by a Google Faculty Award, the Spanish MINECO (TUNER TIN2015-65308-C5-1-R, MUSTER PCIN-2015-226 and TADEEP TIN2015-70214-P, cofunded by EU FEDER), the Basque Government (MODELA KK-2016/00082) and the UPV/EHU (excellence research group). Mikel Artetxe enjoys a doctoral grant from the Spanish MECD.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2289–2294. https://aclweb.org/anthology/D16-1250.

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

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A framework for the construction of monolingual and cross-lingual word similarity datasets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 1–7. http://www.aclweb.org/anthology/P15-2001.

Hailong Cao, Tiejun Zhao, Shu Zhang, and Yao Meng. 2016. A distribution-based model to learn bilingual word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1818–1827. http://aclweb.org/anthology/C16-1171.

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 1853–1861. http://papers.nips.cc/paper/5270-an-autoencoder-approach-to-learning-bilingual-word-representations.pdf.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), workshop track*.

Qing Dou, Ashish Vaswani, Kevin Knight, and Chris Dyer. 2015. Unifying bayesian inference and vector space models for improved decipherment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 836–845. http://www.aclweb.org/anthology/P15-1081.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, pages 462–471. http://www.aclweb.org/anthology/E14-1049.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning*. pages 748–756.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*. The COLING 2012 Organizing Committee, Mumbai, India, pages 1459–1474. http://www.aclweb.org/anthology/C12-1089.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 270–280. http://www.aclweb.org/anthology/P15-1027.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 250–256. http://www.aclweb.org/anthology/N15-1028.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Association for Computational Linguistics, Denver, Colorado, pages 151–159. http://www.aclweb.org/anthology/W15-1521.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2016. Adversarial autoencoders. In *Proceedings of the 4rd International Conference on Learning Representations (ICLR 2016), workshop track*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. http://www.aclweb.org/anthology/P14-5010.

Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Berlin, Germany, pages 121–126. http://anthology.aclweb.org/W16-1614.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Aditya Mogadala and Achim Rettinger. 2016. Bilingual word embeddings from parallel and non-parallel corpora for cross-language text classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 692–702. http://www.aclweb.org/anthology/N16-1083.

Yves Peirsman and Sebastian Padó. 2010. Cross-lingual induction of selectional preferences with bilingual vector spaces. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 921–929. http://www.aclweb.org/anthology/N10-1135.

Samuel L. Smith, David H.P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017), conference track*.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual NLP. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1713–1722. http://www.aclweb.org/anthology/P15-1165.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual wikification using multilingual embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 589–598. http://www.aclweb.org/anthology/N16-1072.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1661–1670. http://www.aclweb.org/anthology/P16-1157.

Ivan Vulić and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 247–257. http://www.aclweb.org/anthology/P16-1024.

Ivan Vulić and Marie-Francine Moens. 2013. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1613–1624. http://www.aclweb.org/anthology/D13-1168.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research* 55(1):953–994.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 119–129. http://www.aclweb.org/anthology/W14-1613.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1006–1011. http://www.aclweb.org/anthology/N15-1104.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag – multilingual pos tagging via coarse mapping between embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies. Association for Computational Linguistics, San Diego, California, pages 1307–1317. http://www.aclweb.org/anthology/N16-1156.

Kai Zhao, Hany Hassan, and Michael Auli. 2015. Learning translation models from monolingual continuous representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1527–1536. http://www.aclweb.org/anthology/N15-1176.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1393–1398. http://www.aclweb.org/anthology/D13-1141.

Abstract Meaning Representation Parsing using LSTM Recurrent Neural Networks

William R. Foland Jr.
Department of Computer Science
University of Colorado
Boulder, CO 80309
William.Foland@colorado.edu

James H. Martin
Department of Computer Science and
Institute of Cognitive Science
University of Colorado
Boulder, CO 80309
James.Martin@colorado.edu

Abstract

We present a system which parses sentences into Abstract Meaning Representations, improving state-of-the-art results for this task by more than 5%. AMR graphs represent semantic content using linguistic properties such as semantic roles, coreference, negation, and more. The AMR parser does not rely on a syntactic preparse, or heavily engineered features, and uses five recurrent neural networks as the key architectural components for inferring AMR graphs.

1 Introduction

Semantic analysis is the process of extracting meaning from text, revealing key ideas such as "who did what to whom, when, how, and where?", and is considered to be one of the most complex tasks in natural language processing. Historically, an important consideration has been the definition of the output of the task - how can the concepts in a sentence be captured in a general, consistent and expressive manner that facilitates downstream semantic processing? Over the years many formalisms have been proposed as suitable target representations including variants of first order logic, semantic networks, and frame-based slot-filler notations. Such representations have found a place in many semantic applications but there is no clear consensus as to the best representation. However, with the rise of supervised machine learning techniques, a new requirement has come to the fore: the ability of human annotators to quickly and reliably generate semantic representations as training data.

Abstract Meaning Representation (AMR) (Banarescu et al., 2012)[1] was developed to provide a computationally useful and expressive representation that could be reliably generated by human annotators. Sentence meanings in AMR are represented in the form of graphs consisting of concepts (nodes) connected by labeled relations (edges). AMR graphs include a number of traditional NLP representations including named entities (Nadeau and Sekine, 2007), word senses (Banerjee and Pedersen, 2002), coreference relations, and predicate-argument structures (Kingsbury and Palmer, 2002; Palmer et al., 2005). More recent innovations include wikification of named entities and normalization of temporal expressions (Verhagen et al., 2010; Strötgen and Gertz, 2010). (2016) provides an insightful discussion of the relationship between AMR and other formal representations including first order logic.

The process of creating AMR's for sentences is called AMR Parsing and was first introduced in (Flanigan et al., 2014). A key factor driving the development of AMR systems has been the increasing availability of training resources in the form of corpora where each sentence is paired with a corresponding AMR representation [2]. A consistent framework for evaluating AMR parsers was defined by the Semeval-2016 Meaning Representation Parsing Task[3]. Standard training, development and test splits for the AMR Annotation Release 1 corpus are provided, as well as an additional out-of-domain test dataset, for system comparisons. [4]

Viewed as a structured prediction task, AMR parsing poses some difficult challenges not faced by other related language processing tasks including part of speech tagging, syntactic parsing or se-

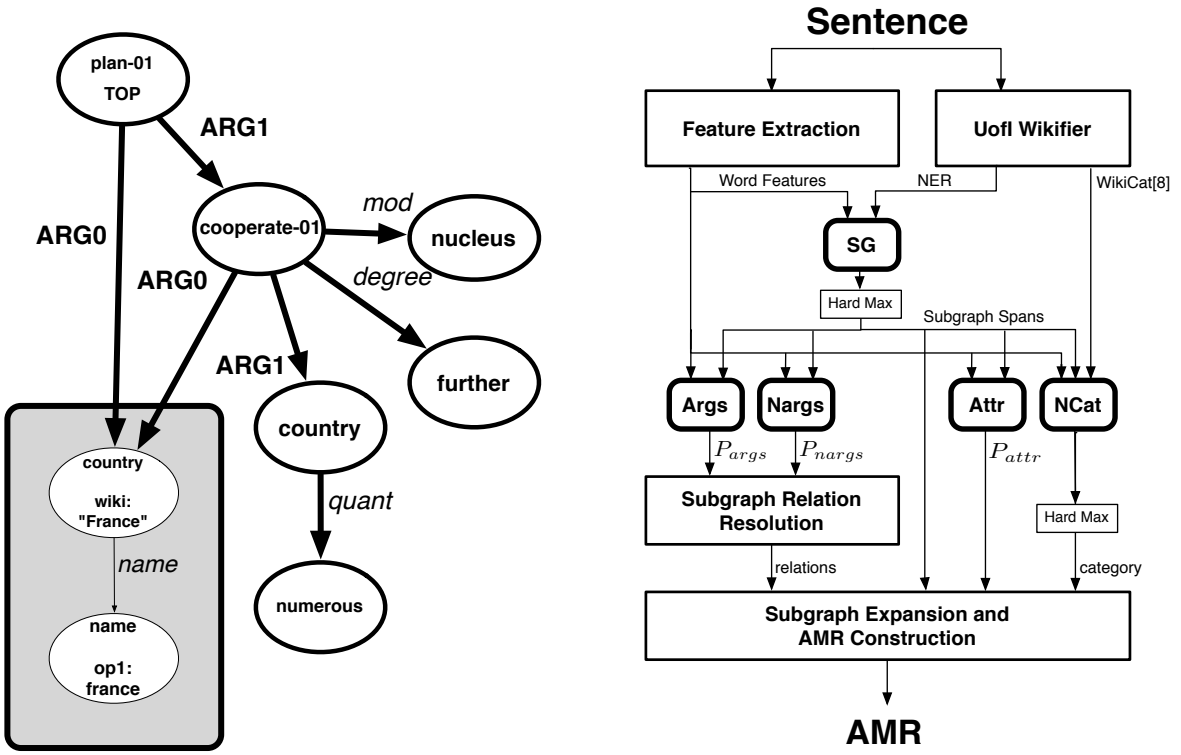
[1] http://amr.isi.edu/language.html

[2] See amr.isi.edu for information on currently available resources

[3] http://alt.qcri.org/semeval2016/task8/#

[4] Available from LDC as LDC2015E86_DEFT_Phase_2_AMR_Annotation_R1 dataset.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 463–472
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1043



(a) An AMR graphical depiction of the meaning of the sentence ***France plans further nuclear cooperation with numerous countries .*** Concepts are represented as ovals, and relations are the directed connections between them. Predicate concepts are labelled with their PropBank sense, and semantic roles are indicated by "Arg" relations. Non-Arg relations like name or mod are called "Nargs" in this paper. Note the shaded section, which shows an example of a *subgraph*, containing related concepts and relations. In the example, the subgraph represents "France" which includes the category *country* and a shortened link to the France wiki page.

(b) General Architecture for the AMR Parser, which creates an AMR based on the words in a sentence. The 5 B-LSTM networks infer structures of the AMR. For example, the SG network infers subgraphs, which are mostly single concept, like "plan-01" or "further", but can also be like the more complex shaded "France" subgraph in the example. Other B-LSTM networks are used to infer predicate argument relations (Args), other relations (Nargs), attributes like "TOP" (Attr) and name categories like "country" for France (Ncat).

Figure 1: An example Abstract Meaning Representation and the architecture of the AMR parser, which produces an AMR from a sentence.

mantic role labeling. The prediction task in these settings can be cast as per-token labeling tasks (i.e. IOB tags) or as a sequence of discrete parser actions, as in transition-based (shift-reduce) approaches to dependency parsing.

The first challenge is that AMR representations are by design abstracted away from their associated surface forms. AMR corpora pair sentences with their corresponding representations, without providing an explicit annotation, or alignment, that links the parts of the representation to their corresponding elements of the sentence. Not surprisingly, this complicates training, decoding and evaluation.

The second challenge is the fact that, as noted earlier, the AMR parsing task is an amalgam of predicate identification and classification, entity recognition, co-reference, word sense disambiguation and semantic role labeling — each of which relies on the others for successful analysis. The architecture and system presented in the following sections is largely motivated by these two challenges.

2 Related Work

2.1 AMR Parsers

Most current AMR parsers are constructed using some form of supervised machine learning that exploits existing AMR corpora. In general, these systems make use of features derived from various forms of syntactic analysis, ranging from part-of-speech tagging to more complex dependency or phrase-structure analysis. Currently, most systems fall into two classes: (1) systems that incrementally transform a dependency parse into an AMR

graph using transition-based systems (Wang et al., 2015, 2016), and (2) graph-oriented approaches that use syntactic features to score edges between all concept pairs, and then use a maximum spanning connected subgraph (MSCG) algorithm to select edges that will constitute the graph (Flanigan et al., 2014; Werling et al., 2015).

As expected, there are exceptions to these general approaches. The largely rule-based approach of (2015) converts logical forms from an existing semantic analyzer into AMR graphs. They demonstrate the ability to use their existing system to generate AMRs in German, French, Spanish and Japanese without the need for a native AMR corpus.

(2015) proposes a synchronous hyperedge replacement grammar solution, (2015) uses syntax-based machine translation techniques to create tree structures similar to AMR, while (2015) creates logical form representations of sentences and then converts these to AMR.

An exception to the use of heavily engineered features is the deep learning approach of (2016), which, following (Collobert et al., 2011), relies on word embeddings and recurrent neural networks to generate AMR graphs.

2.2 Bidirectional LSTM Neural Networks

Unlike relatively simple sequence processing tasks like part-of-speech tagging and NER, semantic analysis requires the ability to keep track of relevant information that may be arbitrarily far away from the words currently under consideration. Recurrent neural networks (RNNs) are a class of neural architecture that use a form of short-term memory in order to solve this semantic distance problem. Basic RNN systems have been enhanced with the use of special memory cell units, referred to as Long Short-Term Memory neural networks, or LSTM's (Hochreiter and Schmidhuber, 1997). Such systems can effectively process information dispersed over hundreds of words (Schmidhuber et al., 2002; Gers et al., 2001).

Bidirectional LSTMs (B-LSTM) networks are LSTMs that are connected so that both future and past sequence context can be examined. (2015), successfully used a bidirectional LSTM network for semantic role labelling. We use the LSTM cell as described in (Graves et al., 2013), configured in a B-LSTM shown in Figure 2, as the core network architecture in the system. Five B-LSTM Neural Networks comprise the parser.

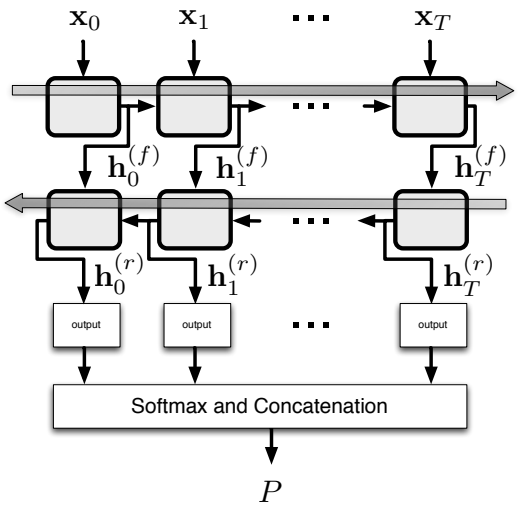


Figure 2: A general diagram of a B-LSTM network, showing the feature input vectors x_i, the forward layer (f) and the reverse layer (r). The network generates vectors of log likelihoods which are converted to probability vectors and then joined together to form an array of probabilities.

3 Parser Overview

Our parser[5] will be explained using this example sentence: ***France plans further nuclear cooperation with numerous countries .***

A graphical depiction of an AMR for this sentence is shown in Figure 1a.

Given an input sentence, the approach taken in our AMR parser is similar to (Flanigan et al., 2014) in that it consists of two subtasks: (1) discover the concepts (nodes and sub-graphs) present in the sentence, and (2) determine the relations (arcs) that connect the concepts (relations capture both traditional predicate-argument structures (ARGs), as well as additional modifier relations that capture notions including quantification, polarity, and cardinality.) Neither of these tasks is straightforward in the AMR context. Among the complications are the fact that individual words may contribute to more than one node (as in the case of France), parts of the graph may be "reentrant", participating in relations with multiple concepts, and predicate-argument and modifier relations can be introduced by arbitrary parts of the input.

At a high level, our system takes an input sentence in form of a vector of word embeddings

[5] source at https://github.com/BillFoland/daisyluAMR

and uses a series of recurrent neural networks to (1) discover the basic set of nodes and subgraphs that comprise the AMR, (2) discover the set of predicate-argument relations among those concepts, and (3) identifying any relevant modifier relations that are present.

A high level block diagram of the parser is shown in Figure 1b. The parser extracts features from the sentence which are processed by a bidirectional LSTM network (B-LSTM) to create a set of AMR subgraphs, which contain one or two concepts as well as their internal relations to each other. Features based on the sentence and these subgraphs are then processed by a pair of B-LSTM networks to compute the probabilities of relations between all subgraphs. All subgraphs are then connected using an iterative, greedy algorithm to compute a single component graph, with all subgraphs connected by relations. Separately, another two B-LSTM networks compute attribute and name categories, which are then appended to the graph. Finally, the subgraphs are expanded into the most probable AMR concept and relation primitives to create the final AMR.

4 Detailed Parser Architecture

4.1 AMR Spans, Subgraphs, and Subgraph Decoding

Mapping the words in a sentence to AMR concepts is a critical first step in the parsing process, and can influence the performance of all subsequent processing. Although the most common mapping is one word to one concept, a series of consecutive words, or **span**, can also be associated with an AMR concept. Likewise, a span of words can be mapped to a small connected **subgraph**, such as the single word span *France* which is mapped to a subgraph composed of two concepts connected by a *name* relation. (see the shaded section of Figure 1a).

Training corpora provide sentences which are annotated by humans with AMR graphs, not necessarily including a reference span to subgraph mapping. An automatic AMR **aligner** can be used to predict relationships between words and gold AMR's. We use the alignments produced by the aligner of (2014), along with the words and reference AMR graphs, to identify a **subgraph type** to associate with each span. Each word in the sentence is then associated with an IOBES subgraph type tag. We call the algorithm which defines span to subgraph mapping the **Expert Span Identifier**, and use it to train the SG Network.

A convenient development detail stems from the fact that during the AMR creation process, the identified subgraphs must be expanded into individual concepts and relations. For example, the subgraph type "Named", along with the span *France*, must be expanded to create the concepts, relations, and attributes shown in Figure 1a. A **Subgraph Expander** algorithm implements this task, which is essentially the inverse of the Expert Span Identifier. The Expert Span Identifier and Subgraph Expander were developed by cascading the two in a test configuration as shown in Figure 3a.

4.2 Features

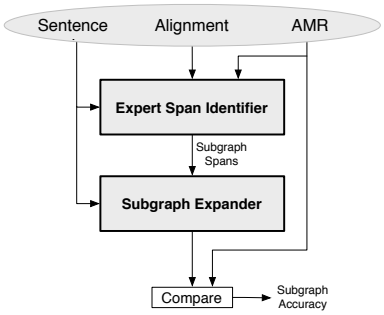
All input features for the five networks correspond to the sequence of words in the input sentence, and are presented to the networks as indices into lookup tables. With the exception of pre-trained word embeddings, these lookup tables are randomly initialized prior to training and representations are created during the training process.

4.2.1 Word Embeddings

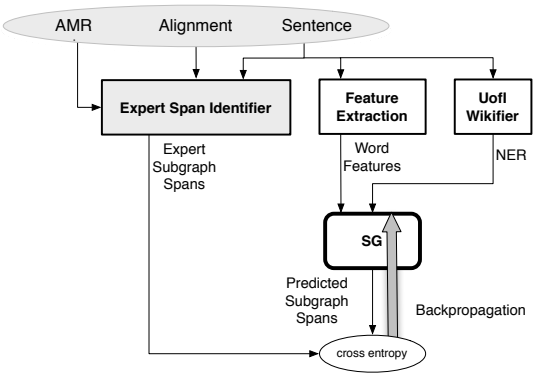
The use of distributed word representations generated from large text corpora is pervasive in modern NLP. We start with 300 dimension GloVe representations (Pennington et al., 2014) trained on the 840 billion word common crawl (Smith et al., 2013). We added two binary dimensions: one for out of vocabulary words, and one for padding, resulting in vectors with a width of 302. These embeddings are mapped from the words in the sentence, and are then trained using back propagation just like other parameters in the network.

4.2.2 Wikifier

The AMR standard was expanded to include the annotation of named entities with a canonical form, using Wikipedia as the standard (see *France* in Figure 1a). The wiki link associated with this "wikification" is expressed using the :wiki attribute, which requires some kind of global external knowledge of the Wikipedia ontology. We use the University of Illinois Wikifier (Ratinov et al., 2011; Cheng and Roth, 2013) to identify the :link directly, and use the possible categories output from the wikifier as feature inputs to the NCat Network.



(a) Expert System and Subgraph Expander Development. The alignment between the words in the sentence and elements of the AMR is provided by an automatic aligner. The expert system uses the sentence, reference AMR, and alignment to identify spans of words which are related to concepts within the AMR. These spans are also labelled with a subgraph type. A "subgraph expander" uses the words and subgraph type to expand into AMR subgraphs.



(b) SG Network Training. The SG Network uses just the words in the sentence as input, and is trained to imitate the output of the Expert System. This output defines spans of words and their subgraph types, which are the nodes of the AMR graph. Later stages of the system use this information to infer other aspects of the AMR, like relations (edges).

Figure 3: SG Model Development Details.

Named Entity Recognition can be valuable input to a parser, and state-of-the-art NER systems can be created using convolutional neural networks (Collobert et al., 2011) or LSTM (Chiu and Nichols, 2015) aided by information from gazetteers. These gazetteers are large dictionaries containing well known named entities (e.g., (Florian et al., 2003)).

Rather than add gazetteer features to our system, we make use of the NER information already calculated and provided by the Univ. of Illinois Wikifier. We then encode the classified named entities output from the wikifier as feature embeddings, which are used by the SG Network.

4.2.3 AMR Subgraph (SG) Network

The features used as input to the SG network are:

- word: 45Kx302, the word embeddings
- suffix: 430x5, embeddings based on the final two letters of each word.
- caps: 5x5, embeddings based on the capitalization pattern of the word.
- NER: 5x5, embeddings indexed by NER from the Wikifier, 'O', 'LOC', 'ORG', 'PER' or 'MISC'.

The SG Network produces probabilities for 46 BIOES tagged subgraph types, and the highest probability tag is chosen for each word, as shown for the example sentence in Table 1.

4.2.4 Predicate Argument Relations (Args) Network

The AMR concepts (nodes) are connected by relations (arcs). We found it convenient to distinguish predicate argument relations, or "Args" from other relations, which we call "Nargs". For example, see ARG0 and ARG1 relations in Figure 1a are "Args", compared with the name, degree, mod, or quant relations which are "Nargs".

The Args Network is run once for each predicate subgraph, and produces a matrix P_{args} which defines the probability (prior to the identification of any relations[6]) of a type of predicate argument relation from a predicate subgraph to any other SG identified subgraph. (For example, see ARG0 and ARG1 relations in Figure 1a.) The matrix has dimensions 5 by s, where 5 is the number of predicate arg relations identified by the network, and s is the total number of subgraphs identified by the SG Network for the sentence.

The Args features, calculated for each source predicate subgraph, are:

- Word, Suffix and Caps as in the SG network.
- SG: 46x5, indexed by the SG network identified subgraph.
- PredWords[5], 45Kx302: The word embeddings of the word and surrounding 2 words associated with the source predicate subgraph.

[6]relation probabilities change as hard decisions are made, see section 4.3

words	BIOES	Prob	kind
France	S_Named	0.995	Named subgraph
plans	S_Pred-01	0.997	plan-01
further	S_NonPred	0.931	further
nuclear	S_NonPred	0.990	nucleus
cooperation	S_Pred-01	0.986	cooperate-01
with	O	1.000	O
numerous	S_NonPred	0.982	numerous
countries	S_NonPred	0.860	country
.	O	0.999	O

Table 1: SG Network Example Output

feature	width
Word[france]	302
Suffix[ce]	5
Caps[firstUp]	5
SG[S_Named]	10
Word[further]	302
Word[nuclear]	302
Word[cooperation]	302
Word[with]	302
Word[numerous]	302
SG[S_NonPred]	10
SG[S_NonPred]	10
SG[S_Pred-01]	10
SG[O]	10
SG[S_NonPred]	10
Distance[4]	5

Table 2: Args Network Features for the word *France* while evaluating outgoing args for the word *cooperation*, associated with predicate **cooperate-01**

- PredSG[5], 46x10: The SG embedding of the word and surrounding 2 words associated with the source predicate subgraph.
- regionMark: 21x5, indexed by the distance in words between the word and the word associated with the source predicate subgraph.

Table 2 shows an example feature set for one subgraph while evaluating a predicate subgraph.

4.2.5 Non-Predicate Relations (Nargs) Network

The Nargs Network uses features similar to the Args network. It is run once for each subgraph, and produces a matrix P_{nargs} which defines the probability of a type of relation from a subgraph to any other subgraph, prior to the identification of any relations.[7] The matrix has dimensions 43 by s, where 43 is the number of non-arg relations identified by the network, and s is the total number of subgraphs identified by the SG Network for the sentence.

4.2.6 Attributes (Attr) Network

The Attr Network determines a primary attribute for each subgraph, if any.[8] This network is simplified to detect only one attribute (there could be many) per subgraph, and only computes probabilities for the two most common attributes: TOP and polarity. Note that subgraph expansion also identifies many attributes, for example the words associated with named entities, or the normalized quantity and date representations. A known shortcoming of this network is that the TOP and polarity attributes are not mutually exclusive, but noting that the cooccurrence of the two does not occur in the training data, we chose to avoid adding a separate network to allow the prediction of both attributes for a single subgraph.

4.2.7 Named Category (NCat) Network

The NCat Network uses features similar to the SG Network, along with the suggested categories (up to eight) from the Wikifier, and produces probabilities for each of 68 :instance roles, or categories, for named entities identified in the training set AMR's.

- Word, Suffix and Caps as in the SG network.
- WikiCat[8]: 108 x 5, indexed by suggested categories from the Wikifier.

4.3 Relation Resolution

The generated P_{args} and P_{nargs} for each SG identified subgraph are processed to determine the most likely relation connections, using the constraints:

[7] Degree, mod, or quant are examples of Narg relations in Figure 1a.

[8] (TOP: plan-01) and (op1: france) are attribute examples shown in Figure 1a.

1. AMR's are single component graphs without cycles.
2. AMR's are simple directed graphs, a max of one relation between any two subgraphs is allowed.
3. Outgoing predicate relations are limited to one of each kind (i.e. can't have two ARG0's)

We initialize a graph description with all the subgraphs identified by the SG network. Probabilities for all possible edges are represented in the P_{args} and P_{nargs} matrices. The Subgraphs are connected to one another by applying a greedy algorithm, which repeatedly selects the most probable edge from the P_{args} and P_{nargs} matrices and adds the edge to the graph description. After an edge is selected to be added to the graph, we adjust P_{args} and P_{nargs} based on the constraints (hard decisions change the probabilities), and repeat adding edges until all remaining edge probabilities are below a threshold. (The optimum value of this threshold, 0.55, was found by experimenting with the development data set). From then on, only the most probable edges which span graph components are chosen, until the graph contains a single component.

Expressed as a step by step procedure, we first define $p_{connect}$ as the probability threshold at which to require graph component spanning, and we repeat the following, until any two subgraphs in the graph are connected by at least one path.

1. Select the most probable outgoing relation from any of the identified subgraph probability matrices. Denote this probability as p_r.
2. If $p_r < p_{connect}$, keep selecting most probable relations until a component spanning connection is found.
3. Add the selected relation to the graph. If a cycle is created, reverse the relation direction and label.
4. Eliminate impossible relations based on the constraints and re-normalize the affected P_{args} and P_{nargs} matrices.

4.4 AMR Construction

AMR Construction converts the connected subgraph AMR into the final AMR graph form, with proper concepts, relations, and root, as follows:

1. The TOP attribute occurs exactly once in each AMR, so the subgraph with highest TOP probability produced by the Attr network is identified. The AMR graph is adjusted so that it is rooted with the most probable TOP subgraph. After graph adjustment, new cycles are sometimes created, which are removed by using *-of* relation reversal.
2. The subgraphs identified by the SG network, which were considered to be single nodes during relation resolution, are expanded to basic AMR concepts and relations to form a concept/relation AMR graph representation, using the Subgraph Expander component developed as shown in Figure 3b. When a subgraph contains two concepts, the choice of connecting to parent or child within the subgraph is made based on training data statistics of each relation type (Arg or Narg) for each subgraph type.
3. Nationalities are normalized (e.g. French to France).
4. A very basic coreference resolution is performed by merging all concepts representing "I" into a single concept. Coreference resolution was otherwise ignored due to development time constraints.

5 Experimental Setup

Semantic graph comparison can be tricky because direct graph alignment fails in the presence of just a few miscompares. A practical graph comparison program called Smatch (Cai and Knight, 2013) is used to consistently evaluate AMR parsers. The smatch python script provides an F1 evaluation metric for whole-sentence semantic graph analysis by comparing sets of triples which describe portions of the graphs, and uses a hill climbing algorithm for efficiency.

All networks, including SG, were trained using stochastic gradient descent (SGD) with a fixed learning rate. We tried sentence level log-likelihood, which trains a viterbi decoder, as a training objective, but found no improvement over word-level likelihood (cross entropy). After all LSTM and linear layers, we added dropout to minimize overfitting (Hinton et al., 2012) and batch normalization to reduce sensitivity to learning rates and initialization (Ioffe and Szegedy, 2015).

For each of the five networks, we used the LDC2015E86 training split to train parameters, and periodically interrupted training to run the dev split (forward) in order to monitor performance.

The model parameters which resulted in best dev performance were saved as the final model. The test split was used as the "in domain" data set to assess the fully assembled parser. The inferred AMR's were then evaluated using the smatch program to produce an F1 score.

An evaluation dataset was provided for Semeval 2016 task 8, which is significantly different from the LDC2015E86 split dataset. ((2016) describes the eval dataset as "quite difficult to parse, particularly due to creative approaches to word representation in the web forum portion").

6 Results

We report the statistics for smatch results of the "test" and "eval" datasets for 12 trained systems in Table 3. The top five scores for Semeval 2016 task 8, representing the previous state-of-the-art, are shown for context. With a smatch score of between 0.651 and 0.654, and a mean of 0.652, our system improves the state-of-the-art AMR parser performance by between 5.07% and 5.55%, and by a mean of 5.22%. The best performing systems for in-domain (dev and test) data correlated well with the best ones for the out-of-domain (eval) data, although the scores for the eval dataset were lower overall.

6.1 Individual Network Results

The word spans tagged by the SG network are used to determine the features for the other networks. In particular, every span identified as a predicate will trigger the system to evaluate the Args network in order to determine the probabilities of outgoing predicate ARG relations. Likewise, all spans identified as subgraphs (other than named subgraphs) will lead to a Nargs network evaluation to determine outgoing non-Arg relations. The SG network identifies predicates with 0.93 F1, named subgraphs with 0.91 F1, and all other subgraphs with 0.94 F1.

The Args network identifies ARG0 and ARG1 relations with 0.73 F1, but identification of ARG2, ARG3, and ARG4 drops down to (0.53, 0.20, and 0.43). It is difficult for the system to generalize among these relation tags because they differ significantly between predicates.

7 Conclusion and Future Work

We have shown that B-LSTM neural networks can be used as the basis for a graph based semantic parser. Our AMR parser effectively exploits the ability of B-LSTM networks to learn to selectively extract information from words separated by long distances in a sentence, and to build up higher level representations by rejecting or remembering important information during sequence processing. There are changes which could be made to eliminate all pre-processing and to further improve parser performance.

Eliminating the need for syntactic pre-parsing is valuable since a syntactic parser takes up significant time and computational resources, and errors in the generated syntax will propagate into an AMR parser. Our approach avoids both of these problems, while generating high quality results.

Wikification tasks are generally independent from parsing, but wiki links are a requirement for the latest AMR specification. Since our preferred wikifier application generates NER information, we used the generated NER tags as input to the SG network. But it would also be fairly easy to add gazetteer information to the network features in order to remove the need for NER preprocessing. Therefore, the wikification subtask is the only portion of the parser which requires any pre-processing at all. Incorporating wikification gazetteers as B-LSTM features might allow a performant, fully self contained parser to be created.

Sense disambiguation is not a very generalizable task, senses other than 01 and 02 for different predicates may differ from each other in ways which are very difficult to discern. A better approach to disambiguation is to consider predicates separately, solving for a set of coefficients for each verb found in the training set. A general set of model parameters could then be used to handle unseen examples. Likewise, high level ARGs like ARG2 and ARG3 don't generalize very well among different predicates, and ARG inference accuracy could be improved with predicate-specific network parameters for the most common cases.

The alignment between concepts and words is not a reliable, direct mapping: some concepts cannot be grounded to words, some are ambiguous, and automatic aligners tend to have high error rates relative to human aligning judgements. Improvements in the quality of the alignment in training data would improve parsing results.

System	Description	Test F1	Eval (OOD) F1
Our Parser (summary of 12 trained systems)	**mean**	**0.707**	**0.652**
	min	**0.706**	**0.651**
	max	**0.709**	**0.654**
RIGA (Barzdins and Gosko, 2016)		0.6720	0.6196
Brandeis/cemantix.org/RPI (Wang et al., 2016)		0.6670	0.6195
CU-NLP (Foland Jr and Martin, 2016)		0.6610	0.6060
ICL-HD (Brandt et al., 2016)		0.6200	0.6005
UCL+Sheffield (Goodman et al., 2016)		0.6370	0.5983

Table 3: Smatch F1 results for our parser and top 5 parsers from semeval 2016 task 8.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štepánek, Pavel Stranák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (amr) 1.0 specification. In *Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*. pages 1533–1544.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Computational linguistics and intelligent text processing*, Springer, pages 136–145.

Guntis Barzdins and Didzis Gosko. 2016. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on amr parsing accuracy. *arXiv preprint arXiv:1604.01278* .

Johan Bos. 2016. Expressive power of abstract meaning representations. *Computational Linguistics* 42(3):527–535.

Lauritz Brandt, David Grimm, Mengfei Zhou, and Yannick Versley. 2016. Icl-hd at semeval-2016 task 8: Meaning representation parsing-augmenting amr parsing with a preposition semantic role labeling neural network. *Proceedings of SemEval* pages 1160–1166.

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

X. Cheng and D. Roth. 2013. Relational inference for wikification. In *EMNLP*. http://cogcomp.cs.illinois.edu/papers/ChengRo13.pdf.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308* .

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537.

Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation .

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 168–171. https://doi.org/10.3115/1119176.1119201.

William R Foland Jr and James H Martin. 2016. Cu-nlp at semeval-2016 task 8: Amr parsing using lstm-based recurrent neural networks. *Proceedings of SemEval* pages 1197–1201.

Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. 2001. Applying lstm to time series predictable through time-window approaches. In *Artificial Neural NetworksICANN 2001*, Springer, pages 669–676.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Ucl+ sheffield at semeval-2016 task 8: Imitation learning for amr parsing with an α-bound. *Proceedings of SemEval* pages 1167–1172.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. *CoRR* abs/1303.5778. http://arxiv.org/abs/1303.5778.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580. http://arxiv.org/abs/1207.0580.

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

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR* abs/1502.03167. http://arxiv.org/abs/1502.03167.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*. Citeseer.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. *Proceedings of SemEval* pages 1063–1073.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1):3–26.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31(1):71–106.

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

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)* 12.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *EMNLP*. pages 425–429.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. *Training* 10:218–021.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*. http://cogcomp.cs.illinois.edu/papers/RRDA11.pdf.

Jürgen Schmidhuber, F Gers, and Douglas Eck. 2002. Learning nonregular languages: A comparison of simple recurrent networks and lstm. *Neural Computation* 14(9):2039–2041.

Jason R Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the common crawl. In *ACL (1)*. pages 1374–1383.

Jannik Strötgen and Michael Gertz. 2010. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 321–324.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An amr parser for english, french, german, spanish and japanese and a new amr-annotated corpus. In *Proceedings of NAACL-HLT*. pages 26–30.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, pages 57–62.

Chuan Wang, Sameer Pradhan, Nianwen Xue, Xiaoman Pan, and Heng Ji. 2016. Camr at semeval-2016 task 8: An extended transition-based amr parser. *Proceedings of SemEval* pages 1173–1178.

Chuan Wang, Nianwen Xue, Sameer Pradhan, and Sameer Pradhan. 2015. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 366–375.

Keenon Werling, Gabor Angeli, and Christopher Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. *arXiv preprint arXiv:1506.03139* .

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Deep Semantic Role Labeling: What Works and What's Next

Luheng He[†], Kenton Lee[†], Mike Lewis[‡], and Luke Zettlemoyer[†*]

[†] Paul G. Allen School of Computer Science & Engineering, Univ. of Washington, Seattle, WA
{luheng, kentonl, lsz}@cs.washington.edu
[‡] Facebook AI Research, Menlo Park, CA
mikelewis0@fb.com
[*]Allen Institute for Artificial Intelligence, Seattle, WA
lukez@allenai.org

Abstract

We introduce a new deep learning model for semantic role labeling (SRL) that significantly improves the state of the art, along with detailed analyses to reveal its strengths and limitations. We use a deep highway BiLSTM architecture with constrained decoding, while observing a number of recent best practices for initialization and regularization. Our 8-layer ensemble model achieves 83.2 F1 on the CoNLL 2005 test set and 83.4 F1 on CoNLL 2012, roughly a 10% relative error reduction over the previous state of the art. Extensive empirical analysis of these gains show that (1) deep models excel at recovering long-distance dependencies but can still make surprisingly obvious errors, and (2) that there is still room for syntactic parsers to improve these results.

1 Introduction

Semantic role labeling (SRL) systems aim to recover the predicate-argument structure of a sentence, to determine essentially "who did what to whom", "when", and "where." Recently breakthroughs involving end-to-end deep models for SRL without syntactic input (Zhou and Xu, 2015; Marcheggiani et al., 2017) seem to overturn the long-held belief that syntactic parsing is a prerequisite for this task (Punyakanok et al., 2008). In this paper, we show that this result can be pushed further using deep highway bidirectional LSTMs with constrained decoding, again significantly moving the state of the art (another 2 points on CoNLL 2005). We also present a careful empirical analysis to determine what works well and what might be done to progress even further.

Our model combines a number of best practices in the recent deep learning literature. Following Zhou and Xu (2015), we treat SRL as a BIO tagging problem and use deep bidirectional LSTMs. However, we differ by (1) simplifying the input and output layers, (2) introducing highway connections (Srivastava et al., 2015; Zhang et al., 2016), (3) using recurrent dropout (Gal and Ghahramani, 2016), (4) decoding with BIO-constraints, and (5) ensembling with a product of experts. Our model gives a 10% relative error reduction over previous state of the art on the test sets of CoNLL 2005 and 2012. We also report performance with predicted predicates to encourage future exploration of end-to-end SRL systems.

We present detailed error analyses to better understand the performance gains, including (1) design choices on architecture, initialization, and regularization that have a surprisingly large impact on model performance; (2) different types of prediction errors showing, e.g., that deep models excel at predicting long-distance dependencies but still struggle with known challenges such as PP-attachment errors and adjunct-argument distinctions; (3) the role of syntax, showing that there is significant room for improvement given oracle syntax but errors from existing automatic parsers prevent effective use in SRL.

In summary, our main contributions incluede:

- A new state-of-the-art deep network for end-to-end SRL, supported by publicly available code and models.[1]
- An in-depth error analysis indicating where the model works well and where it still struggles, including discussion of structural consistency and long-distance dependencies.
- Experiments that point toward directions for future improvements, including a detailed discussion of how and when syntactic parsers could be used to improve these results.

[1] https://github.com/luheng/deep_srl

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 473–483
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1044

2 Model

Two major factors contribute to the success of our deep SRL model: (1) applying recent advances in training deep recurrent neural networks such as highway connections (Srivastava et al., 2015) and RNN-dropouts (Gal and Ghahramani, 2016),[2] and (2) using an A* decoding algorithm (Lewis and Steedman, 2014; Lee et al., 2016) to enforce structural consistency at prediction time without adding more complexity to the training process.

Formally, our task is to predict a sequence $\boldsymbol{y}$ given a sentence-predicate pair $(\boldsymbol{w}, v)$ as input. Each $y_i \in \boldsymbol{y}$ belongs to a discrete set of BIO tags $\boldsymbol{T}$. Words outside argument spans have the tag O, and words at the beginning and inside of argument spans with role r have the tags B_r and I_r respectively. Let $n = |\boldsymbol{w}| = |\boldsymbol{y}|$ be the length of the sequence.

Predicting an SRL structure under our model involves finding the highest-scoring tag sequence over the space of all possibilities $\mathcal{Y}$:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{w}, \boldsymbol{y}) \tag{1}$$

We use a deep bidirectional LSTM (BiLSTM) to learn a locally decomposed scoring function conditioned on the input: $\sum_{t=1}^{n} \log p(y_t \mid \boldsymbol{w})$.

To incorporate additional information (e.g., structural consistency, syntactic input), we augment the scoring function with penalization terms:

$$f(\boldsymbol{w}, \boldsymbol{y}) = \sum_{t=1}^{n} \log p(y_t \mid \boldsymbol{w}) - \sum_{c \in \mathcal{C}} c(\boldsymbol{w}, y_{1:t}) \tag{2}$$

Each constraint function c applies a *non-negative* penalty given the input $\boldsymbol{w}$ and a length-t prefix $y_{1:t}$. These constraints can be hard or soft depending on whether the penalties are finite.

2.1 Deep BiLSTM Model

Our model computes the distribution over tags using stacked BiLSTMs, which we define as follows:

$$\boldsymbol{i}_{l,t} = \sigma(\mathbf{W}^l_{\mathrm{i}}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}^l_{\mathrm{i}}) \tag{3}$$
$$\boldsymbol{o}_{l,t} = \sigma(\mathbf{W}^l_{\mathrm{o}}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}^l_{\mathrm{o}}) \tag{4}$$
$$\boldsymbol{f}_{l,t} = \sigma(\mathbf{W}^l_{\mathrm{f}}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}^l_{\mathrm{f}} + 1) \tag{5}$$
$$\tilde{\boldsymbol{c}}_{l,t} = \tanh(\mathbf{W}^l_{\mathrm{c}}[\boldsymbol{h}_{l,t+\delta_l}, \boldsymbol{x}_{l,t}] + \boldsymbol{b}^l_{\mathrm{c}}) \tag{6}$$
$$\boldsymbol{c}_{l,t} = \boldsymbol{i}_{l,t} \circ \tilde{\boldsymbol{c}}_{l,t} + \boldsymbol{f}_{l,t} \circ \boldsymbol{c}_{t+\delta_l} \tag{7}$$
$$\boldsymbol{h}_{l,t} = \boldsymbol{o}_{l,t} \circ \tanh(\boldsymbol{c}_{l,t}) \tag{8}$$

[2] We thank Mingxuan Wang for suggesting highway connections with simplified inputs and outputs. Part of our model is extended from his unpublished implementation.

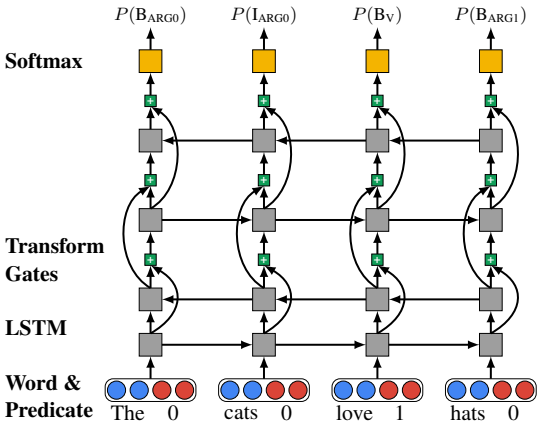


Figure 1: Highway LSTM with four layers. The curved connections represent highway connections, and the plus symbols represent transform gates that control inter-layer information flow.

where $x_{l,t}$ is the input to the LSTM at layer l and timestep t. δ_l is either 1 or -1, indicating the directionality of the LSTM at layer l.

To stack the LSTMs in an interleaving pattern, as proposed by Zhou and Xu (2015), the layer-specific inputs $x_{l,t}$ and directionality δ_l are arranged in the following manner:

$$\boldsymbol{x}_{l,t} = \begin{cases} [\mathbf{W}_{\text{emb}}(w_t), \mathbf{W}_{\text{mask}}(t = v)] & l = 1 \\ \boldsymbol{h}_{l-1,t} & l > 1 \end{cases} \tag{9}$$

$$\delta_l = \begin{cases} 1 & \text{if } l \text{ is even} \\ -1 & \text{otherwise} \end{cases} \tag{10}$$

The input vector $\boldsymbol{x}_{1,t}$ is the concatenation of token w_t's word embedding and an embedding of the binary feature $(t = v)$ indicating whether w_t word is the given predicate.

Finally, the locally normalized distribution over output tags is computed via a softmax layer:

$$p(y_t \mid \boldsymbol{x}) \propto \exp(\mathbf{W}^y_{\text{tag}} \boldsymbol{h}_{L,t} + \boldsymbol{b}_{\text{tag}}) \tag{11}$$

Highway Connections To alleviate the vanishing gradient problem when training deep BiLSTMs, we use gated highway connections (Zhang et al., 2016; Srivastava et al., 2015). We include *transform gates* $\boldsymbol{r}_t$ to control the weight of linear and non-linear transformations between layers (See Figure 1). The output $\boldsymbol{h}_{l,t}$ is changed to:

$$\boldsymbol{r}_{l,t} = \sigma(\mathbf{W}^l_{\mathrm{r}}[\boldsymbol{h}_{l,t-1}, \boldsymbol{x}_t] + \boldsymbol{b}^l_{\mathrm{r}}) \tag{12}$$
$$\boldsymbol{h}'_{l,t} = \boldsymbol{o}_{l,t} \circ \tanh(\boldsymbol{c}_{l,t}) \tag{13}$$
$$\boldsymbol{h}_{l,t} = \boldsymbol{r}_{l,t} \circ \boldsymbol{h}'_{l,t} + (1 - \boldsymbol{r}_{l,t}) \circ \mathbf{W}^l_{\mathrm{h}} \boldsymbol{x}_{l,t} \tag{14}$$

Recurrent Dropout To reduce over-fitting, we use dropout as described in Gal and Ghahramani (2016). A shared dropout mask $\boldsymbol{z}_l$ is applied to the hidden state:

$$\widetilde{\boldsymbol{h}}_{l,t} = \boldsymbol{r}_{l,t} \circ \boldsymbol{h}'_{l,t} + (1 - \boldsymbol{r}_{l,t}) \circ \mathbf{W}^l_{\mathrm{h}} \boldsymbol{x}_{l,t} \quad (15)$$

$$\boldsymbol{h}_{l,t} = \boldsymbol{z}_l \circ \widetilde{\boldsymbol{h}}_{l,t} \quad (16)$$

$\boldsymbol{z}_l$ is shared across timesteps at layer l to avoid amplifying the dropout noise along the sequence.

2.2 Constrained A* Decoding

The approach described so far does not model any dependencies between the output tags. To incorporate constraints on the output structure at decoding time, we use A* search over tag prefixes for decoding. Starting with an empty sequence, the tag sequence is built from left to right. The score for a partial sequence with length t is defined as:

$$f(\boldsymbol{w}, y_{1:t}) = \sum_{i=1}^{t} \log p(y_i \mid \boldsymbol{w}) - \sum_{c \in \mathcal{C}} c(\boldsymbol{w}, y_{1:i}) \quad (17)$$

An admissible A* heuristic can be computed efficiently by summing over the best possible tags for all timesteps after t:

$$g(\boldsymbol{w}, y_{1:t}) = \sum_{i=t+1}^{n} \max_{y_i \in \boldsymbol{T}} \log p(y_i \mid \boldsymbol{w}) \quad (18)$$

Exploration of the prefixes is determined by an agenda $\mathcal{A}$ which is sorted by $f(\boldsymbol{w}, y_{1:t}) + g(\boldsymbol{w}, y_{1:t})$. In the worst case, A* explores exponentially many prefixes, but because the distribution $p(y_t \mid \boldsymbol{w})$ learned by the BiLSTM models is very peaked, the algorithm is efficient in practice. We list some example constraints as follows:

BIO Constraints These constraints reject any sequence that does not produce valid BIO transitions, such as $\mathrm{B}_{\mathrm{ARG0}}$ followed by $\mathrm{I}_{\mathrm{ARG1}}$.

SRL Constraints Punyakanok et al. (2008); Täckström et al. (2015) described a list of SRL-specific global constraints:

- Unique core roles (U): Each core role (ARG0-ARG5, ARGA) should appear at most once for each predicate.
- Continuation roles (C): A continuation role C-X can exist only when its base role X is realized before it.
- Reference roles (R): A reference role R-X can exist only when its base role X is realized (not necessarily before R-X).

We only enforce U and C constraints, since the R constraints are more commonly violated in gold data and enforcing them results in worse performance (see discussions in Section 4.3).

Syntactic Constraints We can enforce consistency with a given parse tree by rejecting or penalizing arguments that are not constituents. In Section 4.4, we will discuss the motivation behind using syntactic constraints and experimental results using both predicted and gold syntax.

2.3 Predicate Detection

While the CoNLL 2005 shared task assumes gold predicates as input (Carreras and Màrquez, 2005), this information is not available in many downstream applications. We propose a simple model for end-to-end SRL, where the system first predicts a set of predicate words $\boldsymbol{v}$ from the input sentence $\boldsymbol{w}$. Then each predicate in $\boldsymbol{v}$ is used as an input to argument prediction. We independently predict whether each word in the sentence is a predicate, using a binary softmax over the outputs of a bidirectional LSTM trained to maximize the likelihood of the gold labels.

3 Experiments

3.1 Datasets

We measure the performance of our SRL system on two PropBank-style, span-based SRL datasets: CoNLL-2005 (Carreras and Màrquez, 2005) and CoNLL-2012 (Pradhan et al., 2013)[3]. Both datasets provide gold predicates (their index in the sentence) as part of the input. Therefore, each provided predicate corresponds to one training/test tag sequence. We follow the train-development-test split for both datasets and use the official evaluation script from the CoNLL 2005 shared task for evaluation on both datasets.

3.2 Model Setup

Our network consists of 8 BiLSTM layers (4 forward LSTMs and 4 reversed LSTMs) with 300-dimensional hidden units, and a softmax layer for predicting the output distribution.

Initialization All the weight matrices in BiLSTMs are initialized with random orthonormal matrices as described in Saxe et al. (2013).

[3]We used the version of OntoNotes downloaded at: http://cemantix.org/data/ontonotes.html.

Method	Development				WSJ Test				Brown Test				Combined
	P	R	F1	Comp.	P	R	F1	Comp.	P	R	F1	Comp.	F1
Ours (PoE)	**83.1**	**82.4**	**82.7**	**64.1**	**85.0**	**84.3**	**84.6**	**66.5**	**74.9**	**72.4**	**73.6**	**46.5**	**83.2**
Ours	81.6	81.6	81.6	62.3	83.1	83.0	83.1	64.3	72.9	71.4	72.1	44.8	81.6
Zhou	79.7	79.4	79.6	-	82.9	82.8	82.8	-	70.7	68.2	69.4	-	81.1
FitzGerald (Struct.,PoE)	81.2	76.7	78.9	55.1	82.5	78.2	80.3	57.3	74.5	70.0	72.2	41.3	-
Täckström (Struct.)	81.2	76.2	78.6	54.4	82.3	77.6	79.9	56.0	74.3	68.6	71.3	39.8	-
Toutanova (Ensemble)	-	-	78.6	58.7	81.9	78.8	80.3	60.1	-	-	68.8	40.8	-
Punyakanok (Ensemble)	80.1	74.8	77.4	50.7	82.3	76.8	79.4	53.8	73.4	62.9	67.8	32.3	77.9

Table 1: Experimental results on CoNLL 2005, in terms of precision (P), recall (R), F1 and percentage of completely correct predicates (Comp.). We report results of our best single and ensemble (PoE) model. The comparison models are Zhou and Xu (2015), FitzGerald et al. (2015), Täckström et al. (2015), Toutanova et al. (2008) and Punyakanok et al. (2008).

Method	Development				Test			
	P	R	F1	Comp.	P	R	F1	Comp.
Ours (PoE)	**83.5**	**83.2**	**83.4**	**67.5**	**83.5**	**83.3**	**83.4**	**68.5**
Ours	81.8	81.4	81.5	64.6	81.7	81.6	81.7	66.0
Zhou	-	-	81.1	-	-	-	81.3	-
FitzGerald (Struct.,PoE)	81.0	78.5	79.7	60.9	81.2	79.0	80.1	62.6
Täckström (Struct.)	80.5	77.8	79.1	60.1	80.6	78.2	79.4	61.8
Pradhan (revised)	-	-	-	-	78.5	76.6	77.5	55.8

Table 2: Experimental results on CoNLL 2012 in the same metrics as above. We compare our best single and ensemble (PoE) models against Zhou and Xu (2015), FitzGerald et al. (2015), Täckström et al. (2015) and Pradhan et al. (2013).

All tokens are lower-cased and initialized with 100-dimensional GloVe embeddings pre-trained on 6B tokens (Pennington et al., 2014) and updated during training. Tokens that are not covered by GloVe are replaced with a randomly initialized UNK embedding.

Training We use Adadelta (Zeiler, 2012) with $\epsilon = 1e^{-6}$ and $\rho = 0.95$ and mini-batches of size 80. We set RNN-dropout probability to 0.1 and clip gradients with norm larger than 1. All the models are trained for 500 epochs with early stopping based on development results. [4]

Ensembling We use a product of experts (Hinton, 2002) to combine predictions of 5 models, each trained on 80% of the training corpus and validated on the remaining 20%. For the CoNLL 2012 corpus, we split the training data from each sub-genre into 5 folds, such that the training data will have similar genre distributions.

Constrained Decoding We experimented with different types of constraints on the CoNLL 2005 and CoNLL 2012 development sets. Only the BIO hard constraints significantly improve over the ensemble model. Therefore, in our final results, we only use BIO hard constraints during decoding. [5]

3.3 Results

In Table 1 and 2, we compare our best single and ensemble model with previous work. Our ensemble (PoE) has an absolute improvement of 2.1 F1 on both CoNLL 2005 and CoNLL 2012 over the previous state of the art. Our single model also achieves more than a 0.4 improvement on both datasets. In comparison with the best reported results, our percentage of completely correct predicates improves by 5.9 points. While the continuing trend of improving SRL without syntax seems to suggest that neural end-to-end systems no longer needs parsers, our analysis in Section 4.4 will show that accurate syntactic information can improve these deep models.

[4] Training the full model on CoNLL 2005 takes about 5 days on a single Titan X Pascal GPU.

[5] A* search in this setting finds the optimal sequence for all sentences and is therefore equivalent to Viterbi decoding.

	Predicate Detection			End-to-end SRL (Single)			End-to-end SRL (PoE)			
Dataset	P	R	F1	P	R	F1	P	R	F1	Δ F1
CoNLL 2005 Dev.	97.4	97.4	97.4	80.3	80.4	80.3	81.8	81.2	81.5	-1.2
WSJ Test	94.5	98.5	96.4	80.2	82.3	81.2	82.0	83.4	82.7	-1.9
Brown Test	89.3	95.7	92.4	67.6	69.6	68.5	69.7	70.5	70.1	-3.5
CoNLL 2012 Dev.	88.7	90.6	89.7	74.9	76.2	75.5	76.5	77.8	77.2	-6.2
CoNLL 2012 Test	93.7	87.9	90.7	78.6	75.1	76.8	80.2	76.6	78.4	-5.0

Table 3: Predicate detection performance and end-to-end SRL results using predicted predicates. Δ F1 shows the absolute performance drop compared to our best ensemble model with gold predicates.

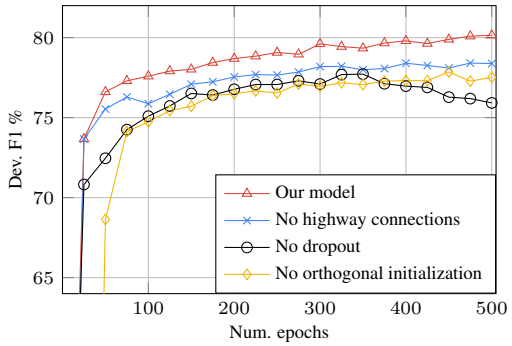


Figure 2: Smoothed learning curve of various ablations. The combination of highway layers, orthonormal parameter initialization and recurrent dropout is crucial to achieving strong performance. The numbers shown here are without constrained decoding.

3.4 Ablations

Figure 2 shows learning curves of our model ablations on the CoNLL 2005 development set. We ablate our full model by removing highway connections, RNN-dropout, and orthonormal initialization independently. Without dropout, the model overfits at around 300 epochs at 78 F1. Orthonormal parameter initialization is surprisingly important—without this, the model achieves only 65 F1 within the first 50 epochs. All 8 layer ablations suffer a loss of more than 1.7 in absolute F1 compared to the full model.

3.5 End-to-end SRL

The network for predicate detection (Section 2.3) contains 2 BiLSTM layers with 100-dimensional hidden units, and is trained for 30 epochs. For end-to-end evaluation, all arguments predicted for the false positive predicates are counted as precision loss, and all arguments for the false negative predicates are considered as recall loss.

Table 3 shows the predicate detection F1 as well as end-to-end SRL results with predicted predicates.[6] On CoNLL 2005, the predicate detector achieved over 96 F1, and the final SRL results only drop 1.2-3.5 F1 compared to using the gold predicates. However, on CoNLL 2012, the predicate detector has only about 90 F1, and the final SRL results decrease by up to 6.2 F1. This is at least in part due to the fact that CoNLL 2012 contains some nominal and copula predicates (Weischedel et al., 2013), making predicate identification a more challenging problem.

4 Analysis

To better understand our deep SRL model and its relation to previous work, we address the following questions with a suite of empirical analyses:

- What is the model good at and what kinds of mistakes does it make?
- How well do LSTMs model global structural consistency, despite conditionally independent tagging decisions?
- Is our model implicitly learning syntax, and could explicitly modeling syntax still help?

All the analysis in this section is done on the CoNLL 2005 development set with gold predicates, unless otherwise stated. We are also able to compare to previous systems whose model predictions are available online (Punyakanok et al., 2005; Pradhan et al., 2005).[7]

4.1 Error Types Breakdown

Inspired by Kummerfeld et al. (2012), we define a set of oracle transformations that fix various prediction errors *sequentially* and observe the relative improvement after each operation (see Table 4). Figure 3 shows how our work compares to the pre-

[6]The frame identification numbers reported in Pradhan et al. (2013) are not comparable, due to errors in the original release of the data, as mentioned in Täckström et al. (2015).

[7]Model predictions of CoNLL 2005 systems: http://www.cs.upc.edu/~srlconll/st05/st05.html

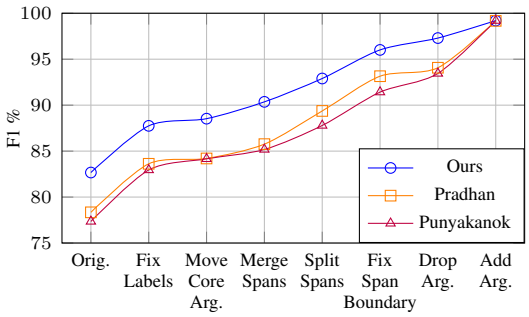


Figure 3: Performance after doing each type of oracle transformation in sequence, compared to two strong non-neural baselines. The gap is closed after the *Add Arg.* transformation, showing how our approach is gaining from predicting more arguments than traditional systems.

vious systems in terms of different types of mistakes. While our model makes a similar number of labeling errors to traditional syntax-based systems, it has far fewer missing arguments (perhaps due to parser errors making some arguments difficult to recover for syntax-based systems).

Label Confusion As shown in Table 4, our system most commonly makes labeling errors, where the predicted span is an argument but the role was incorrectly labeled. Table 5 shows a confusion matrix for the most frequent labels. The model often confuses ARG2 with AM-DIR, AM-LOC and AM-MNR. These confusions can arise due to the use of ARG2 in many verb frames to represent semantic relations such as direction or location. For example, ARG2 in the frame *move.01* is defined as *Arg2-GOL: destination*. [8] This type of argument-adjunct distinction is known to be difficult (Kingsbury et al., 2002), and it is not surprising that our neural model has many such failure cases.

Attachment Mistakes A second common source of error is reflected by the *Merge Spans* transformation (10.6%) and the *Split Spans* transformation (14.7%). These errors are closely tied to prepositional phrase (PP) attachment errors, which are also known to be some of the biggest challenges for linguistic analysis (Kummerfeld et al., 2012). Figure 4 shows the distribution of syntactic span labels involved in an attachment mistake, where 62% of the syntactic spans are prepositional phrases. For example, in *Sumitomo*

[8] Source: Unified verb index: `http://verbs.colorado.edu`.

Operation	Description	%
Fix Labels	Correct the span label if its boundary matches gold.	29.3
Move Arg.	Move a unique core argument to its correct position.	4.5
Merge Spans	Combine two predicted spans into a gold span if they are separated by at most one word.	10.6
Split Spans	Split a predicted span into two gold spans that are separated by at most one word.	14.7
Fix Boundary	Correct the boundary of a span if its label matches an overlapping gold span.	18.0
Drop Arg.	Drop a predicted argument that does not overlap with any gold span.	7.4
Add Arg.	Add a gold argument that does not overlap with any predicted span.	11.0

Table 4: Oracle transformations paired with the relative error reduction after each operation. All the operations are permitted only if they do not cause any overlapping arguments.

pred. \ gold	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	-	55	11	13	4	0	0	0	0	0
A1	78	-	46	0	0	22	11	10	25	14
A2	11	23	-	48	15	56	33	41	25	0
A3	3	2	2	-	4	0	0	0	25	14
ADV	0	0	0	4	-	0	15	29	25	36
DIR	0	0	5	4	0	-	11	2	0	0
LOC	5	9	12	0	4	0	-	10	0	14
MNR	3	0	12	26	33	0	0	-	0	21
PNC	0	3	5	4	0	11	4	2	-	0
TMP	0	8	5	0	41	11	26	6	0	-

Table 5: Confusion matrix for labeling errors, showing the percentage of predicted labels for each gold label. We only count predicted arguments that match gold span boundaries.

financed the acquisition from Sears, our model mistakenly labels the prepositional phrase *from Sears* as the ARG2 of *financed*, whereas it should instead attach to *acquisition*.

4.2 Long-range Dependencies

To analyze the model's ability to capture long-range dependencies, we compute the F1 of our model on arguments with various distances to the predicate. Figure 5 shows that performance tends to degrade, for all models, for arguments further from the predicate. Interestingly, the gap between shallow and deep models becomes much larger for the long-distance predicate-argument structures. The absolute gap between our 2 layer and 8 layer models is 3-4 F1 for arguments that are within 2 words to the predicate, and 5-6 F1 for arguments that are farther away from the predicate. Surpris-

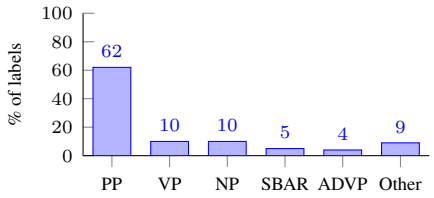


Figure 4: For cases where our model either splits a gold span into two ($Z \rightarrow XY$) or merges two gold constituents ($XY \rightarrow Z$), we show the distribution of syntactic labels for the Y span. Results show the major cause of these errors is inaccurate prepositional phrase attachment.

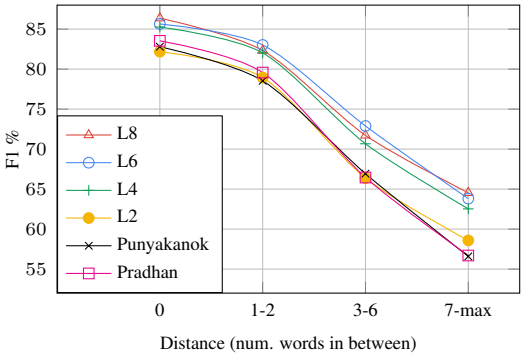


Figure 5: F1 by surface distance between predicates and arguments. Performance degrades least rapidly on long-range arguments for the deeper neural models.

ingly, the neural model performance deteriorates less severely on long-range dependencies than traditional syntax-based models.

4.3 Structural Consistency

We can quantify two types of structural consistencies: the BIO constraints and the SRL-specific constraints. Via our ablation study, we show that deeper BiLSTMs are better at enforcing these structural consistencies, although not perfectly.

BIO Violations The BIO format requires argument spans to begin with a B tag. Any I tag directly following an O tag or a tag with different label is considered a violation. Table 6 shows the number of BIO violations per token for BiLSTMs with different depths. The number of BIO violations decreases when we use a deeper model. The gap is biggest between 2-layer and 4-layer models, and diminishes after that.

It is surprising that although the deeper models generate impressively accurate token-level predictions, they still make enough BIO errors to significantly hurt performance—when these constraints are simple enough to be enforced by trivial rules. We compare the average entropy between tokens involved in BIO violations with the averaged entropy of all tokens. For the 8-layer model, the average entropy on these tokens is 30 times higher than the averaged entropy on all tokens. This suggests that BIO inconsistencies occur when there is some ambiguity. For example, if the model is unsure whether two consecutive words should belong to an ARG0 or ARG1, it might generate inconsistent BIO sequences such as B_{ARG0}, I_{ARG1} when decoding at the token level. Using BIO-constrained decoding can resolve this ambiguity and result in a structurally consistent solution.

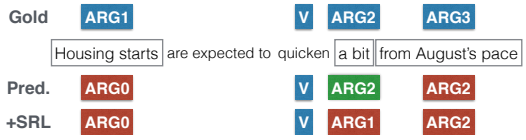


Figure 6: Example where performance is hurt by enforcing the constraint that core roles may only occur once (+SRL).

SRL Structure Violations The model predictions can also violate the SRL-specific constraints commonly used in prior work (Punyakanok et al., 2008; Täckström et al., 2015). As shown in Table 7, the model occasionally violates these SRL constraints. With our constrained decoding algorithm, it is straightforward to enforce the unique core roles (U) and continuation roles (C) constraints during decoding. The constrained decoding results are shown with the model named *L8+PoE+SRL* in Table 7.

Although the violations are eliminated, the performance does not significantly improve. This is mainly due to two factors: (1) the model often already satisfies these constraints on its own, so the number of violations to be fixed are relatively small, and (2) the gold SRL structure sometimes violates the constraints and enforcing hard constraints can hurt performance. Figure 6 shows a sentence in the CoNLL 2005 development set. Our original model produces two ARG2s for the predicate *quicken*, and this violates the SRL constraints. When the A* decoder fixes this violation, it changes the first ARG1 into ARG2 because ARG0, ARG1, ARG2 is a more frequent pattern in the training data and has higher overall score.

	Accuracy		Violations	Avg. Entropy	
Model (no BIO)	F1	Token	BIO	All	BIO
L8+PoE	81.5	91.5	0.07	0.02	0.72
L8	80.5	90.9	0.07	0.02	0.73
L6	80.1	90.3	0.06	0.02	0.72
L4	79.1	90.2	0.08	0.02	0.70
L2	74.6	88.4	0.18	0.03	0.66

Table 6: Comparison of BiLSTM models without BIO decoding. We compare F1 and token-level accuracy (Token), averaged BIO violations per token (BIO), overall model entropy (All) model entropy at tokens involved in BIO violations (BIO). Increasing the depth of the model beyond 4 does not produce more structurally consistent output, emphasizing the need for constrained decoding.

4.4 Can Syntax Still Help SRL?

The Propbank-style SRL formalism is closely tied to syntax (Bonial et al., 2010; Weischedel et al., 2013). In Table 7, we show that 98.7% of the gold SRL arguments match an unlabeled constituent in the gold syntax tree. Similar to some recent work (Zhou and Xu, 2015), our model achieves strong performance without directly modeling syntax. A natural question follows: are neural SRL models implicitly learning syntax? Table 7 shows the trend of deeper models making predictions that are more consistent with the gold syntax in terms of span boundaries. With our best model (*L8+PoE*), 94.3% of the predicted arguments spans are part of the gold parse tree. This consistency is on par with previous CoNLL 2005 systems that directly model constituency and use predicted parse trees as features (Punyakanok, 95.3% and Pradhan, 93.0%).

Constrained Decoding with Syntax The above analysis raises a further question: would improving consistency with syntax provide improvements for SRL? Our constrained decoding algorithm described in Section 2.2 enables us to inject syntax as a decoding constraint without having to retrain the model. Specifically, if the decoded sequence contains k arguments that do not match any unlabeled syntactic constituent, it will receive a penalty of kC, where C is a single parameter dictating how much the model should trust the provided syntax. In Figure 7, we compare the SRL accuracy with syntactic constraints specified by gold parse or automatic parses. When using gold syntax, the predictions improve up to 2 F1 as the penalty increases. A state-of-the-art parser (Choe and Charniak, 2016) provides smaller gains, while using the Charniak parser (Charniak, 2000) hurts performance if the model places too much trust in it. These results suggest that high-quality syntax can still make a large impact on SRL.

			SRL-Violations		
Model or Oracle	F1	Syn %	U	C	R
Gold	100.0	98.7	24	0	61
L8+PoE	82.7	94.3	37	3	68
L8	81.6	94.0	48	4	73
L6	81.4	93.7	39	3	85
L4	80.5	93.2	51	3	84
L2	77.2	91.3	96	5	72
L8+PoE+SRL	82.8	94.2	5	1	68
L8+PoE+AutoSyn	83.2	96.1	113	3	68
L8+PoE+GoldSyn	85.0	97.6	102	3	68
Punyakanok	77.4	95.3	0	0	0
Pradhan	78.3	93.0	84	3	58

Table 7: Comparison of models with different depths and decoding constraints (in addition to BIO) as well as two previous systems. We compare F1, unlabeled agreement with gold constituency (Syn%) and each type of SRL-constraint violations (**U**nique core roles, **C**ontinuation roles and **R**eference roles). Our best model produces a similar number of constraint violations to the gold annotation, explaining why deterministically enforcing these constraints is not helpful.

A known challenge for syntactic parsers is robustness on out-of-domain data. Therefore we provide experimental results in Table 8 for both CoNLL 2005 and CoNLL 2012, which consists of 8 different genres. The penalties are tuned on the two development sets separately ($C = 10000$ on CoNLL 2005 and $C = 20$ on CoNLL 2012). On the CoNLL 2005 development set, the predicted syntax gives a 0.5 F1 improvement over our best model, while on in-domain test and out-of-domain development sets, the improvement is much smaller. As expected, on CoNLL 2012, syntax improves most on the newswire (NW) domain. These improvements suggest that while decoding with hard constraints is beneficial, joint training or multi-task learning could be even more effective by leveraging full, labeled syntactic structures.

5 Related Work

Traditional approaches to semantic role labeling have used syntactic parsers to identify constituents and model long-range dependencies, and enforced

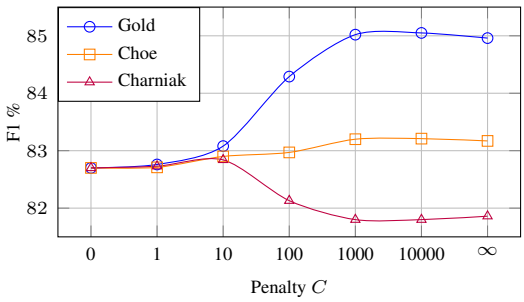


Figure 7: Performance of syntax-constrained decoding as the non-constituent penalty increases for syntax from two parsers (from Choe and Charniak (2016) and Charniak (2000)) and gold syntax. The best existing parser gives a small improvement, but the improvement from gold syntax shows that there is still potential for syntax to help SRL.

	CoNLL-05		CoNLL-2012 Dev.						
	Dev.	Test	BC	BN	NW	MZ	PT	TC	WB
L8+PoE	82.7	84.6	81.4	82.8	82.8	80.4	93.6	84.8	81.0
+AutoSyn	83.2	84.8	81.5	82.8	83.2	80.6	93.7	84.9	81.1

Table 8: F1 on CoNLL 2005, and the development set of CoNLL 2012, broken down by genres. Syntax-constrained decoding (+AutoSyn) shows bigger improvement on in-domain data (CoNLL 05 and CoNLL 2012 NW).

global consistency using integer linear programming (Punyakanok et al., 2008) or dynamic programs (Täckström et al., 2015). More recently, neural methods have been employed on top of syntactic features (FitzGerald et al., 2015; Roth and Lapata, 2016). Our experiments show that off-the-shelf neural methods have a remarkable ability to learn long-range dependencies, syntactic constituency structure, and global constraints without coding task-specific mechanisms for doing so.

An alternative line of work has attempted to reduce the dependency on syntactic input for semantic role labeling models. Collobert et al. (2011) first introduced an end-to-end neural-based approach with sequence-level training and uses a convolutional neural network to model the context window. However, their best system fell short of traditional feature-based systems. Neural methods have also been used as classifiers in transition-based SRL systems (Henderson et al., 2013; Swayamdipta et al., 2016). Most recently, several successful LSTM-based architectures have achieved state-of-the-art results in English span-based SRL (Zhou and Xu, 2015), Chinese SRL (Wang et al., 2015), and dependency-based SRL (Marcheggiani et al., 2017) with little to no syntactic input. Our techniques push results to more than 3 F1 over the best syntax-based models. However, we also show that there is potential for syntax to further improve performance.

6 Conclusion and Future Work

We presented a new deep learning model for span-based semantic role labeling with a 10% relative error reduction over the previous state of the art. Our ensemble of 8 layer BiLSTMs incorporated some of the recent best practices such as orthonormal initialization, RNN-dropout, and highway connections, and we have shown that they are crucial for getting good results with deep models.

Extensive error analysis sheds light on the strengths and limitations of our deep SRL model, with detailed comparison against shallower models and two strong non-neural systems. While our deep model is better at recovering long-distance predicate-argument relations, we still observe structural inconsistencies, which can be alleviated by constrained decoding.

Finally, we posed the question of whether deep SRL still needs syntactic supervision. Despite recent success without syntactic input, we found that our best neural model can still benefit from accurate syntactic parser output via straightforward constrained decoding. In our oracle experiment, we observed a 3 F1 improvement by leveraging gold syntax, showing the potential for high quality parsers to further improve deep SRL models.

Acknowledgments

The research was supported in part by DARPA under the DEFT program (FA8750-13-2-0019), the ARO (W911NF-16-1-0121), the NSF (IIS-1252835, IIS-1562364), gifts from Google and Tencent, and an Allen Distinguished Investigator Award. We are grateful to Mingxuan Wang for sharing his highway LSTM implementation and Sameer Pradhan for help with the CoNLL 2012 dataset. We thank Nicholas FitzGerald, Dan Garrette, Julian Michael, Hao Peng, and Swabha Swayamdipta for helpful comments, and the anonymous reviewers for valuable feedback.

References

Claire Bonial, Olga Babko-Malaya, Jinho D Choi, Jena Hwang, and Martha Palmer. 2010. Propbank annotation guidelines. *Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder* .

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 152–164.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of the First North American chapter of the Association for Computational Linguistics conference (NAACL)*. Association for Computational Linguistics, pages 132–139.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proc. of the 2016 Conference of Empirical Methods in Natural Language Processing (EMNLP)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 960–970.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 1019–1027.

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

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation* 14(8):1771–1800.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the penn treebank. In *Proceedings of the human language technology conference*. pages 252–256.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proc. of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1048–1059.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2016. Global neural ccg parsing with optimality guarantees. In *Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mike Lewis and Mark Steedman. 2014. A* ccg parsing with a supertag-factored model. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 990–1000.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. *arXiv preprint arXiv:1701.02593* .

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proc. of the 2005 Conference on Computational Natural Language Learning (CoNLL)*. pages 217–220.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proc. of the 2013 Conference on Computational Natural Language Learning (CoNLL)*. pages 143–152.

Vasin Punyakanok, Peter Koomen, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proc. of the 2005 Conference on Computational Natural Language Learning (CoNLL)*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics* 34(2):257–287.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* .

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in neural information processing systems*. pages 2377–2385.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. In *Proc. of the 2016 Conference on Computational Natural Language Learning (CoNLL)*. page 187.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics* 3:29–41.

Kristina Toutanova, Aria Haghighi, and Christopher D Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics* 34(2):161–191.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1626–1631.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. 2016. Highway long short-term memory rnns for distant speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 5755–5759.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access

Bhuwan Dhingra[⋆] **Lihong Li**[†] **Xiujun Li**[†] **Jianfeng Gao**[†]
Yun-Nung Chen[‡] **Faisal Ahmed**[†] **Li Deng**[†]
[⋆]Carnegie Mellon University, Pittsburgh, PA, USA
[†]Microsoft Research, Redmond, WA, USA
[‡]National Taiwan University, Taipei, Taiwan
[⋆]bdhingra@andrew.cmu.edu [†]{lihongli,xiul,jfgao}@microsoft.com [‡]y.v.chen@ieee.org

Abstract

This paper proposes *KB-InfoBot*[1] — a multi-turn dialogue agent which helps users search Knowledge Bases (KBs) without composing complicated queries. Such goal-oriented dialogue agents typically need to interact with an external database to access real-world knowledge. Previous systems achieved this by issuing a symbolic query to the KB to retrieve entries based on their attributes. However, such symbolic operations break the differentiability of the system and prevent end-to-end training of neural dialogue agents. In this paper, we address this limitation by replacing symbolic queries with an induced "soft" posterior distribution over the KB that indicates which entities the user is interested in. Integrating the soft retrieval process with a reinforcement learner leads to higher task success rate and reward in both simulations and against real users. We also present a fully neural end-to-end agent, trained entirely from user feedback, and discuss its application towards personalized dialogue agents.

1 Introduction

The design of intelligent assistants which interact with users in natural language ranks high on the agenda of current NLP research. With an increasing focus on the use of statistical and machine learning based approaches (Young et al., 2013), the last few years have seen some truly remarkable conversational agents appear on the market (e.g. Apple Siri, Microsoft Cortana, Google Allo). These agents can perform simple tasks, answer factual questions, and sometimes also aimlessly chit-chat with the user, but they still lag far behind a human assistant in terms of both the variety and complexity of tasks they can perform. In particular, they lack the ability to learn from interactions with a user in order to improve and adapt with time. Recently, Reinforcement Learning (RL) has been explored to leverage user interactions to adapt various dialogue agents designed, respectively, for task completion (Gašić et al., 2013), information access (Wen et al., 2016b), and chitchat (Li et al., 2016a).

We focus on KB-InfoBots, a particular type of dialogue agent that helps users navigate a Knowledge Base (KB) in search of an entity, as illustrated by the example in Figure 1. Such agents must necessarily query databases in order to retrieve the requested information. This is usually done by performing semantic parsing on the input to construct a symbolic query representing the beliefs of the agent about the user goal, such as Wen et al. (2016b), Williams and Zweig (2016), and Li et al. (2017)'s work. We call such an operation a *Hard-KB* lookup. While natural, this approach has two drawbacks: (1) the retrieved results do not carry any information about uncertainty in semantic parsing, and (2) the retrieval operation is non differentiable, and hence the parser and dialog policy are trained separately. This makes online end-to-end learning from user feedback difficult once the system is deployed.

In this work, we propose a probabilistic framework for computing the posterior distribution of the user target over a knowledge base, which we term a *Soft-KB* lookup. This distribution is constructed from the agent's belief about the attributes of the entity being searched for. The dialogue policy network, which decides the next system action, receives as input this full distribution instead of a handful of retrieved results. We show in our ex-

[1]The source code is available at: https://github.com/MiuLab/KB-InfoBot

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 484–495
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1045

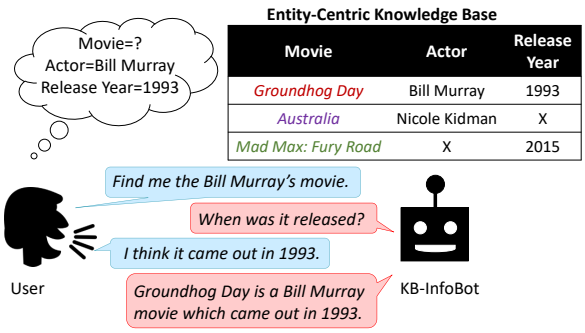


Figure 1: An interaction between a user looking for a movie and the KB-InfoBot. An entity-centric knowledge base is shown above the KB-InfoBot (missing values denoted by **X**).

periments that this framework allows the agent to achieve a higher task success rate in fewer dialogue turns. Further, the retrieval process is differentiable, allowing us to construct an end-to-end trainable KB-InfoBot, all of whose components are updated online using RL.

Reinforcement learners typically require an environment to interact with, and hence static dialogue corpora cannot be used for their training. Running experiments on human subjects, on the other hand, is unfortunately too expensive. A common workaround in the dialogue community (Young et al., 2013; Schatzmann et al., 2007b; Scheffler and Young, 2002) is to instead use *user simulators* which mimic the behavior of real users in a consistent manner. For training KB-InfoBot, we adapt the publicly available[2] simulator described in Li et al. (2016b).

Evaluation of dialogue agents has been the subject of much research (Walker et al., 1997; Möller et al., 2006). While the metrics for evaluating an InfoBot are relatively clear — the agent should return the correct entity in a minimum number of turns — the environment for testing it not so much. Unlike previous KB-based QA systems, our focus is on multi-turn interactions, and as such there are no publicly available benchmarks for this problem. We evaluate several versions of KB-InfoBot with the simulator and on real users, and show that the proposed Soft-KB lookup helps the reinforcement learner discover better dialogue policies. Initial experiments on the end-to-end agent also demonstrate its strong learning capability.

[2] https://github.com/MiuLab/TC-Bot

2 Related Work

Our work is motivated by the neural GenQA (Yin et al., 2016a) and neural enquirer (Yin et al., 2016b) models for querying KBs via natural language in a fully "neuralized" way. However, the key difference is that these systems assume that users can compose a complicated, compositional natural language query that can uniquely identify the element/answer in the KB. The research task is to parse the query, i.e., turning the natural language query into a sequence of SQL-like operations. Instead we focus on how to query a KB interactively without composing such complicated queries in the first place. Our work is motivated by the observations that (1) users are more used to issuing simple queries of length less than 5 words (Spink et al., 2001); (2) in many cases, it is unreasonable to assume that users can construct compositional queries without prior knowledge of the structure of the KB to be queried.

Also related is the growing body of literature focused on building end-to-end dialogue systems, which combine feature extraction and policy optimization using deep neural networks. Wen et al. (2016b) introduced a modular neural dialogue agent, which uses a Hard-KB lookup, thus breaking the differentiability of the whole system. As a result, training of various components of the dialogue system is performed separately. The intent network and belief trackers are trained using supervised labels specifically collected for them; while the policy network and generation network are trained separately on the system utterances. We retain modularity of the network by keeping the belief trackers separate, but replace the hard lookup with a differentiable one.

Dialogue agents can also interface with the database by augmenting their output action space with predefined API calls (Williams and Zweig, 2016; Zhao and Eskenazi, 2016; Bordes and Weston, 2016; Li et al., 2017). The API calls modify a query hypothesis maintained outside the end-to-end system which is used to retrieve results from this KB. This framework does not deal with uncertainty in language understanding since the query hypothesis can only hold one slot-value at a time. Our approach, on the other hand, directly models the uncertainty to construct the posterior over the KB.

Wu et al. (2015) presented an entropy minimization dialogue management strategy for In-

foBots. The agent always asks for the value of the slot with maximum entropy over the remaining entries in the database, which is optimal in the absence of language understanding errors, and serves as a baseline against our approach. Reinforcement learning neural turing machines (RL-NTM) (Zaremba and Sutskever, 2015) also allow neural controllers to interact with discrete external interfaces. The interface considered in that work is a one-dimensional memory tape, while in our work it is an entity-centric KB.

3 Probabilistic KB Lookup

This section describes a probabilistic framework for querying a KB given the agent's beliefs over the fields in the KB.

3.1 Entity-Centric Knowledge Base (EC-KB)

A Knowledge Base consists of triples of the form (h, r, t), which denotes that *relation* r holds between the *head* h and *tail* t. We assume that the KB-InfoBot has access to a domain-specific entity-centric knowledge base (EC-KB) (Zwicklbauer et al., 2013) where all head entities are of a particular type (such as movies or persons), and the relations correspond to attributes of these head entities. Such a KB can be converted to a table format whose rows correspond to the unique head entities, columns correspond to the unique relation types (*slots* henceforth), and some entries may be missing. An example is shown in Figure 1.

3.2 Notations and Assumptions

Let $\mathcal{T}$ denote the KB table described above and $\mathcal{T}_{i,j}$ denote the jth slot-value of the ith entity. $1 \leq i \leq N$ and $1 \leq j \leq M$. We let V^j denote the vocabulary of each slot, i.e. the set of all distinct values in the j-th column. We denote missing values from the table with a special token and write $\mathcal{T}_{i,j} = \Psi$. $M_j = \{i : \mathcal{T}_{i,j} = \Psi\}$ denotes the set of entities for which the value of slot j is missing. Note that the user may still know the actual value of $\mathcal{T}_{i,j}$, and we assume this lies in V^j. We do not deal with new entities or relations at test time.

We assume a uniform prior $G \sim \mathcal{U}[\{1, ...N\}]$ over the rows in the table $\mathcal{T}$, and let binary random variables $\Phi_j \in \{0, 1\}$ indicate whether the user knows the value of slot j or not. The agent maintains M multinomial distributions $p_j^t(v)$ for $v \in V^j$ denoting the probability at turn t that the user constraint for slot j is v, given their utterances U_1^t till that turn. The agent also maintains M binomials $q_j^t = \Pr(\Phi_j = 1)$ which denote the probability that the user knows the value of slot j.

We assume that column values are independently distributed to each other. This is a strong assumption but it allows us to model the user goal for each slot independently, as opposed to modeling the user goal over KB entities directly. Typically $\max_j |V^j| < N$ and hence this assumption reduces the number of parameters in the model.

3.3 Soft-KB Lookup

Let $p_{\mathcal{T}}^t(i) = \Pr(G = i|U_1^t)$ be the posterior probability that the user is interested in row i of the table, given the utterances up to turn t. We assume all probabilities are conditioned on user inputs U_1^t and drop it from the notation below. From our assumption of independence of slot values, we can write $p_{\mathcal{T}}^t(i) \propto \prod_{j=1}^{M} \Pr(G_j = i)$, where $\Pr(G_j = i)$ denotes the posterior probability of user goal for slot j pointing to $\mathcal{T}_{i,j}$. Marginalizing this over Φ_j gives:

$$\begin{aligned}\Pr(G_j = i) &= \sum_{\phi=0}^{1} \Pr(G_j = i, \Phi_j = \phi) \qquad (1)\\ &= q_j^t \Pr(G_j = i|\Phi_j = 1) + \\ &\quad (1 - q_j^t) \Pr(G_j = i|\Phi_j = 0).\end{aligned}$$

For $\Phi_j = 0$, the user does not know the value of the slot, and from the prior:

$$\Pr(G_j = i|\Phi_j = 0) = \frac{1}{N}, \quad 1 \leq i \leq N \qquad (2)$$

For $\Phi_j = 1$, the user knows the value of slot j, but this may be missing from $\mathcal{T}$, and we again have two cases:

$$\Pr(G_j = i|\Phi_j = 1) = \begin{cases} \frac{1}{N}, & i \in M_j \\ \frac{p_j^t(v)}{N_j(v)}\left(1 - \frac{|M_j|}{N}\right), & i \notin M_j \end{cases} \qquad (3)$$

Here, $N_j(v)$ is the count of value v in slot j. Detailed derivation for (3) is provided in Appendix A. Combining (1), (2), and (3) gives us the procedure for computing the posterior over KB entities.

4 Towards an End-to-End-KB-InfoBot

We claim that the Soft-KB lookup method has two benefits over the Hard-KB method – (1) it helps the agent discover better dialogue policies by providing it more information from the language understanding unit, (2) it allows end-to-end training of both dialogue policy and language understanding in an online setting. In this section we describe several agents to test these claims.

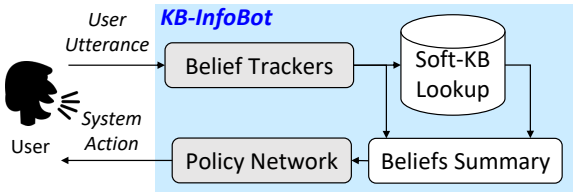


Figure 2: High-level overview of the end-to-end KB-InfoBot. Components with trainable parameters are highlighted in gray.

4.1 Overview

Figure 2 shows an overview of the components of the KB-InfoBot. At each turn, the agent receives a natural language utterance u^t as input, and selects an action a^t as output. The action space, denoted by $\mathcal{A}$, consists of $M+1$ actions — *request(slot=i)* for $1 \leq i \leq M$ will ask the user for the value of slot i, and *inform(I)* will inform the user with an ordered list of results I from the KB. The dialogue ends once the agent chooses *inform*.

We adopt a modular approach, typical to goal-oriented dialogue systems (Wen et al., 2016b), consisting of: a *belief tracker* module for identifying user intents, extracting associated slots, and tracking the dialogue state (Yao et al., 2014; Hakkani-Tür et al., 2016; Chen et al., 2016b; Henderson et al., 2014; Henderson, 2015); an interface with the database to query for relevant results (*Soft-KB* lookup); a *summary* module to summarize the state into a vector; a *dialogue policy* which selects the next system action based on current state (Young et al., 2013). We assume the agent only responds with dialogue acts. A template-based *Natural Language Generator* (NLG) can be easily constructed for converting dialogue acts into natural language.

4.2 Belief Trackers

The InfoBot consists of M belief trackers, one for each slot, which get the user input x^t and produce two outputs, p_j^t and q_j^t, which we shall collectively call the *belief state*: p_j^t is a multinomial distribution over the slot values v, and q_j^t is a scalar probability of the user knowing the value of slot j. We describe two versions of the belief tracker.

Hand-Crafted Tracker: We first identify mentions of slot-names (such as "actor") or slot-values (such as "Bill Murray") from the user input u^t, using token-level keyword search. Let $\{w \in x\}$ denote the set of tokens in a string x[3], then for each slot in $1 \leq j \leq M$ and each value $v \in V^j$, we compute its *matching score* as follows:

$$s_j^t[v] = \frac{|\{w \in u^t\} \cap \{w \in v\}|}{|\{w \in v\}|} \quad (4)$$

A similar score b_j^t is computed for the slot-names. A one-hot vector $req^t \in \{0,1\}^M$ denotes the previously requested slot from the agent, if any. q_j^t is set to 0 if $req^t[j]$ is 1 but $s_j^t[v] = 0\ \forall v \in V^j$, i.e. the agent requested for a slot but did not receive a valid value in return, else it is set to 1.

Starting from an prior distribution p_j^0 (based on the counts of the values in the KB), $p_j^t[v]$ is updated as:

$$p_j^t[v] \propto p_j^{t-1}[v] + C\left(s_j^t[v] + b_j^t + \mathbb{1}(req^t[j] = 1)\right) \quad (5)$$

Here C is a tuning parameter, and the normalization is given by setting the sum over v to 1.

Neural Belief Tracker: For the neural tracker the user input u^t is converted to a vector representation x^t, using a bag of n-grams (with $n = 2$) representation. Each element of x^t is an integer indicating the count of a particular n-gram in u^t. We let V^n denote the number of unique n-grams, hence $x^t \in \mathbb{N}_0^{V^n}$.

Recurrent neural networks have been used for belief tracking (Henderson et al., 2014; Wen et al., 2016b) since the output distribution at turn t depends on all user inputs till that turn. We use a Gated Recurrent Unit (GRU) (Cho et al., 2014) for each tracker, which, starting from $h_j^0 = \mathbf{0}$ computes $h_j^t = \text{GRU}(x^1, \ldots, x^t)$ (see Appendix B for details). $h_j^t \in \mathbb{R}^d$ can be interpreted as a summary of what the user has said about slot j till turn t. The belief states are computed from this vector as follows:

$$p_j^t = \text{softmax}(W_j^p h_j^t + b_j^p) \quad (6)$$

$$q_j^t = \sigma(W_j^\Phi h_j^t + b_j^\Phi) \quad (7)$$

Here $W_j^p \in \mathbb{R}^{V^j \times d}$, $b_j^p \in \mathbb{R}^{V^j}$, $W_j^\Phi \in \mathbb{R}^d$ and $b_j^\Phi \in \mathbb{R}$, are trainable parameters.

4.3 Soft-KB Lookup + Summary

This module uses the Soft-KB lookup described in section 3.3 to compute the posterior $p_\mathcal{T}^t \in \mathbb{R}^N$ over the EC-KB from the belief states $(p_j^t,\ q_j^t)$.

[3] We use the NLTK tokenizer available at http://www.nltk.org/api/nltk.tokenize.html

Collectively, outputs of the belief trackers and the soft-KB lookup can be viewed as the current dialogue state internal to the KB-InfoBot. Let $s^t = [p_1^t, p_2^t, ..., p_M^t, q_1^t, q_2^t, ..., q_M^t, p_\mathcal{T}^t]$ be the vector of size $\sum_j V^j + M + N$ denoting this state. It is possible for the agent to directly use this state vector to select its next action a^t. However, the large size of the state vector would lead to a large number of parameters in the policy network. To improve efficiency we extract summary statistics from the belief states, similar to (Williams and Young, 2005).

Each slot is summarized into an entropy statistic over a distribution w_j^t computed from elements of the KB posterior $p_\mathcal{T}^t$ as follows:

$$w_j^t(v) \propto \sum_{i:\mathcal{T}_{i,j}=v} p_\mathcal{T}^t(i) + p_j^0(v) \sum_{i:\mathcal{T}_{i,j}=\Psi} p_\mathcal{T}^t(i)\,. \tag{8}$$

Here, p_j^0 is a prior distribution over the values of slot j, estimated using counts of each value in the KB. The probability mass of v in this distribution is the agent's confidence that the user goal has value v in slot j. This two terms in (8) correspond to rows in KB which have value v, and rows whose value is unknown (weighted by the prior probability that an unknown might be v). Then the summary statistic for slot j is the entropy $H(w_j^t)$. The KB posterior $p_\mathcal{T}^t$ is also summarized into an entropy statistic $H(p_\mathcal{T}^t)$.

The scalar probabilities q_j^t are passed as is to the dialogue policy, and the final summary vector is $\tilde{s}^t = [H(\tilde{p}_1^t), ..., H(\tilde{p}_M^t), q_1^t, ..., q_M^t, H(p_\mathcal{T}^t)]$. Note that this vector has size $2M + 1$.

4.4 Dialogue Policy

The dialogue policy's job is to select the next action based on the current summary state $\tilde{s}^t$ and the dialogue history. We present a hand-crafted baseline and a neural policy network.

Hand-Crafted Policy: The rule based policy is adapted from (Wu et al., 2015). It asks for the slot $\hat{j} = \arg\min H(\tilde{p}_j^t)$ with the minimum entropy, except if – (i) the KB posterior entropy $H(p_\mathcal{T}^t) < \alpha_R$, (ii) $H(\tilde{p}_j^t) < \min(\alpha_T, \beta H(\tilde{p}_j^0)$, (iii) slot j has already been requested Q times. α_R, α_T, β, Q are tuned to maximize reward against the simulator.

Neural Policy Network: For the neural approach, similar to (Williams and Zweig, 2016; Zhao and Eskenazi, 2016), we use an RNN to allow the network to maintain an internal state of dialogue history. Specifically, we use a GRU unit followed by a fully-connected layer and softmax nonlinearity to model the policy π over actions in $\mathcal{A}$ ($W^\pi \in \mathrm{R}^{|\mathcal{A}|\times d}$, $b^\pi \in \mathrm{R}^{|\mathcal{A}|}$):

$$h_\pi^t = \text{GRU}(\tilde{s}^1, ..., \tilde{s}^t) \tag{9}$$

$$\pi = \text{softmax}(W^\pi h_\pi^t + b^\pi)\,. \tag{10}$$

During training, the agent samples its actions from the policy to encourage exploration. If this action is *inform()*, it must also provide an ordered set of entities indexed by $I = (i_1, i_2, \ldots, i_R)$ in the KB to the user. This is done by sampling R items from the KB-posterior $p_\mathcal{T}^t$. This mimics a search engine type setting, where R may be the number of results on the first page.

5 Training

Parameters of the neural components (denoted by θ) are trained using the REINFORCE algorithm (Williams, 1992). We assume that the learner has access to a reward signal r_t throughout the course of the dialogue, details of which are in the next section. We can write the expected discounted return of the agent under policy π as $J(\theta) = \mathbf{E}_\pi\left[\sum_{t=0}^H \gamma^t r_t\right]$ (γ is the discounting factor). We also use a baseline reward signal b, which is the average of all rewards in a batch, to reduce the variance in the updates (Greensmith et al., 2004). When only training the dialogue policy π using this signal, updates are given by (details in Appendix C):

$$\nabla_\theta J(\theta) = \mathbf{E}_\pi\Big[\sum_{k=0}^H \nabla_\theta \log \pi_\theta(a^k) \sum_{t=0}^H \gamma^t (r_t - b)\Big]\,, \tag{11}$$

For end-to-end training we need to update both the dialogue policy and the belief trackers using the reinforcement signal, and we can view the retrieval as another policy μ_θ (see Appendix C). The updates are given by:

$$\nabla_\theta J(\theta) = \mathbf{E}_{a\sim\pi, I\sim\mu}\Big[\big(\nabla_\theta \log \mu_\theta(I) + \sum_{h=0}^H \nabla_\theta \log \pi_\theta(a_h)\big) \sum_{k=0}^H \gamma^k (r_k - b)\Big]\,, \tag{12}$$

In the case of end-to-end learning, we found that for a moderately sized KB, the agent almost always fails if starting from random initialization.

In this case, credit assignment is difficult for the agent, since it does not know whether the failure is due to an incorrect sequence of actions or incorrect set of results from the KB. Hence, at the beginning of training we have an *Imitation Learning* (IL) phase where the belief trackers and policy network are trained to mimic the hand-crafted agents. Assume that $\hat{p}_j^t$ and $\hat{q}_j^t$ are the belief states from a rule-based agent, and $\hat{a}^t$ its action at turn t. Then the loss function for imitation learning is:

$$\mathcal{L}(\theta) = \mathbf{E}\big[D(\hat{p}_j^t||p_j^t(\theta)) + H(\hat{q}_j^t, q_j^t(\theta)) - \log \pi_\theta(\hat{a}^t)\big]$$

$D(p||q)$ and $H(p,q)$ denote the KL divergence and cross-entropy between p and q respectively.

The expectations are estimated using a mini-batch of dialogues of size B. For RL we use RMSProp (Hinton et al., 2012) and for IL we use vanilla SGD updates to train the parameters θ.

6 Experiments and Results

Previous work in KB-based QA has focused on single-turn interactions and is not directly comparable to the present study. Instead we compare different versions of the KB-InfoBot described above to test our claims.

6.1 KB-InfoBot versions

We have described two belief trackers – (A) Hand-Crafted and (B) Neural, and two dialogue policies – (C) Hand-Crafted and (D) Neural.

Rule agents use the hand-crafted belief trackers and hand-crafted policy (A+C). **RL** agents use the hand-crafted belief trackers and the neural policy (A+D). We compare three variants of both sets of agents, which differ only in the inputs to the dialogue policy. The *No-KB* version only takes entropy $H(\hat{p}_j^t)$ of each of the slot distributions. The *Hard-KB* version performs a hard-KB lookup and selects the next action based on the entropy of the slots over retrieved results. This is the same approach as in Wen et al. (2016b), except that we take entropy instead of summing probabilities. The *Soft-KB* version takes summary statistics of the slots and KB posterior described in Section 4. At the end of the dialogue, all versions inform the user with the top results from the KB posterior $p_\mathcal{T}^t$, hence the difference only lies in the policy for action selection. Lastly, the **E2E** agent uses the neural belief tracker and the neural policy (B+D), with a Soft-KB lookup. For the RL agents, we also append $\hat{q}_j^t$ and a one-hot encoding of the previous agent action to the policy network input. Hyperparameter details for the agents are provided in Appendix D.

KB-split	N	M	$\max_j \lvert V^j \rvert$	$\lvert M_j \rvert$
Small	277	6	17	20%
Medium	428	6	68	20%
Large	857	6	101	20%
X-Large	3523	6	251	20%

Table 1: Movies-KB statistics for four splits. Refer to Section 3.2 for description of columns.

6.2 User Simulator

Training reinforcement learners is challenging because they need an environment to operate in. In the dialogue community it is common to use simulated users for this purpose (Schatzmann et al., 2007a,b; Cuayáhuitl et al., 2005; Asri et al., 2016). In this work we adapt the publicly-available user simulator presented in Li et al. (2016b) to follow a simple agenda while interacting with the KB-InfoBot, as well as produce natural language utterances . Details about the simulator are included in Appendix E. During training, the simulated user also provides a reward signal at the end of each dialogue. The dialogue is a success if the user target is in top $R = 5$ results returned by the agent; and the reward is computed as $\max(0, 2(1-(r-1)/R))$, where r is the actual rank of the target. For a failed dialogue the agent receives a reward of -1, and at each turn it receives a reward of -0.1 to encourage short sessions[4]. The maximum length of a dialogue is 10 turns beyond which it is deemed a failure.

6.3 Movies-KB

We use a movie-centric KB constructed using the IMDBPy[5] package. We constructed four different splits of the dataset, with increasing number of entities, whose statistics are given in Table 1. The original KB was modified to reduce the number of actors and directors in order to make the task more challenging[6]. We randomly remove 20% of the values from the agent's copy of the KB to simulate a scenario where the KB may be incomplete. The user, however, may still know these values.

[4] A turn consists of one user action and one agent action.

[5] http://imdbpy.sourceforge.net/

[6] We restricted the vocabulary to the first few unique values of these slots and replaced all other values with a random value from this set.

	Agent	Small KB			Medium KB			Large KB			X-Large KB		
		T	S	R	T	S	R	T	S	R	T	S	R
No KB	Rule	5.04	.64	.26±.02	5.05	.77	.74±.02	4.93	.78	.82±.02	4.84	.66	.43±.02
	RL	2.65	.56	.24±.02	3.32	.76	.87±.02	3.71	.79	.94±.02	3.64	.64	.50±.02
Hard KB	Rule	5.04	.64	.25±.02	3.66	.73	.75±.02	4.27	.75	.78±.02	4.84	.65	.42±.02
	RL	3.36	.62	.35±.02	**3.07**	.75	.86±.02	3.53	.79	.98±.02	**2.88**	.62	.53±.02
Soft KB	Rule	**2.12**	.57	.32±.02	3.94	.76	.83±.02	3.74	.78	.93±.02	4.51	.66	.51±.02
	RL	2.93	.63	.43±.02	3.37	.80	.98±.02	3.79	.83	1.05±.02	3.65	**.68**	**.62±.02**
	E2E	3.13	**.66**	**.48±.02**	3.27	**.83**	**1.10±.02**	**3.51**	**.83**	**1.10±.02**	3.98	.65	.50±.02
Max		*3.44*	*1.0*	*1.64*	*2.96*	*1.0*	*1.78*	*3.26*	*1.0*	*1.73*	*3.97*	*1.0*	*1.37*

Table 2: Performance comparison. Average (±std error) for 5000 runs after choosing the best model during training. **T:** Average number of turns. **S:** Success rate. **R:** Average reward.

6.4 Simulated User Evaluation

We compare each of the discussed versions along three metrics: the average rewards obtained (R), success rate (S) (where success is defined as providing the user target among top R results), and the average number of turns per dialogue (T). For the RL and E2E agents, during training we fix the model every 100 updates and run 2000 simulations with greedy action selection to evaluate its performance. Then after training we select the model with the highest average reward and run a further 5000 simulations and report the performance in Table 2. For reference we also show the performance of an agent which receives perfect information about the user target without any errors, and selects actions based on the entropy of the slots (*Max*). This can be considered as an upper bound on the performance of any agent (Wu et al., 2015).

In each case the Soft-KB versions achieve the highest average reward, which is the metric all agents optimize. In general, the trade-off between minimizing average turns and maximizing success rate can be controlled by changing the reward signal. Note that, except the E2E version, all versions share the same belief trackers, but by re-asking values of some slots they can have different posteriors p_T^t to inform the results. This shows that having full information about the current state of beliefs over the KB helps the Soft-KB agent discover better policies. Further, reinforcement learning helps discover better policies than the hand-crafted rule-based agents, and we see a higher reward for RL agents compared to Rule ones. This is due to the noisy natural language inputs; with perfect information the rule-based strategy is optimal. Interestingly, the RL-Hard agent has the minimum number of turns in 2 out of the 4 settings, at the cost of a lower success rate and average reward. This agent does not receive any information about the uncertainty in semantic parsing, and it tends to inform as soon as the number of retrieved results becomes small, even if they are incorrect.

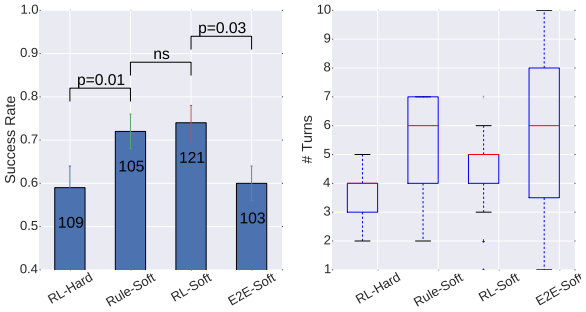


Figure 3: Performance of KB-InfoBot versions when tested against real users. **Left:** Success rate, with the number of test dialogues indicated on each bar, and the p-values from a two-sided permutation test. **Right:** Distribution of the number of turns in each dialogue (differences in mean are significant with $p < 0.01$).

Among the Soft-KB agents, we see that E2E>RL>Rule, except for the X-Large KB. For E2E, the action space grows exponentially with the size of the KB, and hence credit assignment gets more difficult. Future work should focus on improving the E2E agent in this setting. The difficulty of a KB-split depends on number of entities it has, as well as the number of unique values for each slot (more unique values make the problem easier). Hence we see that both the "Small" and "X-Large" settings lead to lower reward for the agents, since $\frac{\max_j |V^j|}{N}$ is small for them.

6.5 Human Evaluation

We further evaluate the KB-InfoBot versions trained using the simulator against real subjects, recruited from the author's affiliations. In each session, in a typed interaction, the subject was first presented with a target movie from the "Medium" KB-split along with a subset of its associated slot-

Turn	Dialogue	Rank	Dialogue	Rank	Dialogue	Rank
1	can i get a movie directed by maiellaro *request actor*	75	find a movie directed by hemecker *request actor*	7	peter greene acted in a family comedy - what was it? *request actor*	35
2	neal *request mpaa_rating*	2	i dont know *request mpaa_rating*	7	peter *request mpaa_rating*	28
3	not sure about that *request critic_rating*	2	i dont know *request critic_rating*	7	i don't know that *request critic_rating*	28
4	i don't remember *request genre*	2	7.6 *request critic_rating*	13	the critics rated it as 6.5 *inform*	3
5	i think it's a crime movie *inform*	1	7.9 *request critic_rating*	23		
6			7.7 *inform*	41		

Figure 4: Sample dialogues between users and the KB-InfoBot (RL-Soft version). Each turn begins with a user utterance followed by the *agent response*. **Rank** denotes the rank of the target movie in the KB-posterior after each turn.

values from the KB. To simulate the scenario where end-users may not know slot values correctly, the subjects in our evaluation were presented multiple values for the slots from which they could choose any one while interacting with the agent. Subjects were asked to initiate the conversation by specifying some of these values, and respond to the agent's subsequent requests, all in natural language. We test RL-Hard and the three Soft-KB agents in this study, and in each session one of the agents was picked at random for testing. In total, we collected 433 dialogues, around 20 per subject. Figure 3 shows a comparison of these agents in terms of success rate and number of turns, and Figure 4 shows some sample dialogues from the user interactions with RL-Soft.

In comparing Hard-KB versus Soft-KB lookup methods we see that both Rule-Soft and RL-Soft agents achieve a higher success rate than RL-Hard, while E2E-Soft does comparably. They do so in an increased number of average turns, but achieve a higher average reward as well. Between RL-Soft and Rule-Soft agents, the success rate is similar, however the RL agent achieves that rate in a lower number of turns on average. RL-Soft achieves a success rate of 74% on the human evaluation and 80% against the simulated user, indicating minimal overfitting. However, all agents take a higher number of turns against real users as compared to the simulator, due to the noisier inputs.

The E2E gets the highest success rate against the simulator, however, when tested against real users it performs poorly with a lower success rate and a higher number of turns. Since it has more trainable components, this agent is also most prone to overfitting. In particular, the vocabulary of the simulator it is trained against is quite limited ($V^n = 3078$), and hence when real users

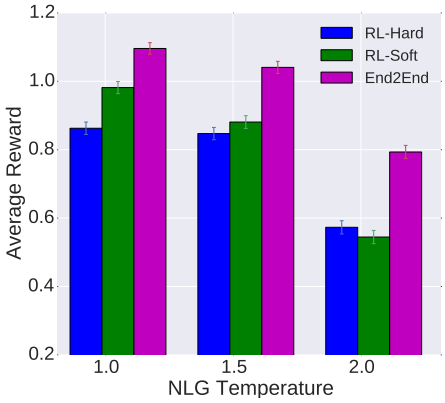


Figure 5: Average rewards against simulator as temperature of softmax in NLG output is increased. Higher temperature leads to more noise in output. Average over 5000 simulations after selecting the best model during training.

provided inputs outside this vocabulary, it performed poorly. In the future we plan to fix this issue by employing a better architecture for the language understanding and belief tracker components Hakkani-Tür et al. (2016); Liu and Lane (2016); Chen et al. (2016b,a), as well as by pretraining on separate data.

While its generalization performance is poor, the E2E system also exhibits the strongest learning capability. In Figure 5, we compare how different agents perform against the simulator as the temperature of the output softmax in its NLG is increased. A higher temperature means a more uniform output distribution, which leads to generic simulator responses irrelevant to the agent questions. This is a simple way of introducing noise in the utterances. The performance of all agents drops as the temperature is increased, but less so for the E2E agent, which can adapt its belief tracker to the inputs it receives. Such adaptation

is key to the personalization of dialogue agents, which motivates us to introduce the E2E agent.

7 Conclusions and Discussion

This work is aimed at facilitating the move towards end-to-end trainable dialogue agents for information access. We propose a differentiable probabilistic framework for querying a database given the agent's beliefs over its fields (or slots). We show that such a framework allows the downstream reinforcement learner to discover better dialogue policies by providing it more information. We also present an E2E agent for the task, which demonstrates a strong learning capacity in simulations but suffers from overfitting when tested on real users. Given these results, we propose the following deployment strategy that allows a dialogue system to be tailored to specific users via learning from agent-user interactions. The system could start off with an RL-Soft agent (which gives good performance out-of-the-box). As the user interacts with this agent, the collected data can be used to train the E2E agent, which has a strong learning capability. Gradually, as more experience is collected, the system can switch from RL-Soft to the personalized E2E agent. Effective implementation of this, however, requires the E2E agent to learn quickly and this is the research direction we plan to focus on in the future.

Acknowledgements

We would like to thank Dilek Hakkani-Tür and reviewers for their insightful comments on the paper. We would also like to acknowledge the volunteers from Carnegie Mellon University and Microsoft Research for helping us with the human evaluation. Yun-Nung Chen is supported by the Ministry of Science and Technology of Taiwan under the contract number 105-2218-E-002-033, Institute for Information Industry, and MediaTek.

References

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems. *arXiv preprint arXiv:1607.00070* .

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683* .

Yun-Nung Chen, Dilek Hakanni-Tür, Gokhan Tur, Asli Celikyilmaz, Jianfeng Guo, and Li Deng. 2016a. Syntax or semantics? knowledge-guided joint semantic frame parsing.

Yun-Nung Chen, Dilek Hakkani-Tür, Gokhan Tur, Jianfeng Gao, and Li Deng. 2016b. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP* .

Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. 2005. Human-computer dialogue simulation using hidden markov models. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*. IEEE, pages 290–295.

M Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. Online policy optimisation of bayesian spoken dialogue systems via human interaction. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pages 8367–8371.

Peter W Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM* 33(10):75–84.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research* 5(Nov):1471–1530.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*.

Matthew Henderson. 2015. Machine learning for dialog state tracking: A review. *Machine Learning in Spoken Language Processing Workshop* .

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. pages 292–299.

Geoffrey Hinton, N Srivastava, and Kevin Swersky. 2012. Lecture 6a overview of mini–batch gradient descent. *Coursera Lecture slides https://class. coursera. org/neuralnets-2012-001/lecture,[Online* .

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016a. Deep reinforcement learning for dialogue generation. *EMNLP* .

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016b. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688* .

Xuijun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. 2017. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008* .

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech 2016* pages 685–689.

Sebastian Möller, Roman Englert, Klaus-Peter Engelbrecht, Verena Vanessa Hafner, Anthony Jameson, Antti Oulasvirta, Alexander Raake, and Norbert Reithinger. 2006. Memo: towards automatic usability evaluation of spoken dialogue services by user error simulations. In *INTERSPEECH*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007a. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Association for Computational Linguistics, pages 149–152.

Jost Schatzmann, Blaise Thomson, and Steve Young. 2007b. Statistical user simulation with a hidden agenda. *Proc SIGDial, Antwerp* 273282(9).

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., pages 12–19.

Amanda Spink, Dietmar Wolfram, Major BJ Jansen, and Tefko Saracevic. 2001. Searching the web: The public and their queries. *Journal of the Association for Information Science and Technology* 52(3):226–234.

Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 271–280.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016a. Conditional generation and snapshot learning in neural dialogue systems. *EMNLP* .

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016b. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562* .

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *EMNLP* .

Jason D Williams and Steve Young. 2005. Scaling up POMDPs for dialog management: The "Summary POMDP" method. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*. IEEE, pages 177–182.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269* .

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Ji Wu, Miao Li, and Chin-Hui Lee. 2015. A probabilistic framework for representing dialog systems and entropy-based dialog management through dynamic stochastic state evolution. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23(11):2026–2035.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pages 189–194.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016a. Neural generative question answering. *International Joint Conference on Artificial Intelligence* .

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016b. Neural enquirer: Learning to query tables. *International Joint Conference on Artificial Intelligence* .

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural Turing machines-revised. *arXiv preprint arXiv:1505.00521* .

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560* .

Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. 2013. Do we need entity-centric knowledge bases for entity disambiguation? In *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*. ACM, page 4.

A Posterior Derivation

Here, we present a derivation for equation 3, i.e., the posterior over the KB slot when the user knows the value of that slot. For brevity, we drop $\Phi_j = 0$ from the condition in all probabilities below. For the case when $i \in M_j$, we can write:

$$\begin{aligned}
&\Pr(G_j = i) \\
&= \Pr(G_j \in M_j) \Pr(G_j = i | G_j \in M_j) \\
&= \frac{|M_j|}{N} \frac{1}{|M_j|} = \frac{1}{N}\,,
\end{aligned} \tag{13}$$

where we assume all missing values to be equally likely, and estimate the prior probability of the goal being missing from the count of missing values in that slot. For the case when $i = v \notin M_j$:

$$\begin{aligned}
&\Pr(G_j = i) \\
&= \Pr(G_j \notin M_j) \Pr(G_j = i | G_j \notin M_j) \\
&= \left(1 - \frac{|M_j|}{N}\right) \times \frac{p_j^t(v)}{N_j(v)}\,,
\end{aligned} \tag{14}$$

where the second term comes from taking the probability mass associated with v in the belief tracker and dividing it equally among all rows with value v.

We can also verify that the above distribution is valid: i.e., it sums to 1:

$$\begin{aligned}
&\sum_i \Pr(G_j = i) \\
&= \sum_{i \in M_j} \Pr(G_j = i) + \sum_{i \notin M_j} \Pr(G_j = i) \\
&= \sum_{i \in M_j} \frac{1}{N} + \sum_{i \notin M_j} \left(1 - \frac{|M_j|}{N}\right) \frac{p_j^t(v)}{\#_j v} \\
&= \frac{|M_j|}{N} + \left(1 - \frac{|M_j|}{N}\right) \sum_{i \notin M_j} \frac{p_j^t(v)}{\#_j v} \\
&= \frac{|M_j|}{N} + \left(1 - \frac{|M_j|}{N}\right) \sum_{i \in V^j} \#_j v \frac{p_j^t(v)}{\#_j v} \\
&= \frac{|M_j|}{N} + \left(1 - \frac{|M_j|}{N}\right) \times 1 \\
&= 1\,.
\end{aligned}$$

B Gated Recurrent Units

A Gated Recurrent Unit (GRU) (Cho et al., 2014) is a recurrent neural network which operates on an input sequence $x_1, \ldots, x_t$. Starting from an initial state h_0 (usually set to $\mathbf{0}$ it iteratively computes the final output h_t as follows:

$$\begin{aligned}
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\
\tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t\,.
\end{aligned} \tag{15}$$

Here σ denotes the sigmoid nonlinearity, and $\odot$ an element-wise product.

C REINFORCE updates

We assume that the learner has access to a reward signal r_t throughout the course of the dialogue, details of which are in the next section. We can write the expected discounted return of the agent under policy π as follows:

$$J(\theta) = \mathbf{E}\left[\sum_{t=0}^{H} \gamma^t r_t\right] \tag{16}$$

Here, the expectation is over all possible trajectories τ of the dialogue, θ denotes the trainable parameters of the learner, H is the maximum length of an episode, and γ is the discounting factor. We can use the likelihood ratio trick (Glynn, 1990) to write the gradient of the objective as follows:

$$\nabla_\theta J(\theta) = \mathbf{E}\left[\nabla_\theta \log p_\theta(\tau) \sum_{t=0}^{H} \gamma^t r_t\right]\,, \tag{17}$$

where $p_\theta(\tau)$ is the probability of observing a particular trajectory under the current policy. With a Markovian assumption, we can write

$$p_\theta(\tau) = p(s_0) \prod_{k=0}^{H} p(s_{k+1}|s_k, a_k) \pi_\theta(a_k|s_k), \tag{18}$$

where θ denotes dependence on the neural network parameters. From 17,18 we obtain

$$\nabla_\theta J(\theta) = \mathbf{E}_{a \sim \pi}\Big[\sum_{k=0}^{H} \nabla_\theta \log \pi_\theta(a_k) \sum_{t=0}^{H} \gamma^t r_t\Big]\,, \tag{19}$$

If we need to train both the policy network and the belief trackers using the reinforcement signal, we can view the KB posterior $p_\mathcal{T}^t$ as another policy. During training then, to encourage exploration, when the agent selects the *inform* action we

sample R results from the following distribution to return to the user:

$$\mu(I) = p_{\mathcal{T}}^t(i_1) \times \frac{p_{\mathcal{T}}^t(i_2)}{1 - p_{\mathcal{T}}^t(i_1)} \times \cdots . \quad (20)$$

This formulation also leads to a modified version of the episodic REINFORCE update rule (Williams, 1992). Specifically, eq. 18 now becomes,

$$p_\theta(\tau) = \left[p(s_0) \prod_{k=0}^{H} p(s_{k+1}|s_k, a_k) \pi_\theta(a_k|s_k) \right] \mu_\theta(I), \quad (21)$$

Notice the last term μ_θ above which is the posterior of a set of results from the KB. From 17,21 we obtain

$$\nabla_\theta J(\theta) = \mathbf{E}_{a\sim\pi, I\sim\mu} \Big[\big(\nabla_\theta \log \mu_\theta(I) + \sum_{h=0}^{H} \nabla_\theta \log \pi_\theta(a_h) \big) \sum_{k=0}^{H} \gamma^k r_k \Big] , \quad (22)$$

D Hyperparameters

We use GRU hidden state size of $d = 50$ for the RL agents and $d = 100$ for the E2E, a learning rate of 0.05 for the imitation learning phase and 0.005 for the reinforcement learning phase, and minibatch size 128. For the rule agents, hyperparameters were tuned to maximize the average reward of each agent in simulations. For the E2E agent, imitation learning was performed for 500 updates, after which the agent switched to reinforcement learning. The input vocabulary is constructed from the NLG vocabulary and bigrams in the KB, and its size is 3078.

E User Simulator

At the beginning of each dialogue, the simulated user randomly samples a target entity from the EC-KB and a random combination of *informable slots* for which it knows the value of the target. The remaining slot-values are unknown to the user. The user initiates the dialogue by providing a subset of its informable slots to the agent and requesting for an entity which matches them. In subsequent turns, if the agent requests for the value of a slot, the user complies by providing it or informs the agent that it does not know that value. If the agent informs results from the KB, the simulator checks whether the target is among them and provides the reward.

We convert dialogue acts from the user into natural language utterances using a separately trained natural language generator (NLG). The NLG is trained in a sequence-to-sequence fashion, using conversations between humans collected by crowd-sourcing. It takes the dialogue actions (DAs) as input, and generates template-like sentences with slot placeholders via an LSTM decoder. Then, a post-processing scan is performed to replace the slot placeholders with their actual values, which is similar to the decoder module in (Wen et al., 2015, 2016a). In the LSTM decoder, we apply beam search, which iteratively considers the top k best sentences up to time step t when generating the token of the time step $t + 1$. For the sake of the trade-off between the speed and performance, we use the beam size of 3 in the following experiments.

There are several sources of error in user utterances. Any value provided by the user may be corrupted by noise, or substituted completely with an incorrect value of the same type (e.g., "Bill Murray" might become just "Bill" or "Tom Cruise"). The NLG described above is inherently stochastic, and may sometimes generate utterances irrelevant to the agent request. By increasing the temperature of the output softmax in the NLG we can increase the noise in user utterances.

Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-Based Chatbots

Yu Wu[†], **Wei Wu**[‡], **Chen Xing**[♢], **Zhoujun Li**[†*], **Ming Zhou**[‡]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[♢]College of Computer and Control Engineering, Nankai University, Tianjin, China
[‡] Microsoft Research, Beijing, China
{wuyu,lizj}@buaa.edu.cn {wuwei,v-chxing,mingzhou}@microsoft.com

Abstract

We study response selection for multi-turn conversation in retrieval-based chatbots. Existing work either concatenates utterances in context or matches a response with a highly abstract context vector finally, which may lose relationships among utterances or important contextual information. We propose a sequential matching network (SMN) to address both problems. SMN first matches a response with each utterance in the context on multiple levels of granularity, and distills important matching information from each pair as a vector with convolution and pooling operations. The vectors are then accumulated in a chronological order through a recurrent neural network (RNN) which models relationships among utterances. The final matching score is calculated with the hidden states of the RNN. An empirical study on two public data sets shows that SMN can significantly outperform state-of-the-art methods for response selection in multi-turn conversation.

1 Introduction

Conversational agents include task-oriented dialog systems and non-task-oriented chatbots. Dialog systems focus on helping people complete specific tasks in vertical domains (Young et al., 2010), while chatbots aim to naturally and meaningfully converse with humans on open domain topics (Ritter et al., 2011). Existing work on building chatbots includes generation -based methods and retrieval-based methods. Retrieval based chatbots enjoy the advantage of informative and fluent responses, because they select a proper response for the current conversation from a repository with response selection algorithms. While most existing work on retrieval-based chatbots studies response selection for single-turn conversation (Wang et al., 2013) which only considers the last input message, we consider the problem in a multi-turn scenario. In a chatbot, multi-turn response selection takes a message and utterances in its previous turns as input and selects a response that is natural and relevant to the whole context.

	Context
utterance 1	*Human*: How are you doing?
utterance 2	*ChatBot*: I am going to **hold a drum class** in Shanghai. Anyone wants to join? The location is near Lujiazui.
utterance 3	*Human*: Interesting! Do you have coaches who can help me practice **drum**?
utterance 4	*ChatBot*: Of course.
utterance 5	*Human*: Can I have a free first lesson?
	Response Candidates
response 1	Sure. Have you ever played drum before? ✓
response 2	What lessons do you want? ✗

Table 1: An example of multi-turn conversation

The key to response selection lies in input-response matching. Different from single-turn conversation, multi-turn conversation requires matching between a response and a conversation context in which one needs to consider not only the matching between the response and the input message but also matching between responses and utterances in previous turns. The challenges of the task include (1) how to identify important information (words, phrases, and sentences) in context, which is crucial to selecting a proper response and leveraging relevant information in matching; and (2) how to model relationships among the utterances in the context. Table 1 illustrates the challenges with an example. First, "hold a drum class" and "drum" in context are very important. Without them, one may find responses relevant to the message (i.e., the fifth utterance of the context) but nonsense in the context (e.g., "what lessons do you want?"). Second, the message highly depends on the second utterance in the context, and

* Corresponding Author

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 496–505
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1046

the order of the utterances matters in response selection: exchanging the third utterance and the fifth utterance may lead to different responses. Existing work, however, either ignores relationships among utterances when concatenating them together (Lowe et al., 2015), or loses important information in context in the process of converting the whole context to a vector without enough supervision from responses (e.g., by a hierarchical RNN (Zhou et al., 2016)).

We propose a sequential matching network (SMN), a new context based matching model that can tackle both challenges in an end-to-end way. The reason that existing models lose important information in the context is that they first represent the whole context as a vector and then match the context vector with a response vector. Thus, responses in these models connect with the context until the final step in matching. To avoid information loss, SMN matches a response with each utterance in the context at the beginning and encodes important information in each pair into a matching vector. The matching vectors are then accumulated in the utterances' temporal order to model their relationships. The final matching degree is computed with the accumulation of the matching vectors. Specifically, for each utterance-response pair, the model constructs a word-word similarity matrix and a sequence-sequence similarity matrix by the word embeddings and the hidden states of a recurrent neural network with gated recurrent units (GRU) (Chung et al., 2014) respectively. The two matrices capture important matching information in the pair on a word level and a segment (word subsequence) level respectively, and the information is distilled and fused as a matching vector through an alternation of convolution and pooling operations on the matrices. By this means, important information from multiple levels of granularity in context is recognized under sufficient supervision from the response and carried into matching with minimal loss. The matching vectors are then uploaded to another GRU to form a matching score for the context and the response. The GRU accumulates the pair matching in its hidden states in the chronological order of the utterances in context. It models relationships and dependencies among the utterances in a matching fashion and has the utterance order supervise the accumulation of pair matching. The matching degree of the context and the response is computed by a logit model with the hidden states of the GRU. SMN extends the powerful "2D" matching paradigm in text pair matching for single-turn conversation to context based matching for multi-turn conversation, and enjoys the advantage of both important information in utterance-response pairs and relationships among utterances being sufficiently preserved and leveraged in matching.

We test our model on the Ubuntu dialogue corpus (Lowe et al., 2015) which is a large scale publicly available English data set for research in multi-turn conversation. The results show that our model can significantly outperform state-of-the-art methods, and improvement to the best baseline model on $R_{10}@1$ is over 6%. In addition to the Ubuntu corpus, we create a human-labeled Chinese data set, namely the Douban Conversation Corpus, and test our model on it. In contrast to the Ubuntu corpus in which data is collected from a specific domain and negative candidates are randomly sampled, conversations in this data come from the open domain, and response candidates in this data set are collected from a retrieval engine and labeled by three human judges. On this data, our model improves the best baseline model by 3% on $R_{10}@1$ and 4% on $P@1$. As far as we know, Douban Conversation Corpus is the first human-labeled data set for multi-turn response selection and could be a good complement to the Ubuntu corpus. We have released Douban Conversation Corups and our source code at `https://github.com/MarkWuNLP/MultiTurnResponseSelection`

Our contributions in this paper are three-folds: (1) the proposal of a new context based matching model for multi-turn response selection in retrieval-based chatbots; (2) the publication of a large human-labeled data set to research communities; (3) empirical verification of the effectiveness of the model on public data sets.

2 Related Work

Recently, building a chatbot with data driven approaches (Ritter et al., 2011; Ji et al., 2014) has drawn significant attention. Existing work along this line includes retrieval-based methods (Hu et al., 2014; Ji et al., 2014; Wang et al., 2015; Yan et al., 2016; Wu et al., 2016b; Zhou et al., 2016; Wu et al., 2016a) and generation-based methods (Shang et al., 2015; Serban et al., 2015; Vinyals and Le, 2015; Li et al., 2015, 2016; Xing et al.,

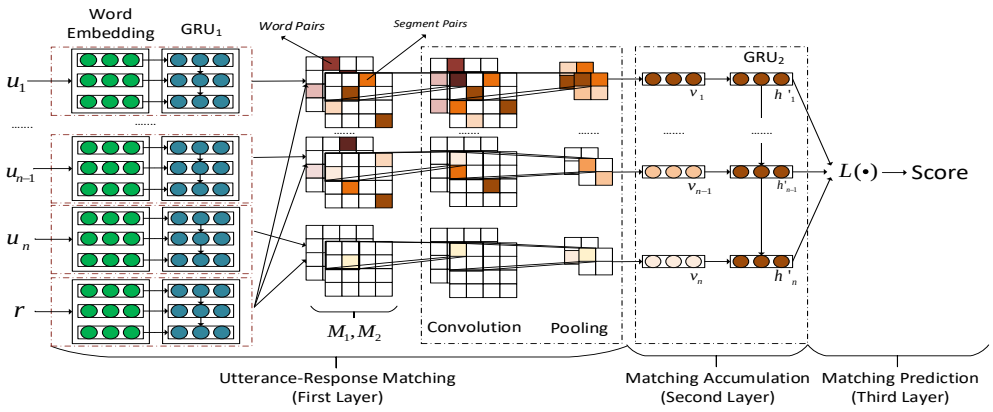


Figure 1: Architecture of SMN

2016; Serban et al., 2016). Our work is a retrieval-based method, in which we study context-based response selection.

Early studies of retrieval-based chatbots focus on response selection for single-turn conversation (Wang et al., 2013; Ji et al., 2014; Wang et al., 2015; Wu et al., 2016b). Recently, researchers have begun to pay attention to multi-turn conversation. For example, Lowe et al. (2015) match a response with the literal concatenation of context utterances. Yan et al. (2016) concatenate context utterances with the input message as reformulated queries and perform matching with a deep neural network architecture. Zhou et al. (2016) improve multi-turn response selection with a multi-view model including an utterance view and a word view. Our model is different in that it matches a response with each utterance at first and accumulates matching information instead of sentences by a GRU, thus useful information for matching can be sufficiently retained.

3 Sequential Matching Network

3.1 Problem Formalization

Suppose that we have a data set $\mathcal{D} = \{(y_i, s_i, r_i)\}_{i=1}^{N}$, where $s_i = \{u_{i,1}, \ldots, u_{i,n_i}\}$ represents a conversation context with $\{u_{i,k}\}_{k=1}^{n_i}$ as utterances. r_i is a response candidate and $y_i \in \{0, 1\}$ denotes a label. $y_i = 1$ means r_i is a proper response for s_i, otherwise $y_i = 0$. Our goal is to learn a matching model $g(\cdot, \cdot)$ with $\mathcal{D}$. For any context-response pair (s, r), $g(s, r)$ measures the matching degree between s and r.

3.2 Model Overview

We propose a sequential matching network (SMN) to model $g(\cdot, \cdot)$. Figure 1 gives the architecture. SMN first decomposes context-response matching into several utterance-response pair matching and then all pairs matching are accumulated as a context based matching through a recurrent neural network. SMN consists of three layers. The first layer matches a response candidate with each utterance in the context on a word level and a segment level, and important matching information from the two levels is distilled by convolution, pooling and encoded in a matching vector. The matching vectors are then fed into the second layer where they are accumulated in the hidden states of a recurrent neural network with GRU following the chronological order of the utterances in the context. The third layer calculates the final matching score with the hidden states of the second layer.

SMN enjoys several advantages over existing models. First, a response candidate can match each utterance in the context at the very beginning, thus matching information in every utterance-response pair can be sufficiently extracted and carried to the final matching score with minimal loss. Second, information extraction from each utterance is conducted on different levels of granularity and under sufficient supervision from the response, thus semantic structures that are useful for response selection in each utterance can be well identified and extracted. Third, matching and utterance relationships are coupled rather than separately modeled, thus utterance relationships (e.g., order), as a kind of knowledge, can supervise the formation of the matching score.

By taking utterance relationships into account, SMN extends the "2D" matching that has proven effective in text pair matching for single-turn response selection to sequential "2D" matching for

context based matching in response selection for multi-turn conversation. In the following sections, we will describe details of the three layers.

3.3 Utterance-Response Matching

Given an utterance u in a context s and a response candidate r, the model looks up an embedding table and represents u and r as $\mathbf{U} = [e_{u,1}, \ldots, e_{u,n_u}]$ and $\mathbf{R} = [e_{r,1}, \ldots, e_{r,n_r}]$ respectively, where $e_{u,i}, e_{r,i} \in \mathbb{R}^d$ are the embeddings of the i-th word of u and r respectively. $\mathbf{U} \in \mathbb{R}^{d \times n_u}$ and $\mathbf{R} \in \mathbb{R}^{d \times n_r}$ are then used to construct a word-word similarity matrix $\mathbf{M}_1 \in \mathbb{R}^{n_u \times n_r}$ and a sequence-sequence similarity matrix $\mathbf{M}_2 \in \mathbb{R}^{n_u \times n_r}$ which are two input channels of a convolutional neural network (CNN). The CNN distills important matching information from the matrices and encodes the information into a matching vector v.

Specifically, $\forall i, j$, the (i, j)-th element of $\mathbf{M}_1$ is defined by

$$e_{1,i,j} = e_{u,i}^{\top} \cdot e_{r,j}. \tag{1}$$

$\mathbf{M}_1$ models the matching between u and r on a word level.

To construct $\mathbf{M}_2$, we first employ a GRU to transform $\mathbf{U}$ and $\mathbf{R}$ to hidden vectors. Suppose that $\mathbf{H}_u = [h_{u,1}, \ldots, h_{u,n_u}]$ are the hidden vectors of $\mathbf{U}$, then $\forall i$, $h_{u,i} \in \mathbb{R}^m$ is defined by

$$\begin{aligned} z_i &= \sigma(\mathbf{W_z} e_{u,i} + \mathbf{U_z} h_{u,i-1}) \\ r_i &= \sigma(\mathbf{W_r} e_{u,i} + \mathbf{U_r} h_{u,i-1}) \\ \widetilde{h}_{u,i} &= tanh(\mathbf{W_h} e_{u,i} + \mathbf{U_h}(r_i \odot h_{u,i-1})) \\ h_{u,i} &= z_i \odot \widetilde{h}_{u,i} + (1 - z_i) \odot h_{u,i-1}, \end{aligned} \tag{2}$$

where $h_{u,0} = 0$, z_i and r_i are an update gate and a reset gate respectively, $\sigma(\cdot)$ is a sigmoid function, and $\mathbf{W_z}$, $\mathbf{W_h}$, $\mathbf{W_r}$, $\mathbf{U_z}$, $\mathbf{U_r}$,$\mathbf{U_h}$ are parameters. Similarly, we have $\mathbf{H}_r = [h_{r,1}, \ldots, h_{r,n_r}]$ as the hidden vectors of $\mathbf{R}$. Then, $\forall i, j$, the (i, j)-th element of $\mathbf{M}_2$ is defined by

$$e_{2,i,j} = h_{u,i}^{\top} \mathbf{A} h_{r,j}, \tag{3}$$

where $\mathbf{A} \in \mathbb{R}^{m \times m}$ is a linear transformation. $\forall i$, GRU models the sequential relationship and the dependency among words up to position i and encodes the text segment until the i-th word to a hidden vector. Therefore, $\mathbf{M}_2$ models the matching between u and r on a segment level.

$\mathbf{M}_1$ and $\mathbf{M}_2$ are then processed by a CNN to form v. $\forall f = 1, 2$, CNN regards $\mathbf{M}_f$ as an input channel, and alternates convolution and max-pooling operations. Suppose that $z^{(l,f)} = \left[z_{i,j}^{(l,f)}\right]_{I^{(l,f)} \times J^{(l,f)}}$ denotes the output of feature maps of type-f on layer-l, where $z^{(0,f)} = \mathbf{M}_f$, $\forall f = 1, 2$. On the convolution layer, we employ a 2D convolution operation with a window size $r_w^{(l,f)} \times r_h^{(l,f)}$, and define $z_{i,j}^{(l,f)}$ as

$$z_{i,j}^{(l,f)} = \sigma\Big(\sum_{f'=0}^{F_{l-1}} \sum_{s=0}^{r_w^{(l,f)}} \sum_{t=0}^{r_h^{(l,f)}} \mathbf{W}_{s,t}^{(l,f)} \cdot z_{i+s,j+t}^{(l-1,f')} + \mathbf{b}^{l,k}\Big), \tag{4}$$

where $\sigma(\cdot)$ is a ReLU, $\mathbf{W}^{(l,f)} \in \mathbb{R}^{r_w^{(l,f)} \times r_h^{(l,f)}}$ and $\mathbf{b}^{l,k}$ are parameters, and F_{l-1} is the number of feature maps on the $(l-1)$-th layer. A max pooling operation follows a convolution operation and can be formulated as

$$z_{i,j}^{(l,f)} = \max_{p_w^{(l,f)} > s \geq 0} \max_{p_h^{(l,f)} > t \geq 0} z_{i+s,j+t}, \tag{5}$$

where $p_w^{(l,f)}$ and $p_h^{(l,f)}$ are the width and the height of the 2D pooling respectively. The output of the final feature maps are concatenated and mapped to a low dimensional space with a linear transformation as the matching vector $v \in \mathbb{R}^q$.

According to Equation (1), (3), (4), and (5), we can see that by learning word embedding and parameters of GRU from training data, words or segments in an utterance that are useful for recognizing the appropriateness of a response may have high similarity with some words or segments in the response and result in high value areas in the similarity matrices. These areas will be transformed and selected by convolution and pooling operations and carry important information in the utterance to the matching vector. This is how our model identifies important information in context and leverage it in matching under the supervision of the response. We consider multiple channels because we want to capture important matching information on multiple levels of granularity of text.

3.4 Matching Accumulation

Suppose that $[v_1, \ldots, v_n]$ is the output of the first layer (corresponding to n pairs), at the second layer, a GRU takes $[v_1, \ldots, v_n]$ as an input and encodes the matching sequence into its hidden states $H_m = [h'_1, \ldots, h'_n] \in \mathbb{R}^{q \times n}$ with a detailed parameterization similar to Equation (2). This layer has two functions: (1) it models the dependency and the temporal relationship of utterances in the

context; (2) it leverages the temporal relationship to supervise the accumulation of the pair matching as a context based matching. Moreover, from Equation (2), we can see that the reset gate (i.e., r_i) and the update gate (i.e., z_i) control how much information from the previous hidden state and the current input flows to the current hidden state, thus important matching vectors (corresponding to important utterances) can be accumulated while noise in the vectors can be filtered out.

3.5 Matching Prediction and Learning

With $[h'_1, \ldots, h'_n]$, we define $g(s, r)$ as

$$g(s, r) = softmax(\mathbf{W_2} L[h'_1, \ldots, h'_n] + \mathbf{b_2}), \quad (6)$$

where $\mathbf{W_2}$ and $\mathbf{b_2}$ are parameters. We consider three parameterizations for $L[h'_1, \ldots, h'_n]$: (1) only the last hidden state is used. Then $L[h'_1, \ldots, h'_n] = h'_n$. (2) the hidden states are linearly combined. Then, $L[h'_1, \ldots, h'_n] = \sum_{i=1}^{n} w_i h'_i$, where $w_i \in \mathbb{R}$. (3) we follow (Yang et al., 2016) and employ an attention mechanism to combine the hidden states. Then, $L[h'_1, \ldots, h'_n]$ is defined as

$$\begin{aligned} t_i &= tanh(\mathbf{W_{1,1}} h_{u_i,n_u} + \mathbf{W_{1,2}} h'_i + \mathbf{b_1}), \\ \alpha_i &= \frac{exp(t_i^\top t_s)}{\sum_i (exp(t_i^\top t_s))}, \\ L[h'_1, \ldots, h'_n] &= \sum_{i=1}^{n} \alpha_i h'_i, \end{aligned} \quad (7)$$

where $\mathbf{W_{1,1}} \in \mathbb{R}^{q \times m}$, $\mathbf{W_{1,2}} \in \mathbb{R}^{q \times q}$ and $\mathbf{b_1} \in \mathbb{R}^q$ are parameters. h'_i and h_{u_i,n_u} are the i-th matching vector and the final hidden state of the i-th utterance respectively. $t_s \in \mathbb{R}^q$ is a virtual context vector which is randomly initialized and jointly learned in training.

Both (2) and (3) aim to learn weights for $\{h'_1, \ldots, h'_n\}$ from training data and highlight the effect of important matching vectors in the final matching. The difference is that weights in (2) are static, because the weights are totally determined by the positions of utterances, while weights in (3) are dynamically computed by the matching vectors and utterance vectors. We denote our model with the three parameterizations of $L[h'_1, \ldots, h'_n]$ as SMN$_{last}$, SMN$_{static}$, and SMN$_{dynamic}$, and empirically compare them in experiments.

We learn $g(\cdot, \cdot)$ by minimizing cross entropy with $\mathcal{D}$. Let Θ denote the parameters of SMN, then the objective function $\mathcal{L}(\mathcal{D}, \Theta)$ of learning can be formulated as

$$-\sum_{i=1}^{N} [y_i log(g(s_i, r_i)) + (1 - y_i) log(1 - g(s_i, r_i))]\,. \quad (8)$$

4 Response Candidate Retrieval

In practice, a retrieval-based chatbot, to apply the matching approach to the response selection, one needs to retrieve a number of response candidates from an index beforehand. While candidate retrieval is not the focus of the paper, it is an important step in a real system. In this work, we exploit a heuristic method to obtain response candidates from the index. Given a message u_n with $\{u_1, \ldots, u_{n-1}\}$ utterances in its previous turns, we extract the top 5 keywords from $\{u_1, \ldots, u_{n-1}\}$ based on their tf-idf scores[1] and expand u_n with the keywords. Then we send the expanded message to the index and retrieve response candidates using the inline retrieval algorithm of the index. Finally, we use $g(s, r)$ to rerank the candidates and return the top one as a response to the context.

5 Experiments

We tested our model on a publicly available English data set and a Chinese data set published with this paper.

5.1 Ubuntu Corpus

The English data set is the Ubuntu Corpus (Lowe et al., 2015) which contains multi-turn dialogues collected from chat logs of the Ubuntu Forum. The data set consists of 1 million context-response pairs for training, 0.5 million pairs for validation, and 0.5 million pairs for testing. Positive responses are true responses from humans, and negative ones are randomly sampled. The ratio of the positive and the negative is 1:1 in training, and 1:9 in validation and testing. We used the copy shared by Xu et al. (2016) [2] in which numbers, urls, and paths are replaced by special placeholders. We followed (Lowe et al., 2015) and employed recall at position k in n candidates ($R_n@k$) as evaluation metrics.

[1] Tf is word frequency in the context, while idf is calculated using the entire index.

[2] https://www.dropbox.com/s/2fdn26rj6h9bpvl/ubuntudata.zip?dl=0

5.2 Douban Conversation Corpus

The Ubuntu Corpus is a domain specific data set, and response candidates are obtained from negative sampling without human judgment. To further verify the efficacy of our model, we created a new data set with open domain conversations, called the Douban Conversation Corpus. Response candidates in the test set of the Douban Conversation Corpus are collected following the procedure of a retrieval-based chatbot and are labeled by human judges. It simulates the real scenario of a retrieval-based chatbot. We publish it to research communities to facilitate the research of multi-turn response selection.

Specifically, we crawled 1.1 million dyadic dialogues (conversation between two persons) longer than 2 turns from Douban group[3] which is a popular social networking service in China. We randomly sampled 0.5 million dialogues for creating a training set, 25 thousand dialouges for creating a validation set, and $1,000$ dialogues for creating a test set, and made sure that there is no overlap between the three sets. For each dialogue in training and validation, we took the last turn as a positive response for the previous turns as a context and randomly sampled another response from the 1.1 million data as a negative response. There are 1 million context-response pairs in the training set and 50 thousand pairs in the validation set.

To create the test set, we first crawled 15 million post-reply pairs from Sina Weibo[4] which is the largest microblogging service in China and indexed the pairs with Lucene[5]. We took the last turn of each Douban dyadic dialogue in the test set as a message, retrieved 10 response candidates from the index following the method in Section 4, and finally formed a test set with $10,000$ context-response pairs. We recruited three labelers to judge if a candidate is a proper response to the context. A proper response means the response can naturally reply to the message given the whole context. Each pair received three labels and the majority of the labels were taken as the final decision. Table 2 gives the statistics of the three sets. Note that the Fleiss' kappa (Fleiss, 1971) of the labeling is 0.41, which indicates that the three labelers reached a relatively high agreement.

Besides $R_n@k$s, we also followed the convention of information retrieval and employed mean average precision (MAP) (Baeza-Yates et al., 1999), mean reciprocal rank (MRR) (Voorhees et al., 1999), and precision at position 1 (P@1) as evaluation metrics. We did not calculate $R_2@1$ because in Douban corpus one context could have more than one correct responses, and we have to randomly sample one for $R_2@1$, which may bring bias to evaluation. When using the labeled set, we removed conversations with all negative responses or all positive responses, as models make no difference with them. There are $6,670$ context-response pairs left in the test set.

	train	val	test
# context-response pairs	1M	50k	10k
# candidates per context	2	2	10
# positive candidates per context	1	1	1.18
Min. # turns per context	3	3	3
Max. # turns per context	98	91	45
Avg. # turns per context	6.69	6.75	6.45
Avg. # words per utterance	18.56	18.50	20.74

Table 2: Statistics of Douban Conversation Corpus

5.3 Baseline

We considered the following baselines:

Basic models: models in (Lowe et al., 2015) and (Kadlec et al., 2015) including TF-IDF, RNN, CNN, LSTM and BiLSTM.

Multi-view: the model proposed by Zhou et al. (2016) that utilizes a hierarchical recurrent neural network to model utterance relationships.

Deep learning to respond (DL2R): the model proposed by Yan et al. (2016) that reformulates the message with other utterances in the context.

Advanced single-turn matching models: since BiLSTM does not represent the state-of-the-art matching model, we concatenated the utterances in a context and matched the long text with a response candidate using more powerful models including MV-LSTM (Wan et al., 2016) (2D matching), Match-LSTM (Wang and Jiang, 2015), Attentive-LSTM (Tan et al., 2015) (two attention based models), and Multi-Channel which is described in Section 3.3. Multi-Channel is a simple version of our model without considering utterance relationships. We also appended the top 5 tf-idf words in context to the input message, and computed the score between the expanded message and a response with Multi-Channel, denoted as Multi-Channel$_{exp}$.

[3] https://www.douban.com/group
[4] http://weibo.com/
[5] https://lucenenet.apache.org/

	Ubuntu Corpus				Douban Conversation Corpus					
	R_2@1	R_{10}@1	R_{10}@2	R_{10}@5	MAP	MRR	P@1	R_{10}@1	R_{10}@2	R_{10}@5
TF-IDF	0.659	0.410	0.545	0.708	0.331	0.359	0.180	0.096	0.172	0.405
RNN	0.768	0.403	0.547	0.819	0.390	0.422	0.208	0.118	0.223	0.589
CNN	0.848	0.549	0.684	0.896	0.417	0.440	0.226	0.121	0.252	0.647
LSTM	0.901	0.638	0.784	0.949	0.485	0.527	0.320	0.187	0.343	0.720
BiLSTM	0.895	0.630	0.780	0.944	0.479	0.514	0.313	0.184	0.330	0.716
Multi-View	0.908	0.662	0.801	0.951	0.505	0.543	0.342	0.202	0.350	0.729
DL2R	0.899	0.626	0.783	0.944	0.488	0.527	0.330	0.193	0.342	0.705
MV-LSTM	0.906	0.653	0.804	0.946	0.498	0.538	0.348	0.202	0.351	0.710
Match-LSTM	0.904	0.653	0.799	0.944	0.500	0.537	0.345	0.202	0.348	0.720
Attentive-LSTM	0.903	0.633	0.789	0.943	0.495	0.523	0.331	0.192	0.328	0.718
Multi-Channel	0.904	0.656	0.809	0.942	0.506	0.543	0.349	0.203	0.351	0.709
Multi-Channel$_{exp}$	0.714	0.368	0.497	0.745	0.476	0.515	0.317	0.179	0.335	0.691
SMN$_{last}$	**0.923**	**0.723**	**0.842**	**0.956**	**0.526**	**0.571**	**0.393**	**0.236**	**0.387**	0.729
SMN$_{static}$	**0.927**	**0.725**	**0.838**	**0.962**	**0.523**	**0.572**	**0.387**	**0.228**	**0.387**	0.734
SMN$_{dynamic}$	**0.926**	**0.726**	**0.847**	**0.961**	**0.529**	**0.569**	**0.397**	**0.233**	**0.396**	0.724

Table 3: Evaluation results on the two data sets. Numbers in bold mean that the improvement is statistically significant compared with the best baseline.

5.4 Parameter Tuning

For baseline models, if their results are available in existing literature (e.g., those on the Ubuntu corpus), we just copied the numbers, otherwise we implemented the models following the settings in the literatures. All models were implemented using Theano (Theano Development Team, 2016). Word embeddings were initialized by the results of word2vec (Mikolov et al., 2013) which ran on the training data, and the dimensionality of word vectors is 200. For Multi-Channel and layer one of our model, we set the dimensionality of the hidden states of GRU as 200. We tuned the window size of convolution and pooling in $\{(2,2),(3,3)(4,4)\}$ and chose $(3,3)$ finally. The number of feature maps is 8. In layer two, we set the dimensionality of matching vectors and the hidden states of GRU as 50. The parameters were updated by stochastic gradient descent with Adam algorithm (Kingma and Ba, 2014) on a single Tesla K80 GPU. The initial learning rate is 0.001, and the parameters of Adam, β_1 and β_2 are 0.9 and 0.999 respectively. We employed early-stopping as a regularization strategy. Models were trained in mini-batches with a batch size of 200, and the maximum utterance length is 50. We set the maximum context length (i.e., number of utterances) as 10, because the performance of models does not improve on contexts longer than 10 (details are shown in the Section 5.6). We padded zeros if the number of utterances in a context is less than 10, otherwise we kept the last 10 utterances.

5.5 Evaluation Results

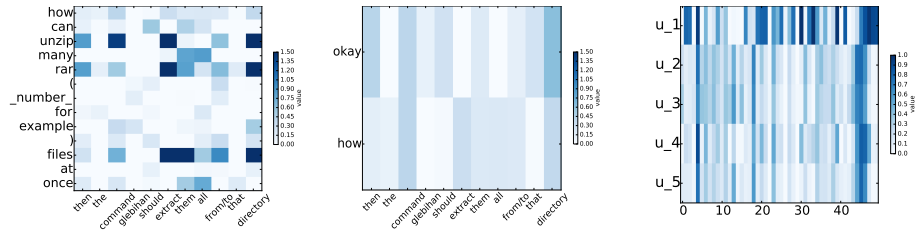
Table 3 shows the evaluation results on the two data sets. Our models outperform baselines greatly in terms of all metrics on both data sets, with the improvements being statistically significant (t-test with p-value ≤ 0.01, except R_{10}@5 on Douban Corpus). Even the state-of-the-art single-turn matching models perform much worse than our models. The results demonstrate that one cannot neglect utterance relationships and simply perform multi-turn response selection by concatenating utterances together. Our models achieve significant improvements over Multi-View, which justified our "matching first" strategy. DL2R is worse than our models, indicating that utterance reformulation with heuristic rules is not a good method for utilizing context information. R_n@ks are low on the Douban Corpus as there are multiple correct candidates for a context (e.g., if there are 3 correct responses, then the maximum R_{10}@1 is 0.33). SMN$_{dynamic}$ is only slightly better than SMN$_{static}$ and SMN$_{last}$. The reason might be that the GRU can select useful signals from the matching sequence and accumulate them in the final state with its gate mechanism, thus the efficacy of an attention mechanism is not obvious for the task at hand.

5.6 Further Analysis

Visualization: we visualize the similarity matrices and the gates of GRU in layer two using an example from the Ubuntu corpus to further clarify how our model identifies important information in the context and how it selects important matching vectors with the gate mechanism of GRU as described in Section 3.3 and Section 3.4. The example is $\{u_1$*: how can unzip many rar (* $\_number\_$ *for example) files at once;* u_2*: sure you can do that in bash;* u_3*: okay how?* u_4*: are the files all*

	Ubuntu Corpus				Douban Conversation Corpus					
	$R_2@1$	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	MAP	MRR	P@1	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
Replace$_M$	0.905	0.661	0.799	0.950	0.503	0.541	0.343	0.201	0.364	0.729
Replace$_A$	0.918	0.716	0.832	0.954	0.522	0.565	0.376	0.220	0.385	0.727
Only M_1	0.919	0.704	0.832	0.955	0.518	0.562	0.370	0.228	0.371	0.737
Only M_2	0.921	0.715	0.836	0.956	0.521	0.565	0.382	0.232	0.380	0.734
SMN$_{last}$	0.923	0.723	0.842	0.956	0.526	0.571	0.393	0.236	0.387	0.729

Table 4: Evaluation results of model ablation.



(a) $\mathbf{M}_1$ of u_1 and r (b) $\mathbf{M}_1$ of u_3 and r (c) Update gate

Figure 2: Model visualization. Darker areas mean larger value.

in the same directory? u_5: yes they all are; r: then the command glebihan should extract them all from/to that directory}. It is from the test set and our model successfully ranked the correct response to the top position. Due to space limitation, we only visualized $\mathbf{M}_1$, $\mathbf{M}_2$ and the update gate (i.e. z) in Figure 2. We can see that in u_1 important words including "unzip", "rar", "files" are recognized and carried to matching by "command", "extract", and "directory" in r, while u_3 is almost useless and thus little information is extracted from it. u_1 is crucial to response selection and nearly all information from u_1 and r flows to the hidden state of GRU, while other utterances are less informative and the corresponding gates are almost "closed" to keep the information from u_1 and r until the final state.

Model ablation: we investigate the effect of different parts of SMN by removing them one by one from SMN$_{last}$, shown in Table 4. First, replacing the multi-channel "2D" matching with a neural tensor network (NTN) (Socher et al., 2013) (denoted as Replace$_M$) makes the performance drop dramatically. This is because NTN only matches a pair by an utterance vector and a response vector and loses important information in the pair. Together with the visualization, we can conclude that "2D" matching plays a key role in the "matching first" strategy as it captures the important matching information in each pair with minimal loss. Second, the performance drops slightly when replacing the GRU for matching accumulation with a multi-layer perceptron (denoted as Replace$_A$). This indicates that utterance relationships are useful. Finally, we left only one channel in matching and found that $\mathbf{M}_2$ is a little more powerful than $\mathbf{M}_1$ and we achieve the best results with both of them (except on $R_{10}@5$ on the Douban Corpus).

Performance across context length: we study how our model (SMN$_{last}$) performs across the length of contexts. Figure 3 shows the comparison on MAP in different length intervals on the Douban corpus. Our model consistently performs better than the baselines, and when contexts become longer, the gap becomes larger. The results demonstrate that our model can well capture the dependencies, especially long dependencies, among utterances in contexts.

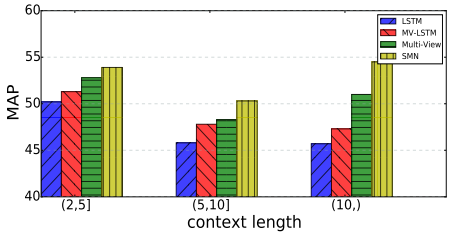
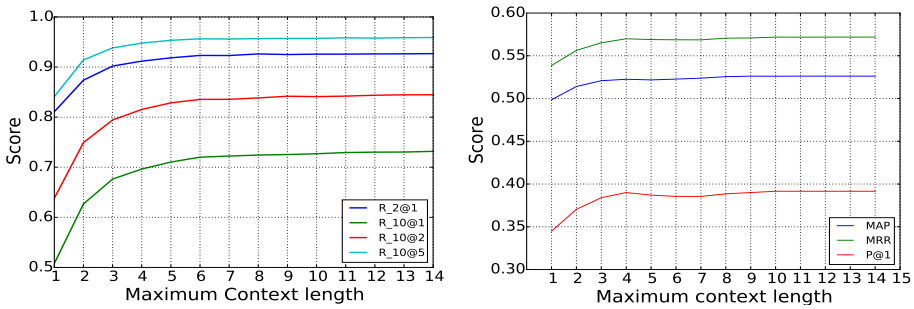


Figure 3: Comparison across context length

Maximum context length: we investigate the influence of maximum context length for SMN. Figure 4 shows the performance of SMN on Ubuntu Corpus and Douban Corpus with respect to maximum context length. From Figure 4, we find that performance improves significantly when the maximum context length is lower than 5, and becomes stable after the context length reaches 10. This indicates that context information is important for multi-turn response selection, and we can set the maximum context length as 10 to balance effectiveness and efficiency.

Error analysis: although SMN outperforms baseline methods on the two data sets, there are



(a) Ubuntu Corpus (b) Douban Conversation Corpus

Figure 4: Performance of SMN across maximum context length

still several problems that cannot be handled perfectly.

(1) Logical consistency. SMN models the context and response on the semantic level, but pays little attention to logical consistency. This leads to several DSATs in the Douban Corpus. For example, given a context {*a: Does anyone know Newton jogging shoes? b: 100 RMB on Taobao. a: I know that. I do not want to buy it because that is a fake which is made in Qingdao ,b: Is it the only reason you do not want to buy it?* }, SMN gives a large score to the response { *It is not a fake. I just worry about the date of manufacture*}. The response is inconsistent with the context on logic, as it claims that the jogging shoes are not fake. In the future, we shall explore the logic consistency problem in retrieval-based chatbots.

(2) No correct candidates after retrieval. In the experiment, we prepared 1000 contexts for testing, but only 667 contexts have correct candidates after candidate response retrieval. This indicates that there is still room for candidate retrieval components to improve, and only expanding the input message with several keywords in context may not be a perfect approach for candidate retrieval. In the future, we will consider advanced methods for retrieving candidates.

6 Conclusion and Future Work

We present a new context based model for multi-turn response selection in retrieval-based chatbots. Experiment results on open data sets show that the model can significantly outperform the state-of-the-art methods. Besides, we publish the first human-labeled multi-turn response selection data set to research communities. In the future, we shall study how to model logical consistency of responses and improve candidate retrieval.

7 Acknowledgment

We appreciate valuable comments provided by anonymous reviewers and our discussions with Zhao Yan. This work was supported by the National Natural Science Foundation of China (Grand Nos. 61672081, U1636211, 61370126), Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001), National High Technology Research and Development Program of China (No.2015AA016004), and the Fund of the State Key Laboratory of Software Development Environment (No.SKLSDE-2015ZX-16).

References

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*, volume 463. ACM press New York.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76(5):378.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*. pages 2042–2050.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988* .

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv:1510.03753* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155* .

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 583–593.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808* .

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville. 2016. Multiresolution recurrent neural networks: An application to dialogue response generation. *arXiv preprint arXiv:1606.00776* .

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. pages 1577–1586.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*. pages 926–934.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108* .

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*. volume 99, pages 77–82.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-srnn: Modeling the recursive matching structure with spatial rnn. *arXiv preprint arXiv:1604.04378* .

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *EMNLP*. pages 935–945.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. *arXiv preprint arXiv:1503.02427* .

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849* .

Bowen Wu, Baoxun Wang, and Hui Xue. 2016a. Ranking responses oriented to conversational relevance in chat-bots. *COLING16* .

Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2016b. Topic augmented neural network for short text conversation. *CoRR* abs/1605.00090.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2016. Topic augmented neural response generation with a joint attention mechanism. *arXiv preprint arXiv:1606.08340* .

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2016. Incorporating loose-structured knowledge into lstm with recall gate for conversation modeling. *arXiv preprint arXiv:1605.05110* .

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *SIGIR 2016, Pisa, Italy, July 17-21, 2016*. pages 55–64. https://doi.org/10.1145/2911451.2911542.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language* 24(2):150–174.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, R Yan, D Yu, Xuan Liu, and H Tian. 2016. Multi-view response selection for human-computer conversation. *EMNLP16* .

Learning word-like units from joint audio-visual analysis

David Harwath and James Glass
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{dharwath,glass}@mit.edu

Abstract

Given a collection of images and spoken audio captions, we present a method for discovering word-like acoustic units in the continuous speech signal and grounding them to semantically relevant image regions. For example, our model is able to detect spoken instances of the words "lighthouse" within an utterance and associate them with image regions containing lighthouses. We do not use any form of conventional automatic speech recognition, nor do we use any text transcriptions or conventional linguistic annotations. Our model effectively implements a form of spoken language acquisition, in which the computer learns not only to recognize word categories by sound, but also to enrich the words it learns with semantics by grounding them in images.

1 Introduction

1.1 Problem Statement and Motivation

Automatically discovering words and other elements of linguistic structure from continuous speech has been a longstanding goal in computational linguists, cognitive science, and other speech processing fields. Practically all humans acquire language at a very early age, but this task has proven to be an incredibly difficult problem for computers. While conventional automatic speech recognition (ASR) systems have a long history and have recently made great strides thanks to the revival of deep neural networks (DNNs), their reliance on highly supervised training paradigms has essentially restricted their application to the major languages of the world, accounting for a small fraction of the more than 7,000 human languages spoken worldwide (Lewis et al., 2016). The main reason for this limitation is the fact that these supervised approaches require enormous amounts of very expensive human transcripts. Moreover, the use of the written word is a convenient but limiting convention, since there are many oral languages which do not even employ a writing system. In constrast, infants learn to communicate verbally before they are capable of reading and writing - so there is no inherent reason why spoken language systems need to be inseparably tied to text.

The key contribution of this paper has two facets. First, we introduce a methodology capable of not only discovering word-like units from continuous speech at the waveform level with no additional text transcriptions or conventional speech recognition apparatus. Instead, we jointly learn the semantics of those units via visual associations. Although we evaluate our algorithm on an English corpus, it could conceivably run on any language without requiring any text or associated ASR capability. Second, from a computational perspective, our method of speech pattern discovery runs in linear time. Previous work has presented algorithms for performing acoustic pattern discovery in continuous speech (Park and Glass, 2008; Jansen et al., 2010; Jansen and Van Durme, 2011) without the use of transcriptions or another modality, but those algorithms are limited in their ability to scale by their inherent $O(n^2)$ complexity, since they do an exhaustive comparison of the data against itself. Our method leverages correlated information from a second modality - the visual domain - to guide the discovery of words and phrases. This enables our method to run in $O(n)$ time, and we demonstrate it scalability by discovering acoustic patterns in over 522 hours of audio.

1.2 Previous Work

A sub-field within speech processing that has garnered much attention recently is unsupervised

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 506–517
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1047

speech pattern discovery. Segmental Dynamic Time Warping (S-DTW) was introduced by Park and Glass (2008), which discovers repetitions of the same words and phrases in a collection of untranscribed acoustic data. Many subsequent efforts extended these ideas (Jansen et al., 2010; Jansen and Van Durme, 2011; Dredze et al., 2010; Harwath et al., 2012; Zhang and Glass, 2009). Alternative approaches based on Bayesian nonparametric modeling (Lee and Glass, 2012; Ondel et al., 2016) employed a generative model to cluster acoustic segments into phoneme-like categories, and related works aimed to segment and cluster either reference or learned phoneme-like tokens into higher-level units (Johnson, 2008; Goldwater et al., 2009; Lee et al., 2015).

While supervised object detection is a standard problem in the vision community, several recent works have tackled the problem of weakly-supervised or unsupervised object localization (Bergamo et al., 2014; Cho et al., 2015; Zhou et al., 2015; Cinbis et al., 2016). Although the focus of this work is discovering acoustic patterns, in the process we jointly associate the acoustic patterns with clusters of image crops, which we demonstrate capture visual patterns as well.

The computer vision and NLP communities have begun to leverage deep learning to create multimodal models of images and text. Many works have focused on generating annotations or text captions for images (Socher and Li, 2010; Frome et al., 2013; Socher et al., 2014; Karpathy et al., 2014; Karpathy and Li, 2015; Vinyals et al., 2015; Fang et al., 2015; Johnson et al., 2016). One interesting intersection between word induction from phoneme strings and multimodal modeling of images and text is that of Gelderloos and Chrupaa (2016), who uses images to segment words within captions at the phoneme string level. Other work has taken these ideas beyond text, and attempted to relate images to spoken audio captions directly at the waveform level (Roy, 2003; Harwath and Glass, 2015; Harwath et al., 2016). The work of (Harwath et al., 2016) is the most similar to ours, in which the authors learned embeddings at the entire image and entire spoken caption level and then used the embeddings to perform bidirectional retrieval. In this work, we go further by automatically segmenting and clustering the spoken captions into individual word-like units, as well as the images into object-like categories.

2 Experimental Data

We employ a corpus of over 200,000 spoken captions for images taken from the Places205 dataset (Zhou et al., 2014), corresponding to over 522 hours of speech data. The captions were collected using Amazon's Mechanical Turk service, in which workers were shown images and asked to describe them verbally in a free-form manner. The data collection scheme is described in detail in Harwath et al. (2016), but the experiments in this paper leverage nearly twice the amount of data. For training our multimodal neural network as well as the pattern discovery experiments, we use a subset of 214,585 image/caption pairs, and we hold out a set of 1,000 pairs for evaluating the multimodal network's retrieval ability. Because we lack ground truth text transcripts for the data, we used Google's Speech Recognition public API to generate proxy transcripts which we use when analyzing our system. Note that the ASR was only used for analysis of the results, and was not involved in any of the learning.

3 Audio-Visual Embedding Neural Networks

We first train a deep multimodal embedding network similar in spirit to the one described in Harwath et al. (2016), but with a more sophisticated architecture. The model is trained to map entire image frames and entire spoken captions into a shared embedding space; however, as we will show, the trained network can then be used to localize patterns corresponding to words and phrases within the spectrogram, as well as visual objects within the image by applying it to small sub-regions of the image and spectrogram. The model is comprised of two branches, one which takes as input images, and the other which takes as input spectrograms. The image network is formed by taking the off-the-shelf VGG 16 layer network (Simonyan and Zisserman, 2014) and replacing the softmax classification layer with a linear transform which maps the 4096-dimensional activations of the second fully connected layer into our 1024-dimensional multimodal embedding space. In our experiments, the weights of this projection layer are trained, but the layers taken from the VGG network below it are kept fixed. The second branch of our network analyzes speech spectrograms as if they were black and white images. Our spectrograms are computed using 40 log Mel

filterbanks with a 25ms Hamming window and a 10ms shift. The input to this branch always has 1 color channel and is always 40 pixels high (corresponding to the 40 Mel filterbanks), but the width of the spectrogram varies depending upon the duration of the spoken caption, with each pixel corresponding to approximately 10 milliseconds worth of audio. The architecture we use is entirely convolutional and shown below, where C denotes the number of convolutional channels, W is filter width, H is filter height, and S is pooling stride.

1. Convolution: C=128, W=1, H=40, ReLU
2. Convolution: C=256, W=11, H=1, ReLU
3. Maxpool: W=3, H=1, S=2
4. Convolution: C=512, W=17, H=1, ReLU
5. Maxpool: W=3, H=1, S=2
6. Convolution: C=512, W=17, H=1, ReLU
7. Maxpool: W=3, H=1, S=2
8. Convolution: C=1024, W=17, H=1, ReLU
9. Meanpool over entire caption
10. L2 normalization

In practice during training, we restrict the caption spectrograms to all be 1024 frames wide (i.e., 10sec of speech) by applying truncation or zero padding. Additionally, both the images and spectrograms are mean normalized before training. The overall multimodal network is formed by tying together the image and audio branches with a layer which takes both of their output vectors and computes an inner product between them, representing the similarity score between a given image/caption pair. We train the network to assign high scores to matching image/caption pairs, and lower scores to mismatched pairs.

Within a minibatch of B image/caption pairs, let $S_j^p, j = 1, \ldots, B$ denote the similarity score of the j^{th} image/caption pair as output by the neural network. Next, for each pair we randomly sample one impostor caption and one impostor image from the same minibatch. Let S_j^i denote the similarity score between the j^{th} caption and its impostor image, and S_j^c be the similarity score between the j^{th} image and its impostor caption. The total loss for the entire minibatch is then computed as

$$\mathcal{L}(\theta) = \sum_{j=1}^{B} [\max(0, S_j^c - S_j^p + 1) + \max(0, S_j^i - S_j^p + 1)] \quad (1)$$

We train the neural network with 50 epochs of stochastic gradient descent using a batch size $B = 128$, a momentum of 0.9, and a learning rate of 1e-5 which is set to geometrically decay by a factor between 2 and 5 every 5 to 10 epochs.

4 Finding and Clustering Audio-Visual Caption Groundings

Although we have trained our multimodal network to compute embeddings at the granularity of entire images and entire caption spectrograms, we can easily apply it in a more localized fashion. In the case of images, we can simply take any arbitrary crop of an original image and resize it to 224x224 pixels. The audio network is even more trivial to apply locally, because it is entirely convolutional and the final mean pooling layer ensures that the output will be a 1024-dim vector no matter the extent of the input. The bigger question is *where* to locally apply the networks in order to discover meaningful acoustic and visual patterns.

Given an image and its corresponding spoken audio caption, we use the term grounding to refer to extracting meaningful segments from the caption and associating them with an appropriate subregion of the image. For example, if an image depicted a person eating ice cream and its caption contained the spoken words "A person is enjoying some ice cream," an ideal set of groundings would entail the acoustic segment containing the word "person" linked to a bounding box around the person, and the segment containing the word "ice cream" linked to a box around the ice cream. We use a constrained brute force ranking scheme to evaluate all possible groundings (with a restricted granularity) between an image and its caption. Specifically, we divide the image into a grid, and extract all of the image crops whose boundaries sit on the grid lines. Because we are mainly interested in extracting regions of interest and not high precision object detection boxes, to keep the number of proposal regions under control we impose several restrictions. First, we use a 10x10 grid on each image regardless of its original size. Second, we define minimum and maximum aspect ratios as 2:3 and 3:2 so as not to introduce too much distortion and also to reduce the number of proposal boxes. Third, we define a minimum bounding width as 30% of the original image width, and similarly a minimum height as 30% of the original image height. In practice, this results in a few thousand proposal regions per image.

To extract proposal segments from the audio

caption spectrogram, we similarly define a 1-dim grid along the time axis, and consider all possible start/end points at 10 frame (pixel) intervals. We impose minimum and maximum segment length constraints at 50 and 100 frames (pixels), implying that our discovered acoustic patterns are restricted to fall between 0.5 and 1 second in duration. The number of proposal segments will vary depending on the caption length, and typically number in the several thousands. Note that when learning groundings we consider the entire audio sequence, and do not incorporate the 10sec duration constraint imposed during training.

Once we have extracted a set of proposed visual bounding boxes and acoustic segments for a given image/caption pair, we use our multimodal network to compute a similarity score between each unique image crop/acoustic segment pair. Each triplet of an image crop, acoustic segment, and similarity score constitutes a proposed grounding. A naive approach would be to simply keep the top N groundings from this list, but in practice we ran into two problems with this strategy. First, many proposed acoustic segments capture mostly silence due to pauses present in natural speech. We solve this issue by using a simple voice activity detector (VAD) which was trained on the TIMIT corpus(Garofolo et al., 1993). If the VAD estimates that 40% or more of any proposed acoustic segment is silence, we discard that entire grounding. The second problem we ran into is the fact that the top of the sorted grounding list is dominated by highly overlapping acoustic segments. This makes sense, because highly informative content words will show up in many different groundings with slightly perturbed start or end times. To alleviate this issue, when evaluating a grounding from the top of the proposal list we compare the interval intersection over union (IOU) of its acoustic segment against all acoustic segments already accepted for further consideration. If the IOU exceeds a threshold of 0.1, we discard the new grounding and continue moving down the list. We stop accumulating groundings once the scores fall to below 50% of the top score in the "keep" list, or when 10 groundings have been added to the "keep" list. Figure 1 displays a pictorial example of our grounding procedure.

Once we have completed the grounding procedure, we are left with a small set of regions of interest in each image and caption spectrogram. We use the respective branches of our multimodal network to compute embedding vectors for each grounding's image crop and acoustic segment. We then employ k-means clustering separately on the collection of image embedding vectors as well as the collection of acoustic embedding vectors. The last step is to establish an affinity score between each image cluster $\mathcal{I}$ and each acoustic cluster $\mathcal{A}$; we do so using the equation

$$\text{Affinity}(\mathcal{I}, \mathcal{A}) = \sum_{\mathbf{i} \in \mathcal{I}} \sum_{\mathbf{a} \in \mathcal{A}} \mathbf{i}^\top \mathbf{a} \cdot \text{Pair}(\mathbf{i}, \mathbf{a}) \quad (2)$$

where $\mathbf{i}$ is an image crop embedding vector, $\mathbf{a}$ is an acoustic segment embedding vector, and $\text{Pair}(\mathbf{i}, \mathbf{a})$ is equal to 1 when $\mathbf{i}$ and $\mathbf{a}$ belong to the same grounding pair, and 0 otherwise. After clustering, we are left with a set of acoustic pattern clusters, a set of visual pattern clusters, and a set of linkages describing which acoustic clusters are associated with which image clusters. In the next section, we investigate these clusters in more detail.

5 Experiments and Analysis

Table 1: Results for image search and annotation on the Places audio caption data (214k training pairs, 1k testing pairs). Recall is shown for the top 1, 5, and 10 hits. The model we use in this paper is compared against the meanpool variant of the model architecture presented in Harwath et al. (2016). For both training and testing, the captions were truncated/zero-padded to 10 seconds.

	Search		
Model	R@1	R@5	R@10
(Harwath et al., 2016)	0.090	0.261	0.372
This work (audio)	0.112	0.312	0.431
This work (text)	0.111	0.383	0.525
	Annotation		
Model	R@1	R@5	R@10
(Harwath et al., 2016)	0.098	0.266	0.352
This work (audio)	0.120	0.307	0.438
This work (text)	0.113	0.341	0.493

We trained our multimodal network on a set of 214,585 image/caption pairs, and vetted it with an image search (given caption, find image) and annotation (given image, find caption) task similar to the one used in Harwath et al. (2016); Karpathy et al. (2014); Karpathy and Li (2015). The image annotation and search recall scores on a 1,000 image/caption pair held-out test set are shown in Table 1. Also shown in this table are the scores

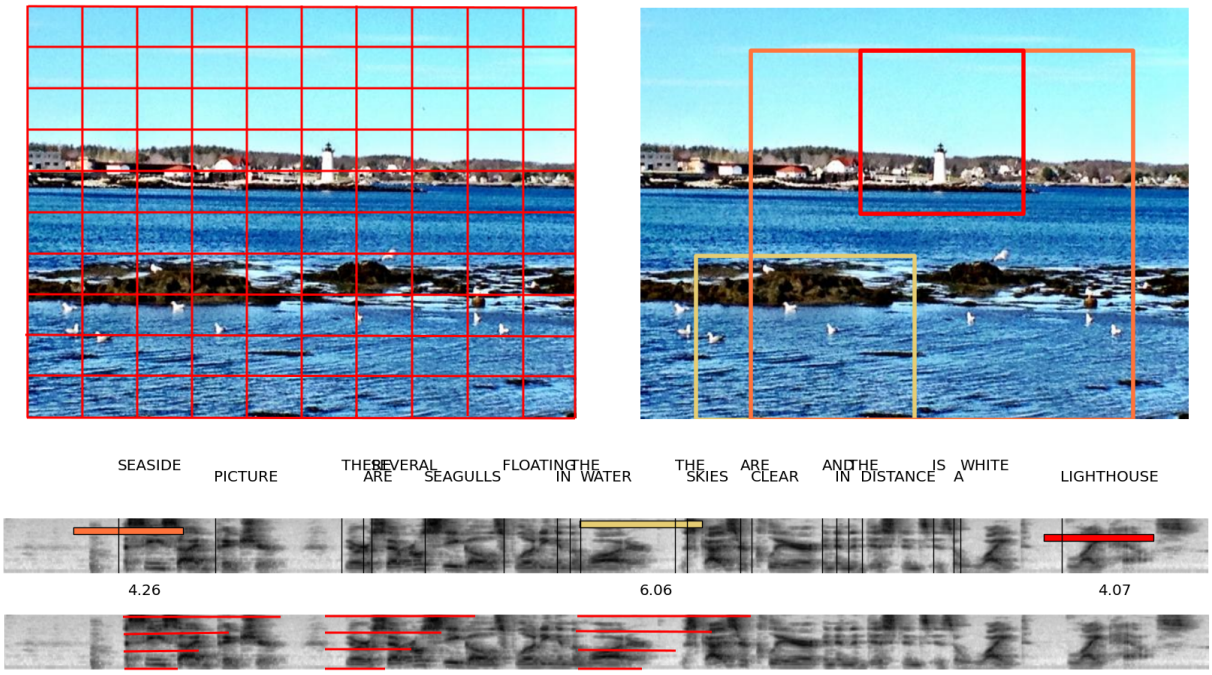


Figure 1: An example of our grounding method. The left image displays a grid defining the allowed start and end coordinates for the bounding box proposals. The bottom spectrogram displays several audio region proposals drawn as the families of stacked red line segments. The image on the right and spectrogram on the top display the final output of the grounding algorithm. The top spectrogram also displays the time-aligned text transcript of the caption, so as to demonstrate which words were captured by the groundings. In this example, the top 3 groundings have been kept, with the colors indicating the audio segment which is grounded to each bounding box.

Word	Count	Word	Count
ocean	2150	castle	766
(silence)	127	(silence)	70
the ocean	72	capital	39
blue ocean	29	large castle	24
body ocean	22	castles	23
oceans	16	(noise)	21
ocean water	16	council	13
(noise)	15	stone castle	12
of ocean	14	capitol	10
oceanside	14	old castle	10

Table 2: Examples of the breakdown of word/phrase identities of several acoustic clusters

achieved by a model which uses the ASR text transcriptions for each caption instead of the speech audio. The text captions were truncated/padded to 20 words, and the audio branch of the network was replaced with a branch with the following architecture:

1. Word embedding layer of dimension 200
2. Temporal Convolution: C=512, W=3, ReLU
3. Temporal Convolution: C=1024, W=3
4. Meanpool over entire caption
5. L2 normalization

One would expect that access to ASR hypotheses should improve the recall scores, but the performance gap is not enormous. Access to the ASR hypotheses provides a relative improvement of approximately 21.8% for image search R@10 and 12.5% for annotation R@10 compared to using no transcriptions or ASR whatsoever.

We performed the grounding and pattern clustering steps on the entire training dataset, which resulted in a total of 1,161,305 unique grounding pairs. For evaluation, we wish to assign a label to each cluster and cluster member, but this is not completely straightforward since each acoustic segment may capture part of a word, a whole word, multiple words, etc. Our strategy is to force-align the Google recognition hypothesis text to the audio, and then assign a label string to each acoustic segment based upon which words it overlaps in time. The alignments are created with the help of a Kaldi (Povey et al., 2011) speech recognizer

Table 3: Top 50 clusters with $k = 500$ sorted by increasing variance. Legend: $|C_c|$ is acoustic cluster size, $|C_i|$ is associated image cluster size, Pur. is acoustic cluster purity, σ^2 is acoustic cluster variance, and Cov. is acoustic cluster coverage. A dash (-) indicates a cluster whose majority label is silence.

Trans	$\|C_c\|$	$\|C_i\|$	Pur.	σ^2	Cov.	Trans	$\|C_c\|$	$\|C_i\|$	Pur.	σ^2	Cov.
-	1059	3480	0.70	0.26	-	snow	4331	3480	0.85	0.26	0.45
desert	1936	2896	0.82	0.27	0.67	kitchen	3200	2990	0.88	0.28	0.76
restaurant	1921	2536	0.89	0.29	0.71	mountain	4571	2768	0.86	0.30	0.38
black	4369	2387	0.64	0.30	0.17	skyscraper	843	3205	0.84	0.30	0.84
bridge	1654	2025	0.84	0.30	0.25	tree	5303	3758	0.90	0.30	0.16
castle	1298	2887	0.72	0.31	0.74	bridge	2779	2025	0.81	0.32	0.41
-	2349	2165	0.31	0.33	-	ocean	2913	3505	0.87	0.33	0.71
table	3765	2165	0.94	0.33	0.23	windmill	1458	3752	0.71	0.33	0.76
window	1890	2795	0.85	0.34	0.21	river	2643	3204	0.76	0.35	0.62
water	5868	3204	0.90	0.35	0.27	beach	1897	2964	0.79	0.35	0.64
flower	3906	2587	0.92	0.35	0.67	wall	3158	3636	0.84	0.35	0.23
sky	4306	6055	0.76	0.36	0.34	street	2602	2385	0.86	0.36	0.49
golf course	1678	3864	0.44	0.36	0.63	field	3896	3261	0.74	0.36	0.37
tree	4098	3758	0.89	0.36	0.13	lighthouse	1254	1518	0.61	0.36	0.83
forest	1752	3431	0.80	0.37	0.56	church	2503	3140	0.86	0.37	0.72
people	3624	2275	0.91	0.37	0.14	baseball	2777	1929	0.66	0.37	0.86
field	2603	3922	0.74	0.37	0.25	car	3442	2118	0.79	0.38	0.27
people	4074	2286	0.92	0.38	0.17	shower	1271	2206	0.74	0.38	0.82
people walking	918	2224	0.63	0.38	0.25	wooden	3095	2723	0.63	0.38	0.28
mountain	3464	3239	0.88	0.38	0.29	tree	3676	2393	0.89	0.39	0.11
-	1976	3158	0.28	0.39	-	snow	2521	3480	0.79	0.39	0.24
water	3102	2948	0.90	0.39	0.14	rock	2897	2967	0.76	0.39	0.26
-	2918	3459	0.08	0.39	-	night	3027	3185	0.44	0.39	0.59
station	2063	2083	0.85	0.39	0.62	chair	2589	2288	0.89	0.39	0.22
building	6791	3450	0.89	0.40	0.21	city	2951	3190	0.67	0.40	0.50

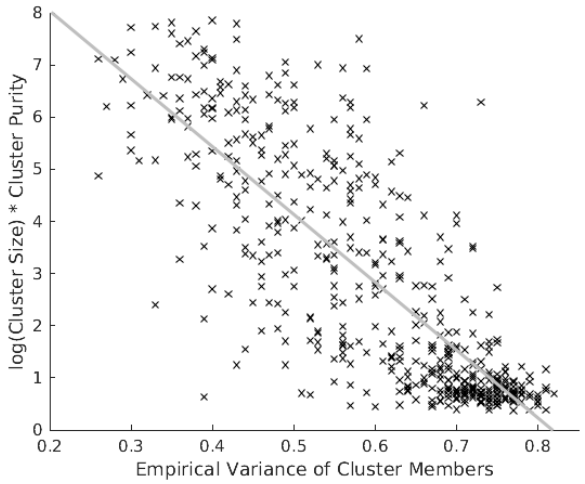


Figure 2: Scatter plot of audio cluster purity weighted by log cluster size vs variance for $k = 500$ (least-squares line superimposed).

based on the standard WSJ recipe and trained using the Google ASR hypothesis as a proxy for the transcriptions. Any word whose duration is overlapped 30% or more by the acoustic segment is included in the label string for the segment. We then employ a majority vote scheme to derive the overall cluster labels. When computing the purity of a cluster, we count a cluster member as matching the cluster label as long as the overall cluster label appears in the member's label string. In other words, an acoustic segment overlapping the words "the lighthouse" would receive credit for matching the overall cluster label "lighthouse". A breakdown of the segments captured by two clusters is shown in Table 2. We investigated some simple schemes for predicting highly pure clusters, and found that the empirical variance of the cluster members (average squared distance to the cluster centroid) was a good indicator. Figure 2 displays a scatter plot of cluster purity weighted by the natural log of the cluster size against the empirical variance. Large, pure clusters are easily predicted by their low empirical variance, while a high variance is indicative of a garbage cluster.

Ranking a set of $k = 500$ acoustic clusters by their variance, Table 3 displays some statistics for the 50 lowest-variance clusters. We see that most of the clusters are very large and highly pure, and their labels reflect interesting object categories being identified by the neural network. We additionally compute the coverage of each cluster by counting the total number of instances of the clus-

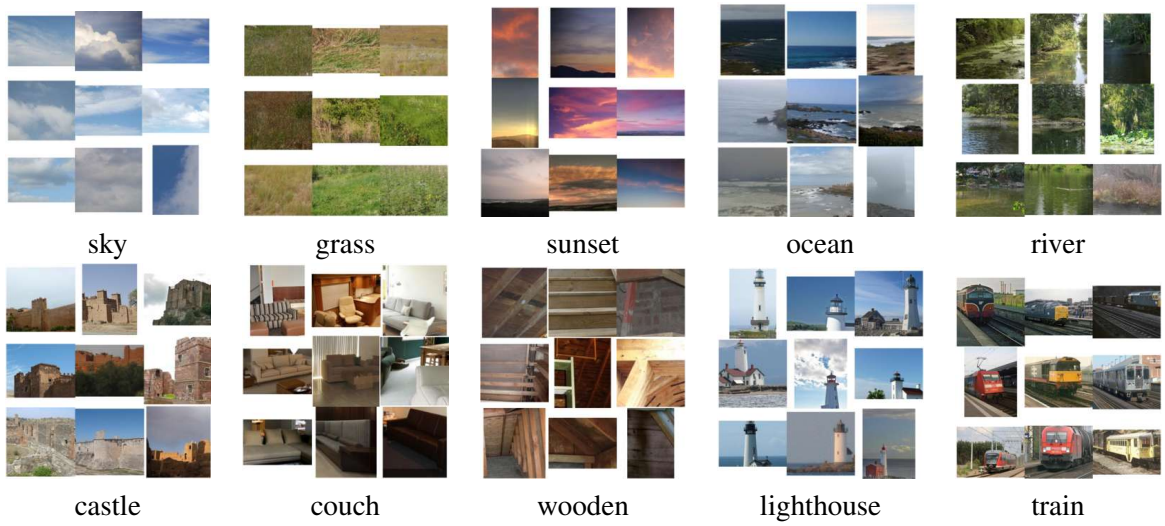


Figure 3: The 9 most central image crops from several image clusters, along with the majority-vote label of their most associated acoustic pattern cluster

Table 4: Clustering statistics of the acoustic clusters for various values of k and different settings of the variance-based cluster pruning threshold. Legend: $|\mathcal{C}|$ = number of clusters remaining after pruning, $|\mathcal{X}|$ = number of datapoints after pruning, Pur = purity, $|\mathcal{L}|$ = number of unique cluster labels, AC = average cluster coverage

	$\sigma^2 < 0.9$					$\sigma^2 < 0.65$				
k	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{X}\rvert$	Pur	$\lvert\mathcal{L}\rvert$	AC	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{X}\rvert$	Pur	$\lvert\mathcal{L}\rvert$	AC
250	249	1081514	.364	149	.423	128	548866	.575	108	.463
500	499	1097225	.396	242	.332	278	623159	.591	196	.375
750	749	1101151	.409	308	.406	434	668771	.585	255	.450
1000	999	1103391	.411	373	.336	622	710081	.568	318	.382
1500	1496	1104631	.429	464	.316	971	750162	.566	413	.366
2000	1992	1106418	.431	540	.237	1354	790492	.546	484	.271

ter label anywhere in the training data, and then compute what fraction of those instances were captured by the cluster. There are many examples of high coverage clusters, e.g. the "skyscraper" cluster captures 84% of all occurrences of the word "skyscraper", while the "baseball" cluster captures 86% of all occurrences of the word "baseball". This is quite impressive given the fact that no conventional speech recognition was employed, and neither the multimodal neural network nor the grounding algorithm had access to the text transcripts of the captions.

To get an idea of the impact of the k parameter as well as a variance-based cluster pruning threshold based on Figure 2, we swept k from 250 to 2000 and computed a set of statistics shown in Table 4. We compute the standard overall cluster purity evaluation metric in addition to the average coverage across clusters. The table shows the natural tradeoff between cluster purity and redundancy (indicated by the average cluster coverage) as k is increased. In all cases, the variance-based cluster pruning greatly increases both the overall purity and average cluster coverage metrics. We also notice that more unique cluster labels are discovered with a larger k.

Next, we examine the image clusters. Figure 3 displays the 9 most central image crops for a set of 10 different image clusters, along with the majority-vote label of each image cluster's associated audio cluster. In all cases, we see that the image crops are highly relevant to their audio cluster label. We include many more example image clusters in Appendix A.

In order to examine the semantic embedding space in more depth, we took the top 150 clusters from the same $k = 500$ clustering run described in Table 3 and performed t-SNE (van der Maaten and Hinton, 2008) analysis on the cluster centroid vectors. We projected each centroid down to 2 di-

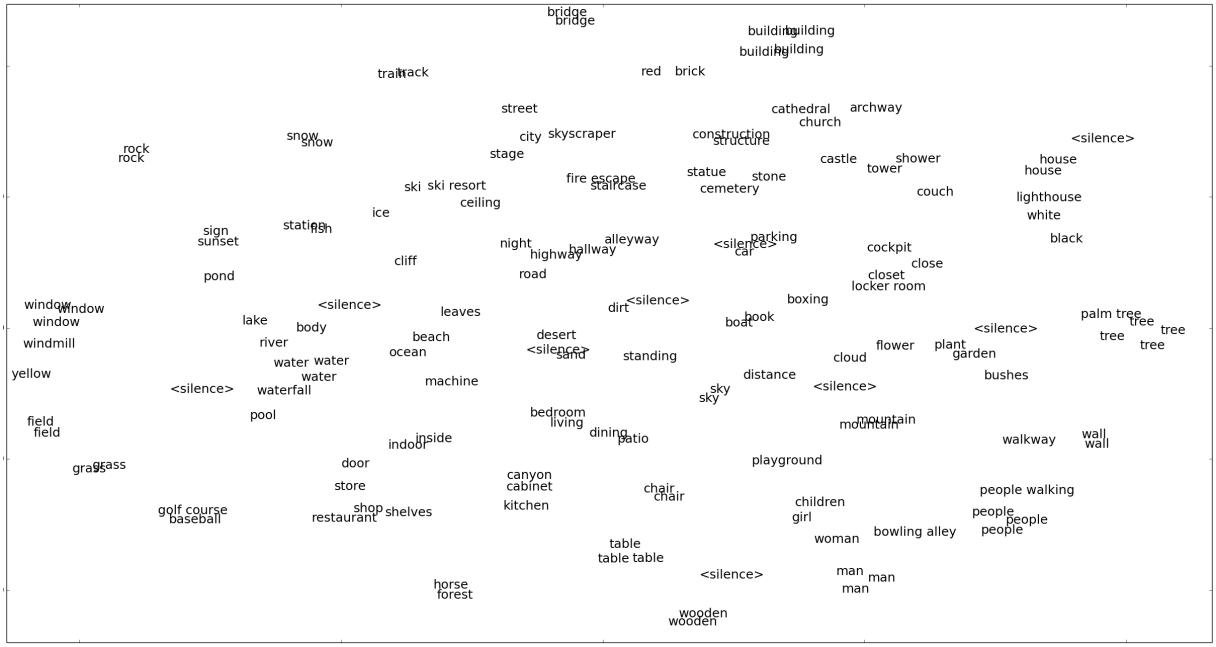


Figure 4: t-SNE analysis of the 150 lowest-variance audio pattern cluster centroids for $k = 500$. Displayed is the majority-vote transcription of the each audio cluster. All clusters shown contained a minimum of 583 members and an average of 2482, with an average purity of .668.

mensions and plotted their majority-vote labels in Figure 4. Immediately we see that different clusters which capture the same label closely neighbor one another, indicating that distances in the embedding space do indeed carry information discriminative across word types (and suggesting that a more sophisticated clustering algorithm than k-means would perform better). More interestingly, we see that semantic information is also reflected in these distances. The cluster centroids for "lake," "river," "body," "water," "waterfall," "pond," and "pool" all form a tight meta-cluster, as do "restaurant," "store," "shop," and "shelves," as well as "children," "girl," "woman," and "man." Many other semantic meta-clusters can be seen in Figure 4, suggesting that the embedding space is capturing information that is highly discriminative both acoustically *and* semantically.

Because our experiments revolve around the discovery of word and object categories, a key question to address is the extent to which the supervision used to train the VGG network constrains or influences the kinds of objects learned. Because the 1,000 object classes from the ILSVRC2012 task (Russakovsky et al., 2015) used to train the VGG network were derived from WordNet synsets (Fellbaum, 1998), we can measure the semantic similarity between the words learned by our network and the ILSVRC2012 class labels by using synset similarity measures within WordNet. We do this by first building a list of the 1,000 WordNet synsets associated with the ILSVRC2012 classes. We then take the set of unique majority-vote labels associated with the discovered word clusters for $k = 500$, filtered by setting a threshold on their variance ($\sigma^2 \leq 0.65$) so as to get rid of garbage clusters, leaving us with 197 unique acoustic cluster labels. We then look up each cluster label in WordNet, and compare all noun senses of the label to every ILSVRC2012 class synset according to the path similarity measure. This measure describes the distance between two synsets in a hyponym/hypernym hierarchy, where a score of 1 represents identity and lower scores indicate less similarity. We retain the highest score between any sense of the cluster label and any ILSVRC2012 synset. Of the 197 unique cluster labels, only 16 had a distance of 1 from any ILSVRC12 class, which would indicate an exact match. A path similarity of 0.5 indicates one degree of separation in the hyponym/hypernym hierarchy - for example, the similarity between "desk" and "table" is 0.5. 47 cluster labels were found to have a similarity of 0.5 to some ILSVRC12 class, leaving 134 cluster labels whose highest similarity to any ILSVRC12 class was less than 0.5. In

other words, more than two thirds of the highly pure pattern clusters learned by our network were dissimilar to all of the 1,000 ILSVRC12 classes used to pretrain the VGG network, indicating that our model is able to generalize far beyond the set of classes found in the ILSVRC12 data. We display the labels of the 40 lowest variance acoustic clusters labels along with the name and similarity score of their closest ILSVRC12 synset in Table 5.

Cluster	ILSVRC synset	Similarity
snow	cliff.n.01	0.14
desert	cliff.n.01	0.12
kitchen	patio.n.01	0.25
restaurant	restaurant.n.01	1.00
mountain	alp.n.01	0.50
black	pool_table.n.01	0.25
skyscraper	greenhouse.n.01	0.33
bridge	steel_arch_bridge.n.01	0.50
tree	daisy.n.01	0.14
castle	castle.n.02	1.00
ocean	cliff.n.01	0.14
table	desk.n.01	0.50
windmill	cash_machine.n.01	0.20
window	screen.n.03	0.33
river	cliff.n.01	0.12
water	menu.n.02	0.25
beach	cliff.n.01	0.33
flower	daisy.n.01	0.50
wall	cliff.n.01	0.33
sky	cliff.n.01	0.11
street	swing.n.02	0.14
golf course	swing.n.02	0.17
field	cliff.n.01	0.20
lighthouse	beacon.n.03	1.00
forest	cliff.n.01	0.20
church	church.n.02	1.00
people	street_sign.n.01	0.17
baseball	baseball.n.02	1.00
car	freight_car.n.01	0.50
shower	swing.n.02	0.17
people walking	(none)	0.00
wooden	(none)	0.00
rock	toilet_tissue.n.01	0.20
night	street_sign.n.01	0.14
station	swing.n.02	0.20
chair	barber_chair.n.01	0.50
building	greenhouse.n.01	0.50
city	cliff.n.01	0.12
white	jean.n.01	0.33
sunset	street_sign.n.01	0.11

Table 5: The 40 lowest variance, uniquely-labeled acoustic clusters paired with their most similar ILSVRC2012 synset.

6 Conclusions and Future Work

In this paper, we have demonstrated that a neural network trained to associate images with the waveforms representing their spoken audio captions can successfully be applied to discover and cluster acoustic patterns representing words or short phrases in untranscribed audio data. An analogous procedure can be applied to visual images to discover visual patterns, and then the two modalities can be linked, allowing the network to learn, for example, that spoken instances of the word "train" are associated with image regions containing trains. This is done without the use of a conventional automatic speech recognition system and zero text transcriptions, and therefore is completely agnostic to the language in which the captions are spoken. Further, this is done in $O(n)$ time with respect to the number of image/caption pairs, whereas previous state-of-the-art acoustic pattern discovery algorithms which leveraged acoustic data alone run in $O(n^2)$ time. We demonstrate the success of our methodology on a large-scale dataset of over 214,000 image/caption pairs comprising over 522 hours of spoken audio data, which is to our knowledge the largest scale acoustic pattern discovery experiment ever performed. We have shown that the shared multimodal embedding space learned by our model is discriminative not only across visual object categories, but also acoustically *and* semantically across spoken words.

The future directions in which this research could be taken are incredibly fertile. Because our method creates a segmentation as well as an alignment between images and their spoken captions, a generative model could be trained using these alignments. The model could provide a spoken caption for an arbitrary image, or even synthesize an image given a spoken description. Modeling improvements are also possible, aimed at the goal of incorporating both visual and acoustic localization into the neural network itself. The same framework we use here could be extended to video, enabling the learning of actions, verbs, environmental sounds, and the like. Additionally, by collecting a second dataset of captions for our images in a different language, such as Spanish, our model could be extended to learn the acoustic correspondences for a given object category in both languages. This paves the way for creating a speech-to-speech translation model not only with absolutely zero need for any sort of text transcriptions, but also with zero need for directly parallel linguistic data or manual human translations.

References

Alessandro Bergamo, Loris Bazzani, Dragomir Anguelov, and Lorenzo Torresani. 2014. Self-taught object localization with deep networks. *CoRR* abs/1409.3964. http://arxiv.org/abs/1409.3964.

Minsu Cho, Suha Kwak, Cordelia Schmid, and Jean Ponce. 2015. Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In *Proceedings of CVPR*.

Ramazan Cinbis, Jakob Verbeek, and Cordelia Schmid. 2016. Weakly supervised object localization with multi-fold multiple instance learning. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Mark Dredze, Aren Jansen, Glen Coppersmith, and Kenneth Church. 2010. NLP on spoken documents without ASR. In *Proceedings of EMNLP*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Srivastava Rupesh, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, Platt John C., C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *Proceedings of CVPR*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Andrea Frome, Greg S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Proceedings of the Neural Information Processing Society*.

John Garofolo, Lori Lamel, William Fisher, Jonathan Fiscus, David Pallet, Nancy Dahlgren, and Victor Zue. 1993. The TIMIT acoustic-phonetic continuous speech corpus.

Lieke Gelderloos and Grzegorz Chrupaa. 2016. From phonemes to images: levels of representation in a recurrent neural model of visually-grounded language learning. In *arXiv:1610.03342*.

Sharon Goldwater, Thomas Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: exploring the effects of context. In *Cognition, vol. 112 pp.21-54*.

David Harwath and James Glass. 2015. Deep multimodal semantic embeddings for speech and images. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*.

David Harwath, Timothy J. Hazen, and James Glass. 2012. Zero resource spoken audio corpus analysis. In *Proceedings of ICASSP*.

David Harwath, Antonio Torralba, and James R. Glass. 2016. Unsupervised learning of spoken language with visual context. In *Proceedings of NIPS*.

Aren Jansen, Kenneth Church, and Hynek Hermansky. 2010. Toward spoken term discovery at scale with zero resources. In *Proceedings of Interspeech*.

Aren Jansen and Benjamin Van Durme. 2011. Efficient spoken term discovery using randomized algorithms. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. 2016. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of CVPR*.

Mark Johnson. 2008. Unsupervised word segmentation for sesotho using adaptor grammars. In *Proceedings of ACL SIG on Computational Morphology and Phonology*.

Andrej Karpathy, Armand Joulin, and Fei-Fei Li. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In *Proceedings of the Neural Information Processing Society*.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of CVPR*.

Chia-Ying Lee and James Glass. 2012. A nonparametric Bayesian approach to acoustic model discovery. In *Proceedings of the 2012 meeting of the Association for Computational Linguistics*.

Chia-Ying Lee, Timothy J. O'Donnell, and James Glass. 2015. Unsupervised lexicon discovery from acoustic input. In *Transactions of the Association for Computational Linguistics*.

M. Paul Lewis, Gary F. Simon, and Charles D. Fennig. 2016. *Ethnologue: Languages of the World, Nineteenth edition*. SIL International. Online version: http://www.ethnologue.com.

Lucas Ondel, Lukas Burget, and Jan Cernocky. 2016. Variational inference for acoustic unit discovery. In *5th Workshop on Spoken Language Technology for Underresourced Language*.

Alex Park and James Glass. 2008. Unsupervised pattern discovery in speech. In *IEEE Transactions on Audio, Speech, and Language Processing vol. 16, no.1, pp. 186-197*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*.

Deb Roy. 2003. Grounded spoken language acquisition: Experiments in word learning. In *IEEE Transactions on Multimedia*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252. https://doi.org/10.1007/s11263-015-0816-y.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. In *Transactions of the Association for Computational Linguistics*.

Richard Socher and Fei-Fei Li. 2010. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *Proceedings of CVPR*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing high-dimensional data using t-sne. In *Journal of Machine Learning Research*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dimitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of CVPR*.

Yaodong Zhang and James Glass. 2009. Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams. In *Proceedings ASRU*.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2015. Object detectors emerge in deep scene CNNs. In *Proceedings of ICLR*.

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. 2014. Learning deep features for scene recognition using places database. In *Proceedings of the Neural Information Processing Society*.

A Additional Cluster Visualizations

Joint CTC/attention decoding for end-to-end speech recognition

Takaaki Hori, Shinji Watanabe, John R. Hershey
Mitsubishi Electric Research Laboratories (MERL)
{thori,watanabe,hershey}@merl.com

Abstract

End-to-end automatic speech recognition (ASR) has become a popular alternative to conventional DNN/HMM systems because it avoids the need for linguistic resources such as pronunciation dictionary, tokenization, and context-dependency trees, leading to a greatly simplified model-building process. There are two major types of end-to-end architectures for ASR: attention-based methods use an attention mechanism to perform alignment between acoustic frames and recognized symbols, and connectionist temporal classification (CTC), uses Markov assumptions to efficiently solve sequential problems by dynamic programming. This paper proposes a joint decoding algorithm for end-to-end ASR with a hybrid CTC/attention architecture, which effectively utilizes both advantages in decoding. We have applied the proposed method to two ASR benchmarks (spontaneous Japanese and Mandarin Chinese), and showing the comparable performance to conventional state-of-the-art DNN/HMM ASR systems without linguistic resources.

1 Introduction

Automatic speech recognition (ASR) is currently a mature set of technologies that have been widely deployed, resulting in great success in interface applications such as voice search. A typical ASR system is factorized into several modules including acoustic, lexicon, and language models based on a probabilistic noisy channel model (Jelinek, 1976). Over the last decade, dramatic improvements in acoustic and language models have been driven by machine learning techniques known as deep learning (Hinton et al., 2012).

However, current systems lean heavily on the scaffolding of complicated legacy architectures that grew up around traditional techniques. For example, when we build an acoustic model from scratch, we have to first build hidden Markov model (HMM) and Gaussian mixture model (GMM) followed by deep neural networks (DNN). In addition, the factorization of acoustic, lexicon, and language models is derived by conditional independence assumptions (especially Markov assumptions), although the data do not necessarily follow such assumptions leading to model misspecification. This factorization form also yields a local optimum since the above modules are optimized separately. Further, to well factorize acoustic and language models, the system requires linguistic knowledge based on a lexicon model, which is usually based on a hand-crafted pronunciation dictionary to map word to phoneme sequence. In addition to the pronunciation dictionary issue, some languages, which do not explicitly have a word boundary, need language-specific tokenization modules (Kudo et al., 2004; Bird, 2006) for language modeling. Finally, inference/decoding has to be performed by integrating all modules resulting in complex decoding. Consequently, it is quite difficult for non-experts to use/develop ASR systems for new applications, especially for new languages.

End-to-end ASR has the goal of simplifying the above module-based architecture into a single-network architecture within a deep learning framework, in order to address the above issues. There are two major types of end-to-end architectures for ASR: attention-based methods use an attention mechanism to perform alignment between acoustic frames and recognized symbols, and connectionist temporal classification (CTC), uses Markov

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 518–529
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1048

assumptions to efficiently solve sequential problems by dynamic programming (Chorowski et al., 2014; Graves and Jaitly, 2014).

The attention-based end-to-end method solves the ASR problem as a sequence mapping from speech feature sequences to text by using encoder-decoder architecture. The decoder network uses an attention mechanism to find an alignment between each element of the output sequence and the hidden states generated by the acoustic encoder network for each frame of acoustic input (Chorowski et al., 2014, 2015; Chan et al., 2015; Lu et al., 2016). This basic temporal attention mechanism is too flexible in the sense that it allows extremely non-sequential alignments. This may be fine for applications such as machine translation where input and output word order are different (Bahdanau et al., 2014; Wu et al., 2016). However, in speech recognition, the feature inputs and corresponding letter outputs generally proceed in the same order. Another problem is that the input and output sequences in ASR can have very different lengths, and these vary greatly from case to case, depending on the speaking rate and writing system, making it more difficult to track the alignment.

However, an advantage is that the attention mechanism does not require any conditional independence assumptions, and could address all the problems cited above. Although the alignment problems of attention-based mechanisms have been partially addressed in (Chorowski et al., 2014; Chorowski and Jaitly, 2016) using various mechanisms, here we propose more rigorous constraints by using CTC-based alignment to guide the decoding.

CTC permits an efficient computation of a strictly monotonic alignment using dynamic programming (Graves et al., 2006; Graves and Jaitly, 2014) although it requires language models and graph-based decoding (Miao et al., 2015) except in the case of huge training data (Amodei et al., 2015; Soltau et al., 2016). We propose to take advantage of the constrained CTC alignment in a hybrid CTC/attention based system during *decoding*. The proposed method adopts a CTC/attention hybrid architecture, which was originally designed to regularize an attention-based encoder network by additionally using a CTC during *training* (Kim et al., 2017). The proposed method extends the architecture to perform one-pass/rescoring joint decoding, where hypotheses of attention-based ASR are boosted by scores obtained by using CTC outputs. This greatly reduces irregular alignments without any heuristic search techniques.

The proposed method is applied to Japanese and Mandarin ASR tasks, which require extra linguistic resources including morphological analyzer (Kudo et al., 2004) or word segmentation (Xue et al., 2003) in addition to pronunciation dictionary to provide accurate lexicon and language models in conventional DNN/HMM ASR. Surprisingly, the method achieved performance comparable to, and in some cases superior to, several state-of-the-art DNN/HMM ASR systems, without using the above linguistic resources.

2 From DNN/HMM to end-to-end ASR

This section briefly provides a formulation of conventional DNN/HMM ASR and CTC or attention based end-to-end ASR.

2.1 Conventional DNN/HMM ASR

ASR deals with a sequence mapping from T-length speech feature sequence $X = \{\mathbf{x}_t \in \mathbb{R}^D | t = 1, \cdots, T\}$ to N-length word sequence $W = \{w_n \in \mathcal{V} | n = 1, \cdots, N\}$. $\mathbf{x}_t$ is a D dimensional speech feature vector (e.g., log Mel filterbanks) at frame t and w_n is a word at position n in vocabulary $\mathcal{V}$. ASR is mathematically formulated with the Bayes decision theory, where the most probable word sequence $\hat{W}$ is estimated among all possible word sequences $\mathcal{V}*$ as follows:

$$\hat{W} = \arg\max_{W \in \mathcal{V}*} p(W|X). \quad (1)$$

The posterior distribution $p(W|X)$ is factorized into the following three distributions by using the Bayes theorem and introducing HMM state sequence $S = \{s_t \in \{1, \cdots, J\} | t = 1, \cdots, T\}$:

$$\text{Eq. (1)} \approx \arg\max_{W} \sum_{S} p(X|S)p(S|W)p(W).$$

The three factors, $p(X|S)$, $p(S|W)$, and $p(W)$, are acoustic, lexicon, and language models, respectively. These are further factorized by using a probabilistic chain rule and conditional independence assumption as follows:

$$\begin{cases} p(X|S) \approx \prod_t \frac{p(s_t|\mathbf{x}_t)}{p(s_t)}, \\ p(S|W) \approx \prod_t p(s_t|s_{t-1}, W), \\ p(W) \quad \approx \prod_n p(w_n|w_{n-1}, \ldots, w_{n-m-1}), \end{cases}$$

where the acoustic model is replaced with the product of framewise posterior distributions $p(s_t|\mathbf{x}_t)$ computed by powerful DNN classifiers by using so-called pseudo likelihood trick (Bourlard and Morgan, 1994). $p(s_t|s_{t-1}, W)$ is represented by an HMM state transition given W, and the conversion from W to HMM states is deterministically performed by using a pronunciation dictionary through a phoneme representation. $p(w_n|w_{n-1}, \ldots, w_{n-m-1})$ is obtained based on an $(m-1)$th-order Markov assumption as a m-gram model.

These conditional independence assumptions are often regarded as too strong assumption, leading to model mis-specification. Also, to train the framewise posterior $p(s_t|\mathbf{x}_t)$, we have to provide a framewise state alignment s_t as a target, which is often provided by a GMM/HMM system. Thus, conventional DNN/HMM systems make the ASR problem formulated with Eq. (1) feasible by using factorization and conditional independence assumptions, at the cost of the problems discussed in Section 1.

2.2 Connectionist Temporal Classification (CTC)

The CTC formulation also follows from Bayes decision theory (Eq. (1)). Note that the CTC formulation uses L-length letter sequence $C = \{c_l \in \mathcal{U} | l = 1, \cdots, L\}$ with a set of distinct letters $\mathcal{U}$. Similarly to Section 2.1, by introducing framewise letter sequence with an additional "blank" ($<\mathtt{b}>$) symbol $Z = \{z_t \in \mathcal{U} \cup <\mathtt{b}> | t = 1, \cdots, T\}$, and by using the probabilistic chain rule and conditional independence assumption, the posterior distribution $p(C|X)$ is factorized as follows:

$$p(C|X) \approx \underbrace{\sum_{Z} \prod_{t} p(z_t|z_{t-1}, C) p(z_t|X)}_{\triangleq p_{\text{ctc}}(C|X)} \frac{p(C)}{p(Z)} \tag{2}$$

As a result, CTC has three distribution components similar to the DNN/HMM case, i.e., framewise posterior distribution $p(z_t|X)$, transition probability $p(z_t|z_{t-1}, C)$[1], and prior distributions of letter and hidden-state sequences, $p(C)$ and $p(Z)$, respectively. We also define the CTC objective function $p_{\text{ctc}}(C|X)$ used in the later formulation. The framewise posterior distribution $p(z_t|X)$ is conditioned on all inputs X, and it is quite natural to be modeled by using bidirectional long short-term memory (BLSTM): $p(z_t|X) = \text{Softmax}(\text{Lin}(\mathbf{h}_t))$ and $\mathbf{h}_t = \text{BLSTM}(X)$. $\text{Softmax}(\cdot)$ is a sofmax activation function, and $\text{Lin}(\cdot)$ is a linear layer to convert hidden vector $\mathbf{h}_t$ to a $(|\mathcal{U}|+1)$ dimensional vector ($+1$ means a blank symbol introduced in CTC).

Although Eq. (2) has to deal with a summation over all possible Z, it is efficiently computed by using dynamic programming (Viterbi/forward-backward algorithm) thanks to the Markov property. In summary, although CTC and DNN/HMM systems are similar to each other due to conditional independence assumptions, CTC does not require pronunciation dictionaries and omits an GMM/HMM construction step.

2.3 Attention mechanism

Compared with hybrid and CTC approaches, the attention-based approach does not make any conditional independence assumptions, and directly estimates the posterior $p(C|X)$ based on a probabilistic chain rule, as follows:

$$p(C|X) = \underbrace{\prod_{l} p(c_l|c_1, \cdots, c_{l-1}, X)}_{\triangleq p_{\text{att}}(C|X)}, \tag{3}$$

where $p_{\text{att}}(C|X)$ is an attention-based objective function. $p(c_l|c_1, \cdots, c_{l-1}, X)$ is obtained by

$$p(c_l|c_1, \cdots, c_{l-1}, X) = \text{Decoder}(\mathbf{r}_l, \mathbf{q}_{l-1}, c_{l-1})$$
$$\mathbf{h}_t = \text{Encoder}(X) \tag{4}$$
$$a_{lt} = \text{Attention}(\{a_{l-1}\}_t, \mathbf{q}_{l-1}, \mathbf{h}_t) \tag{5}$$
$$\mathbf{r}_l = \sum_t a_{lt} \mathbf{h}_t. \tag{6}$$

Eq. (4) converts input feature vectors X into a framewise hidden vector $\mathbf{h}_t$ in an encoder network based on BLSTM, i.e., $\text{Encoder}(X) \triangleq \text{BLSTM}(X)$. $\text{Attention}(\cdot)$ in Eq. (5) is based on a content-based attention mechanism with convolutional features, as described in (Chorowski et al., 2015) (see Appendix A). a_{lt} is an attention weight, and represents a soft alignment of hidden vector $\mathbf{h}_t$ for each output c_l based on the weighted summation of hidden vectors to form letter-wise hidden vector $\mathbf{r}_l$ in Eq. (6). A decoder network is another

[1] Note that in the implementation, the transition value is not normalized (i.e., not a probabilistic value) (Graves and Jaitly, 2014; Miao et al., 2015), similar to the HMM state transition implementation (Povey et al., 2011)

recurrent network conditioned on previous output c_{l-1} and hidden vector $\mathbf{q}_{l-1}$, similar to RNNLM, in addition to letter-wise hidden vector $\mathbf{r}_l$. We use $\text{Decoder}(\cdot) \triangleq \text{Softmax}(\text{Lin}(\text{LSTM}(\cdot)))$.

Attention-based ASR does not explicitly separate each module, and potentially handles the all issues pointed out in Section 1. It implicitly combines acoustic models, lexicon, and language models as encoder, attention, and decoder networks, which can be jointly trained as a single deep neural network.

Compared with DNN/HMM and CTC, which are based on a transition form from $t-1$ to t due to the Markov assumption, the attention mechanism does not maintain this constraint, and often provides irregular alignments. A major focus of this paper is to address this problem by using joint CTC/attention decoding.

3 Joint CTC/attention decoding

This section explains a hybrid CTC/attention network, which potentially utilizes both benefits of CTC and attention in ASR.

3.1 Hybrid CTC/attention architecture

Kim et al. (2017) uses a CTC objective function as an auxiliary task to train the attention model encoder within the multitask learning (MTL) framework, and this paper also uses the same architecture. Figure 1 illustrates the overall architecture of the framework, where the same BLSTM is shared with CTC and attention encoder networks, respectively). Unlike the sole attention model, the forward-backward algorithm of CTC can enforce monotonic alignment between speech and label sequences during training. That is, rather than solely depending on data-driven attention methods to estimate the desired alignments in long sequences, the forward-backward algorithm in CTC helps to speed up the process of estimating the desired alignment. The objective to be maximized is a logarithmic linear combination of the CTC and attention objectives, i.e., $p_{\text{ctc}}(C|X)$ in Eq. (2) and $p_{\text{att}}(C|X)$ in Eq. (3):

$$\mathcal{L}_{\text{MTL}} = \lambda \log p_{\text{ctc}}(C|X) + (1-\lambda) \log p_{\text{att}}(C|X), \tag{7}$$

with a tunable parameter $\lambda : 0 \leq \lambda \leq 1$.

3.2 Decoding strategies

The inference step of our joint CTC/attention-based end-to-end speech recognition is performed

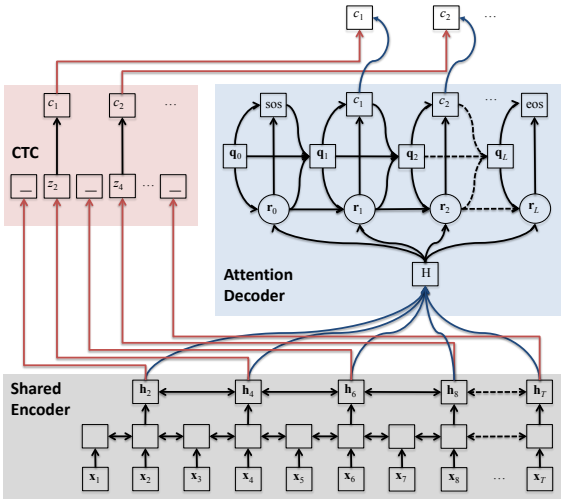


Figure 1: Joint CTC/attention based end-to-end framework: the shared encoder is trained by both CTC and attention model objectives simultaneously. The shared encoder transforms our input sequence $\{\mathbf{x}_t \cdots \mathbf{x}_T\}$ into high level features $H = \{\mathbf{h}_t \cdots \mathbf{h}_T\}$, and the attention decoder generates the letter sequence $\{c_1 \cdots c_L\}$.

by label synchronous decoding with a beam search similar to conventional attention-based ASR. However, we take the CTC probabilities into account to find a hypothesis that is better aligned to the input speech, as shown in Figure 1. Hereafter, we describe the general attention-based decoding and conventional techniques to mitigate the alignment problem. Then, we propose joint decoding methods with a hybrid CTC/attention architecture.

3.2.1 Attention-based decoding in general

End-to-end speech recognition inference is generally defined as a problem to find the most probable letter sequence $\hat{C}$ given the speech input X, i.e.

$$\hat{C} = \arg\max_{C \in \mathcal{U}^*} \log p(C|X). \tag{8}$$

In attention-based ASR, $p(C|X)$ is computed by Eq. (3), and $\hat{C}$ is found by a beam search technique.

Let Ω_l be a set of partial hypotheses of the length l. At the beginning of the beam search, Ω_0 contains only one hypothesis with the starting symbol <sos> and the hypothesis score $\alpha(\texttt{<sos>}, X)$ is set to 0. For $l = 1$ to L_{max}, each partial hypothesis in Ω_{l-1} is expanded by appending possible single letters, and the new hypotheses are stored in Ω_l, where L_{max} is the maximum

length of the hypotheses to be searched. The score of each new hypothesis is computed in the log domain as

$$\alpha(h, X) = \alpha(g, X) + \log p(c|g, X), \quad (9)$$

where g is a partial hypothesis in Ω_{l-1}, c is a letter appended to g, and h is the new hypothesis such that $h = g \cdot c$. If c is a special symbol that represents the end of a sequence, <eos>, h is added to $\hat{\Omega}$ but not Ω_l, where $\hat{\Omega}$ denotes a set of complete hypotheses. Finally, $\hat{C}$ is obtained by

$$\hat{C} = \arg\max_{h \in \hat{\Omega}} \alpha(h, X). \quad (10)$$

In the beam search process, Ω_l is allowed to hold only a limited number of hypotheses with higher scores to improve the search efficiency.

Attention-based ASR, however, may be prone to include deletion and insertion errors because of its flexible alignment property, which can attend to any portion of the encoder state sequence to predict the next label, as discussed in Section 2.3. Since attention is generated by the decoder network, it may prematurely predict the end-of-sequence label, even when it has not attended to all of the encoder frames, making the hypothesis too short. On the other hand, it may predict the next label with a high probability by attending to the same portions as those attended to before. In this case, the hypothesis becomes very long and includes repetitions of the same letter sequence.

3.2.2 Conventional decoding techniques

To alleviate the alignment problem, a length penalty term is commonly used to control the hypothesis length to be selected (Chorowski et al., 2015; Bahdanau et al., 2016). With the length penalty, the decoding objective in Eq. (8) is changed to

$$\hat{C} = \arg\max_{C \in \mathcal{U}^*} \{\log p(C|X) + \gamma|C|\}, \quad (11)$$

where $|C|$ is the length of the sequence C, and γ is a tunable parameter. However, it is actually difficult to completely exclude hypotheses that are too long or too short even if γ is carefully tuned. It is also effective to control the hypothesis length by the minimum and maximum lengths to some extent, where the minimum and maximum are selected as fixed ratios to the length of the input speech. However, since there are exceptionally long or short transcripts compared to the input speech, it is difficult to balance saving such exceptional transcripts and preventing hypotheses with irrelevant lengths.

Another approach is the *coverage* term recently proposed in (Chorowski and Jaitly, 2016), which is incorporated in the decoding objective in Eq. (11) as

$$\hat{C} = \arg\max_{C \in \mathcal{U}^*} \{\log p(C|X) + \gamma|C| + \eta \cdot \text{coverage}(C|X)\}, \quad (12)$$

where the coverage term is computed by

$$\text{coverage}(C|X) = \sum_{t=1}^{T} \left[\sum_{l=1}^{L} a_{lt} > \tau \right]. \quad (13)$$

η and τ are tunable parameters. The coverage term represents the number of frames that have received a cumulative attention greater than τ. Accordingly, it increases when paying close attention to some frames for the first time, but does not increase when paying attention again to the same frames. This property is effective for avoiding looping of the same label sequence within a hypothesis. However, it is still difficult to obtain a common parameter setting for γ, η, τ, and the optional min/max lengths so that they are appropriate for any speech data from different tasks.

3.2.3 Joint decoding

Our joint CTC/attention approach combines the CTC and attention-based sequence probabilities in the inference step, as well as the training step. Suppose $p_{\text{ctc}}(C|X)$ in Eq. (2) and $p_{\text{att}}(C|X)$ in Eq. (3) are the sequence probabilities given by CTC and the attention model. The decoding objective is defined similarly to Eq. (7) as

$$\hat{C} = \arg\max_{C \in \mathcal{U}^*} \{\lambda \log p_{\text{ctc}}(C|X) + (1-\lambda) \log p_{\text{att}}(C|X)\}. \quad (14)$$

The CTC probability enforces a monotonic alignment that does not allow large jumps or looping of the same frames. Accordingly, it is possible to choose a hypothesis with a better alignment and exclude irrelevant hypotheses without relying on the coverage term, length penalty, or min/max lengths.

In the beam search process, the decoder needs to compute a score for each partial hypothesis using Eq. (9). However, it is nontrivial to combine the CTC and attention-based scores in the beam

search, because the attention decoder performs it output-label-synchronously while CTC performs it frame-synchronously. To incorporate the CTC probabilities in the hypothesis score, we propose two methods.

Rescoring

The first method is a two-pass approach, in which the first pass obtains a set of complete hypotheses using the beam search, where only the attention-based sequence probabilities are considered. The second pass rescores the complete hypotheses using the CTC and attention probabilities, where the CTC probabilities are obtained by the forward algorithm for CTC (Graves et al., 2006). The rescoring pass obtains the final result according to

$$\hat{C} = \arg\max_{h \in \hat{\Omega}} \{\lambda \alpha_{\text{ctc}}(h, X) + (1 - \lambda)\alpha_{\text{att}}(h, X)\}, \tag{15}$$

where

$$\begin{cases} \alpha_{\text{ctc}}(h, X) & \triangleq \log p_{\text{ctc}}(h|X) \\ \alpha_{\text{att}}(h, X) & \triangleq \log p_{\text{att}}(h|X) \end{cases}. \tag{16}$$

One-pass decoding

The second method is one-pass decoding, in which we compute the probability of each partial hypothesis using CTC and an attention model. Here, we utilize the CTC prefix probability (Graves, 2008) defined as the cumulative probability of all label sequences that have the partial hypothesis h as their prefix:

$$p_{\text{ctc}}(h, \ldots|X) = \sum_{\nu \in (\mathcal{U} \cup \{\texttt{<eos>}\})^+} p_{\text{ctc}}(h \cdot \nu|X),$$

and we define the CTC score as

$$\alpha_{\text{ctc}}(h, X) \triangleq \log p_{\text{ctc}}(h, \ldots|X), \tag{17}$$

where ν represents all possible label sequences except the empty string. The CTC score cannot be obtained recursively as in Eq. (9), but it can be computed efficiently by keeping the forward probabilities over the input frames for each partial hypothesis. Then it is combined with $\alpha_{\text{att}}(h, X)$.

The beam search algorithm for one-pass decoding is shown in Algorithm 1. Ω_l and $\hat{\Omega}$ are initialized in lines 2 and 3 of the algorithm, which are implemented as queues that accept partial hypotheses of the length l and complete hypotheses, respectively. In lines 4–25, each partial hypothesis g in Ω_{l-1} is extended by each label c in the label set $\mathcal{U}$. Each extended hypothesis h is scored in line 11, where CTC and attention-based scores are obtained by $\alpha_{\text{ctc}}()$ and $\alpha_{\text{att}}()$. After that, if $c =$ <eos>, the hypothesis h is assumed to be complete and stored in $\hat{\Omega}$ in line 13. If $c \neq$ <eos>, h is stored in Ω_l in line 15, where the number of hypotheses in Ω_l is checked in line 16. If the number exceeds the beam width, the hypothesis with the worst score in Ω_l is removed by REMOVEWORST() in line 17.

Algorithm 1 Joint CTC/attention one-pass decoding

```
1: procedure ONEPASSBEAMSEARCH(X, L_max)
2:     Ω_0 ← {<sos>}
3:     Ω̂ ← ∅
4:     for l = 1 ... L_max do
5:         Ω_l ← ∅
6:         while Ω_{l-1} ≠ ∅ do
7:             g ← HEAD(Ω_{l-1})
8:             DEQUEUE(Ω_{l-1})
9:             for each c ∈ U ∪ {<eos>} do
10:                h ← g · c
11:                α(h,X) ← λα_ctc(h,X) + (1−λ)α_att(h,X)
12:                if c = <eos> then
13:                    ENQUEUE(Ω̂, h)
14:                else
15:                    ENQUEUE(Ω_l, h)
16:                    if |Ω_l| > beamWidth then
17:                        REMOVEWORST(Ω_l)
18:                    end if
19:                end if
20:            end for
21:        end while
22:        if ENDDETECT(Ω̂, l) = true then
23:            break                          ▷ exit for loop
24:        end if
25:    end for
26:    return arg max_{h∈Ω̂} α(h, X)
27: end procedure
```

In line 11, the CTC and attention model scores are computed for each partial hypothesis. The attention score is easily obtained in the same manner as Eq. (9), whereas the CTC score requires a modified forward algorithm that computes it label-synchronously. The algorithm to compute the CTC score is summarized in Appendix B. By considering the attention and CTC scores during the beam search, partial hypotheses with irregular alignments can be excluded, and the number of search errors is reduced.

We can optionally apply an end detection technique to reduce the computation by stopping the beam search before l reaches L_{max}. Function ENDDETECT($\hat{\Omega}, l$) in line 22 returns `true` if there is little chance of finding complete hypotheses with higher scores as l increases in the future.

In our implementation, the function returns `true` if

$$\sum_{m=0}^{M-1}\left[\max_{h\in\hat{\Omega}:|h|=l-m}\alpha(h,X)-\max_{h'\in\hat{\Omega}}\alpha(h',X)<D_{\text{end}}\right]=M, \quad (18)$$

where D_{end} and M are predetermined thresholds. This equation becomes true if complete hypotheses with smaller scores are generated M times consecutively. This technique is also available in attention-based decoding and rescoring methods described in Sections 3.2.1–3.2.3.

4 Experiments

We used Japanese and Mandarin Chinese ASR benchmarks to show the effectiveness of the proposed joint CTC/attention decoding approach. The main reason for choosing these two languages is that those ideogram languages have relatively shorter lengths for letter sequences than those in alphabet languages, which reduces computational complexities greatly, and makes it easy to handle context information in a decoder network. Our preliminary investigation shows that Japanese and Mandarin Chinese end-to-end ASR can be easily scaled up, and shows state-of-the-art performance without using various tricks developed in English tasks. Also, we would like to emphasize that the system did not use language-specific processing (e.g., morphological analyzer, Pinyin dictionary), and simply used all appeared characters in their transcriptions including Japanese syllable and Kanji, Chinese, Arabic number, and alphabet characters, as they are.

4.1 Corpus of Spontaneous Japanese (CSJ)

We demonstrated ASR experiments by using the Corpus of Spontaneous Japanese (CSJ) (Maekawa et al., 2000). CSJ is a standard Japanese ASR task based on a collection of monologue speech data including academic lectures and simulated presentations. It has a total of 581 hours of training data and three types of evaluation data, where each evaluation task consists of 10 lectures (totally 5 hours). As input features, we used 40 mel-scale filterbank coefficients, with their first and second order temporal derivatives to obtain a total of 120-dimensional feature vector per frame. The encoder was a 4-layer BLSTM with 320 cells in each layer and direction, and linear projection layer is followed by each BLSTM layer. The 2nd and 3rd bottom layers of the encoder read every second hidden state in the network below, reducing the utterance length by the factor of 4. We used the content-based attention mechanism (Chorowski et al., 2015), where the 10 centered convolution filters of width 100 were used to extract the convolutional features. The decoder network was a 1-layer LSTM with 320 cells. The AdaDelta algorithm (Zeiler, 2012) with gradient clipping (Pascanu et al., 2012) was used for the optimization. D_{end} and M in Eq (18) were set as $\log 1e^{-10}$ and 3, respectively. The hybrid CTC/attention ASR was implemented by using the Chainer deep learning toolkit (Tokui et al., 2015).

Table 1 first compares the character error rate (CER) for conventional attention and MTL based end-to-end ASR without the joint decoding. λ in Eq. (7) was set to 0.1. When decoding, we manually set the minimum and maximum lengths of output sequences by 0.025 and 0.15 times input sequence lengths, respectively. The length penalty γ in Eq. (11) was set to 0.1. Multitask learning (MTL) significantly outperformed attention-based ASR in the all evaluation tasks, which confirms the effectiveness of a hybrid CTC/attention architecture. Table 1 also shows that joint decoding, described in Section 3.2, further improved the performance without setting any search parameters (maximum and minimum lengths, length penalty), but only setting a weight parameter $\lambda = 0.1$ in Eq. (15) similar to the MTL case. Figure 2 also compares the dependency of λ on the CER for the CSJ evaluation tasks, and showing that λ was not so sensitive to the performance if we set λ around the value we used at MTL (i.e., 0.1).

We also compare the performance of the proposed MTL-large, which has a larger network (5-layer encoder network), with the conventional state-of-the-art techniques obtained by using linguistic resources. The state-of-the-art CERs of GMM discriminative training and DNN-sMBR/HMM systems are obtained from the Kaldi recipe (Moriya et al., 2015) and a system based on syllable-based CTC with MAP decoding (Kanda et al., 2016). The Kaldi recipe systems use academic lectures (236h) for AM training and all training-data transcriptions for LM training. Unlike the proposed method, these methods use linguistic resources including a morphological analyzer, pronunciation dictionary, and language model. Note that since the amount of training

Table 1: Character error rate (CER) for conventional attention and hybrid CTC/attention end-to-end ASR. Corpus of Spontaneous Japanese speech recognition (CSJ) task.

Model	Hour	Task1	Task2	Task3
Attention	581	11.4	7.9	9.0
MTL	581	10.5	7.6	8.3
MTL + joint decoding (rescoring)	581	10.1	7.1	7.8
MTL + joint decoding (one pass)	581	10.0	7.1	7.6
MTL-large + joint decoding (rescoring)	581	**8.4**	6.2	**6.9**
MTL-large + joint decoding (one pass)	581	**8.4**	**6.1**	**6.9**
GMM-discr. (Moriya et al., 2015)	236 for AM, 581 for LM	11.2	9.2	12.1
DNN/HMM (Moriya et al., 2015)	236 for AM, 581 for LM	9.0	7.2	9.6
CTC-syllable (Kanda et al., 2016)	581	9.4	7.3	7.5

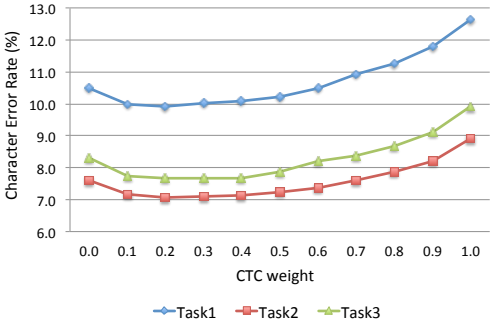


Figure 2: The effect of weight parameter λ in Eq. (14) on the CSJ evaluation tasks (The CERs were obtained by one-pass decoding).

data and experimental configurations of the proposed and reference methods are different, it is difficult to compare the performance listed in the table directly. However, since the CERs of the proposed method are superior to those of the best reference results, we can state that the proposed method achieves the state-of-the-art performance.

4.2 Mandarin telephone speech

We demonstrated ASR experiments on HKUST Mandarin Chinese conversational telephone speech recognition (MTS) (Liu et al., 2006). It has 5 hours recording for evaluation, and we extracted 5 hours from training data as a development set, and used the rest (167 hours) as a training set. All experimental conditions were same as those in Section 4.1 except that we used the $\lambda = 0.5$ in training and decoding instead of 0.1 based on our preliminary investigation and 80 mel-scale filterbank coefficients with pitch features as suggested in (Miao et al., 2016). In decoding, we also added a result of the coverage-term based decoding (Chorowski and Jaitly, 2016), as discussed in Section 3.2 ($\eta = 1.5, \tau = 0.5, \gamma = -0.6$ for attention model and $\eta = 1.0, \tau = 0.5, \gamma = -0.1$ for MTL), since it was difficult to eliminate the irregular alignments during decoding by only tuning the maximum and minimum lengths and length penalty (we set the minimum and maximum lengths of output sequences by 0.0 and 0.1 times input sequence lengths, respectively and set $\gamma = 0.6$ in Table 2).

Table 2 shows the effectiveness of MTL and joint decoding over the attention-based approach, especially showing the significant improvement of the joint CTC/attention decoding. Similar to the CSJ experiments in Section 4.1, we did not use the length-penalty term or the coverage term in joint decoding. This is an advantage of joint decoding over conventional approaches that require many tuning parameters. We also generated more training data by linearly scaling the audio lengths by factors of 0.9 and 1.1 (speed perturb.). The final model achieved **29.9**% without using linguistic resources, which defeats moderate state-of-the-art systems including CTC-based methods[2].

4.3 Decoding speed

We evaluated the speed of the joint decoding methods described in Section 3.2.3. ASR decoding was performed with different beam widths of 1, 3, 5, 10, and 20, and the processing time and CER were measured using a computer with Intel(R) Xeon(R) processors, E5-2690 v3, 2.6 GHz. Although the processors were multicore CPUs and the computer had GPUs, we ran the decoding program as a

[2] Although the proposed method did not reach the performance obtained by a time delayed neural network (TDNN) with lattice-free sequence discriminative training (Povey et al., 2016), our recent work scored **28.0**%, and outperformed the lattice-free MMI result with advanced network architectures.

Table 2: Character error rate (CER) for conventional attention and hybrid CTC/attention end-to-end ASR. HKUST Mandarin Chinese conversational telephone speech recognition (MTS) task.

Model	dev	eval
Attention	40.3	37.8
MTL	38.7	36.6
Attention + coverage	39.4	37.6
MTL + coverage	36.9	35.3
MTL + joint decoding (rescoring)	35.9	34.2
MTL + joint decoding (one pass)	35.5	33.9
MTL-large (speed perturb.) + joint decoding (rescoring)	31.1	30.1
MTL-large (speed perturb.) + joint decoding (one pass)	**31.0**	**29.9**
DNN/HMM	–	35.9
LSTM/HMM (speed perturb.)	–	33.5
CTC with language model (Miao et al., 2016)	–	34.8
TDNN/HMM, lattice-free MMI (speed perturb.) (Povey et al., 2016)	–	28.2

single-threaded process on a CPU to investigate its basic computational cost.

Table 3: RTF versus CER for the one-pass and rescoring methods.

Task	Beam width	Rescoring		One pass	
		RTF	CER	RTF	CER
CSJ Task1	1	0.66	10.9	0.66	10.7
	3	1.11	10.3	1.02	10.1
	5	1.50	10.2	1.31	10.0
	10	2.46	10.1	2.07	10.0
	20	5.02	10.1	3.76	10.0
HKUST Eval set	1	0.68	37.1	0.65	35.9
	3	0.89	34.9	0.86	34.4
	5	1.04	34.6	1.03	34.2
	10	1.55	34.4	1.50	34.0
	20	2.66	34.2	2.55	33.9

Table 3 shows the relationships between the real-time factor (RTF) and the CER for the CSJ and HKUST tasks. We evaluated the rescoring and one-pass decoding methods when using the end detection in Eq. (18). In every beam width, we can see that the one-pass method runs faster with an equal or lower CER than the rescoring method. This result demonstrates that the one-pass decoding is effective for reducing search errors. Finally, we achieved 1xRT with one-pass decoding when using a beam width around 3 to 5, even though it was a single-threaded process on a CPU. However, the decoding process has not yet achieved real-time ASR since CTC and the attention mechanism need to access all of the frames of the input utterance even when predicting the first label. This is an essential problem of most end-to-end ASR approaches and will be solved in future work.

5 Summary and discussion

This paper proposes end-to-end ASR by using joint CTC/attention decoding, which outperformed ordinary attention-based end-to-end ASR by solving the misalignment issues. The joint decoding methods actually reduced most of the irregular alignments, which can be confirmed from the examples of recognition errors and alignment plots shown in Appendix C.

The proposed end-to-end ASR does not require linguistic resources, such as morphological analyzer, pronunciation dictionary, and language model, which are essential components of conventional Japanese and Mandarin Chinese ASR systems. Nevertheless, the method achieved comparable/superior performance to the state-of-the-art conventional systems for the CSJ and MTS tasks. In addition, the proposed method does not require GMM/HMM construction for initial alignments, DNN pre-training, lattice generation for sequence discriminative training, complex search in decoding (e.g., FST decoder or lexical tree search based decoder). Thus, the method greatly simplifies the ASR building process, reducing code size and complexity.

Future work will apply this technique to the other languages including English, where we have to solve an issue of long sequence lengths, which requires heavy computation cost and makes it difficult to train a decoder network. Actually, neural machine translation handles this issue by using a sub word unit (concatenating several letters to form a new sub word unit) (Wu et al., 2016), which would be a promising direction for end-to-end ASR.

References

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. 2015. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 4945–4949.

Steven Bird. 2006. NLTK: the natural language toolkit. In *Joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL) on Interactive presentation sessions*. pages 69–72.

Hervé Bourlard and Nelson Morgan. 1994. *Connectionist speech recognition: A hybrid approach*. Kluwer Academic Publishers.

William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. 2015. Listen, attend and spell. *arXiv preprint arXiv:1508.01211* .

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent NN: First results. *arXiv preprint arXiv:1412.1602* .

Jan Chorowski and Navdeep Jaitly. 2016. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695* .

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems (NIPS)*. pages 577–585.

Alex Graves. 2008. Supervised sequence labelling with recurrent neural networks. *PhD thesis, Technische Universität München* .

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine learning (ICML)*. pages 369–376.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning (ICML)*. pages 1764–1772.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29(6):82–97.

Frederick Jelinek. 1976. Continuous speech recognition by statistical methods. *Proceedings of the IEEE* 64(4):532–556.

Naoyuki Kanda, Xugang Lu, and Hisashi Kawai. 2016. Maximum a posteriori based decoding for CTC acoustic models. In *Interspeech 2016*. pages 1868–1872.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. 2017. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 4835–4839.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *Conference on Empirical Methods on Natural Language Processing (EMNLP)*. volume 4, pages 230–237.

Yi Liu, Pascale Fung, Yongsheng Yang, Christopher Cieri, Shudong Huang, and David Graff. 2006. HKUST/MTS: A very large scale mandarin telephone speech corpus. In *Chinese Spoken Language Processing*, Springer, pages 724–735.

Liang Lu, Xingxing Zhang, and Steve Renals. 2016. On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 5060–5064.

Kikuo Maekawa, Hanae Koiso, Sadaoki Furui, and Hitoshi Isahara. 2000. Spontaneous speech corpus of japanese. In *International Conference on Language Resources and Evaluation (LREC)*. volume 2, pages 947–952.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. 2015. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. pages 167–174.

Yajie Miao, Mohammad Gowayyed, Xingyu Na, Tom Ko, Florian Metze, and Alexander Waibel. 2016. An empirical exploration of ctc acoustic models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 2623–2627.

Takafumi Moriya, Takahiro Shinozaki, and Shinji Watanabe. 2015. Kaldi recipe for Japanese spontaneous speech recognition and its evaluation. In *Autumn Meeting of ASJ*. 3-Q-7.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063* .

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahrmani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. 2016. Purely sequence-trained neural networks for asr based on lattice-free MMI. In *Interspeech*. pages 2751–2755.

Hagen Soltau, Hank Liao, and Hasim Sak. 2016. Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition. *arXiv preprint arXiv:1610.09975* .

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in NIPS*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Nianwen Xue et al. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing* 8(1):29–48.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

A Location-based attention mechanism

This section provides the equations of a location-based attention mechanism Attention$(\cdot)$ in Eq. (5).

$$a_{lt} = \text{Attention}(\{a_{l-1}\}_t, \mathbf{q}_{l-1}, \mathbf{h}_t),$$

where $\{a_{l-1}\}_t = [a_{l-1,1}, \cdots, a_{l-1,T}]^\top$. To obtain a_{lt}, we use the following equations:

$$\{\mathbf{f}_t\}_t = \mathbf{K} * \mathbf{a}_{l-1} \tag{19}$$

$$e_{lt} = \mathbf{g}^\top \tanh(\mathbf{G}^{\text{q}}\mathbf{q}_{l-1} + \mathbf{G}^{\text{h}}\mathbf{h}_t + \mathbf{G}^{\text{f}}\mathbf{f}_t + \mathbf{b}) \tag{20}$$

$$a_{lt} = \frac{\exp(e_{tl})}{\sum_t \exp(e_{tl})} \tag{21}$$

$\mathbf{K}$, $\mathbf{G}^{\text{q}}$, $\mathbf{G}^{\text{h}}$, $\mathbf{G}^{\text{f}}$ are matrix parameters. $\mathbf{b}$ and $\mathbf{g}$ are vector parameters. $*$ denotes convolution along input feature axis t with matrix $\mathbf{K}$ to produce feature $\{\mathbf{f}_t\}_t$.

Algorithm 2 CTC hypothesis score

1: **function** $\alpha_{\text{ctc}}(h, X)$
2: $g, c \leftarrow h$ ▷ split h into the last label c and the rest g
3: **if** $c =$ <eos> **then**
4: **return** $\log\{\gamma_T^{(n)}(g) + \gamma_T^{(b)}(g)\}$
5: **else**
6: $\gamma_1^{(n)}(h) \leftarrow \begin{cases} p(z_1 = c|X) & \text{if } g = \text{<sos>} \\ 0 & \text{otherwise} \end{cases}$
7: $\gamma_1^{(b)}(h) \leftarrow 0$
8: $\Psi \leftarrow \gamma_1^{(n)}(h)$
9: **for** $t = 2 \dots T$ **do**
10: $\Phi \leftarrow \gamma_{t-1}^{(b)}(g) + \begin{cases} 0 & \text{if last}(g) = c \\ \gamma_{t-1}^{(n)}(g) & \text{otherwise} \end{cases}$
11: $\gamma_t^{(n)}(h) \leftarrow \left(\gamma_{t-1}^{(n)}(h) + \Phi\right) p(z_t = c|X)$
12: $\gamma_t^{(b)}(h) \leftarrow \left(\gamma_{t-1}^{(b)}(h) + \gamma_{t-1}^{(n)}(h)\right) p(z_t =$ <b>$|X)$
13: $\Psi \leftarrow \Psi + \Phi \cdot p(z_t = c|X)$
14: **end for**
15: **return** $\log(\Psi)$
16: **end if**
17: **end function**

B CTC-based hypothesis score

The CTC score $\alpha_{\text{ctc}}(h, X)$ in Eq. (17) is computed as shown in Algorithm 2. Let $\gamma_t^{(n)}(h)$ and $\gamma_t^{(b)}(h)$ be the forward probabilities of the hypothesis h over the time frames $1 \dots t$, where the superscripts (n) and (b) denote different cases in which all CTC paths end with a nonblank or blank symbol, respectively. Before starting the beam search, $\gamma_t^{(n)}()$ and $\gamma_t^{(b)}()$ are initialized for $t = 1, \dots, T$ as

$$\gamma_t^{(n)}(\text{<sos>}) = 0, \tag{22}$$

$$\gamma_t^{(b)}(\text{<sos>}) = \prod_{\tau=1}^{t} \gamma_{\tau-1}^{(b)}(\text{<sos>}) p(z_\tau = \text{<b>}|X), \tag{23}$$

where we assume that $\gamma_0^{(b)}(\text{<sos>}) = 1$ and <b> is a blank symbol. Note that the time index t and input length T may differ from those of the input utterance X owing to the subsampling technique for the encoder (Povey et al., 2016; Chan et al., 2015).

In Algorithm 2, the hypothesis h is first split into the last label c and the rest g in line 2. If c is <eos>, it returns the logarithm of the forward probability assuming that h is a complete hypothesis in line 4. The forward probability of h is given by

$$p_{\text{ctc}}(h|X) = \gamma_T^{(n)}(g) + \gamma_T^{(b)}(g) \tag{24}$$

according to the definition of $\gamma_t^{(n)}()$ and $\gamma_t^{(b)}()$. If c is not <eos>, it computes the forward proba-

bilities $\gamma_t^{(n)}(h)$ and $\gamma_t^{(b)}(h)$, and the prefix probability $\Psi = p_{\text{ctc}}(h, \ldots | X)$ assuming that h is not a complete hypothesis. The initialization and recursion steps for those probabilities are described in lines 6–14. In this function, we assume that whenever we compute the probabilities $\gamma_t^{(n)}(h)$, $\gamma_t^{(b)}(h)$ and Ψ, the forward probabilities $\gamma_t^{(n)}(g)$ and $\gamma_t^{(b)}(g)$ have already been obtained through the beam search process because g is a prefix of h such that $|g| < |h|$.

C Examples of irregular alignments

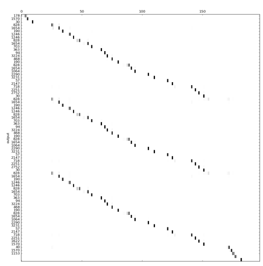
We list examples of irregular alignments caused by attention-based ASR. Figure 3 shows an example of repetitions of word chunks. The first chunk of blue characters in attention-based ASR (MTL) is appeared again, and the whole second chunk part becomes insertion errors. Figure 4 shows an example of deletion errors. The latter half of the sentence in attention-based ASR (MTL) is broken, which causes deletion errors. The hybrid CTC/attention with both multitask learning and joint decoding avoids these issues. Figures 5 and 6 show alignment plots corresponding to Figs. 3 and 4, respectively, where X-axis shows time frames and Y-axis shows the character sequence hypothesis. These visual plots also demonstrate that the proposed joint decoding approach can suppress irregular alignments.

id: (20040717_152947_A010409_B010408-A-057045-057837)
Reference
但是如果你想想如果回到了过去你如果带着这个现在的记忆是不是很痛苦啊
MTL
Scores: (#Correctness #Substitution #Deletion #Insertion) 28 2 3 45
但是如果你想想如果回到了过去你如果带着这个现在的节
如果你想想如果回到了过去你如果带着这个现在的节如果
你想想如果回到了过去你如果带着这个现在的机是不是很
・・・
Joint decoding
Scores: (#Correctness #Substitution #Deletion #Insertion) 31 1 1 0
HYP: 但是如果你想想如果回到了过去你如果带着这个现在的・机是不是很痛苦啊

Figure 3: Example of insertion errors appeared in attention-based ASR with MTL and joint decoding.

id: (A01F0001_0844951_0854386)
Reference
またえ飛行時のエコーロケーション機能をより
詳細に解明する為に超小型マイクロホンおよび
生体アンプをコウモリに搭載することを考えて
おりますそうすることによって
MTL
Scores: (#Correctness #Substitution #Deletion #Insertion) 30 0 47 0
またえ飛行時のエコーロケーション機能をより
詳細に解明する為・・・・・・・・・・・・・
・・・・・・・・・・・・・・・・・・・・・
・・・・・・・・・・に・・・
Joint decoding
Scores: (#Correctness #Substitution #Deletion #Insertion) 67 9 1 0
またえ飛行時のエコーロケーション機能をより
詳細に解明する為に長国型マイクロホンお・い
く声単位方をコウモリに登載することを考えて
おりますそうすることによって

Figure 4: Example of deletion errors appeared in attention-based ASR with MTL and joint decoding.


(a) MTL

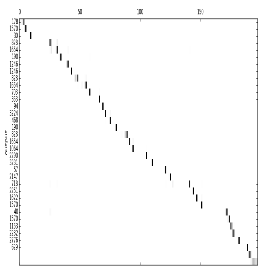
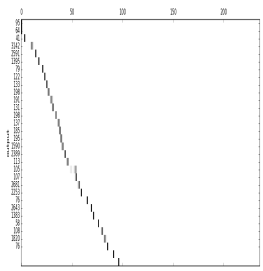

(b) Joint decoding

Figure 5: Example of alignments including insertion errors in attention-based ASR with MTL and joint decoding (Utterance id: 20040717_152947_A010409_B010408-A-057045-057837).


(a) MTL

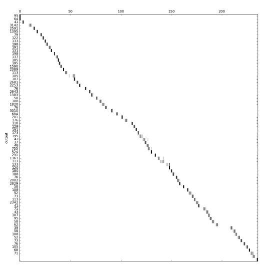

(b) Joint decoding

Figure 6: Example of alignments including deletion errors in attention-based ASR with MTL and joint decoding (Utterance id: A01F0001_0844951_0854386).

Found in Translation: Reconstructing Phylogenetic Language Trees from Translations

Ella Rabinovich[△⋆] **Noam Ordan**[†] **Shuly Wintner**[⋆]
[△]IBM Research Haifa, Israel
[⋆]Department of Computer Science, University of Haifa, Israel
[†]The Arab College for Education, Haifa, Israel
{ellarabi,noam.ordan}@gmail.com, shuly@cs.haifa.ac.il

Abstract

Translation has played an important role in trade, law, commerce, politics, and literature for thousands of years. Translators have always tried to be invisible; ideal translations should look as if they were written originally in the target language. We show that traces of the source language remain in the translation product to the extent that it is possible to uncover the history of the source language by looking only at the translation. Specifically, we automatically reconstruct phylogenetic language trees from *monolingual* texts (translated from several source languages). The signal of the source language is so powerful that it is retained even after two phases of translation. This strongly indicates that source language interference is the most dominant characteristic of translated texts, overshadowing the more subtle signals of universal properties of translation.

1 Introduction

Translation has played a major role in human civilization since the rise of law, religion, and trade in multilingual societies. Evidence of scribe translations goes as far back as four millennia ago, to the time of Hammurabi; this practice is also mentioned in the Bible (Esther 1:22; 8:9). For thousands of years, translators have tried to remain invisible, setting a standard according to which the act of translation should be seamless, and its product should look as if it were written originally in the target language. Cicero (106-43 BC) commented on his translation ethics, "I did not hold it necessary to render word for word, but I preserved the general style and force of the language." These words were echoed 500 years later by St. Jerome (347-420 CE), also known as the patron saint of translators, who wrote, "I render, not word for word, but sense for sense." Translator tendency for invisibility has peaked in the past 150 years in the English speaking world (Venuti, 2008), in spite of some calls for "foreignization" in translations, e.g., the German Romanticists, especially the translations from Greek by Friedrich Hölderlin (Steiner, 1975) and Nabokov's translation of Eugene Onegin. These, however, as both Steiner (1975) and Venuti (2008) argue, are the exception to the rule. In fact, in recent years, the quality of translations has been standardized (ISO 17100). Importantly, the translations we studied in our work conform to this standard.

Despite the continuous efforts of translators, translations are known to feature unique characteristics that set them apart from non-translated texts, referred to as *originals* here (Toury, 1980, 1995; Frawley, 1984; Baker, 1993). This is not the result of poor translation, but rather a statistical phenomenon: various features distribute differently in originals than in translations (Gellerstam, 1986).

Several factors may account for the differences between originals and translations; many are classified as *universal* features of translation. Cognitively speaking, all translations, regardless of the source and target language, are susceptible to the same constraints. Therefore, translation products are expected to share similar artifacts. Such universals include *simplification*: the tendency to make complex source structures simpler in the target (Blum-Kulka and Levenston, 1983; Vanderauwerea, 1985); *standardization*: the tendency to over-conform to target language standards (Toury, 1995); and *explicitation*: the tendency to render implicit source structures more explicit in the target language (Blum-Kulka, 1986; Øverås, 1998).

In contrast to translation universals, *interference* reflects the "fingerprints" of the source lan-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 530–540
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1049

guage on the translation product. Toury (1995) defines interference as "phenomena pertaining to the make-up of the source text tend to be transferred to the target text". Interference, by definition, is a language-pair specific phenomenon; isomorphic structures shared by the source and target languages can easily replace one another, thereby manifesting the underlying process of cross-linguistic influence of the source language on the translation outcome. Pym (2008) points out that interference is a set of both *segmentational and macrostructural features*.

Our main hypothesis is that, due to interference, languages with shared isomorphic structures are likely to share more features in the target language of a translation. Consequently, the distance between two languages, when assessed using such features, can be retained to some extent in translations from these two languages to a third one. Furthermore, we hypothesize that by extracting structures from translated texts, we can generate a phylogenetic tree that reflects the "true" distances among the source languages. Finally, we conjecture that the quality of such trees will improve when constructed using features that better correspond to interference phenomena, and will deteriorate using more universal features of translation.

The main contribution of this paper is thus the demonstration that interference phenomena in translation are powerful to an extent that facilitates clustering source languages into families and (partially) reconstructing intra-families ties; so much so, that these results hold even after two rounds of translation. Moreover, we perform analysis of various linguistic phenomena in the source languages, laying out quantitative grounds for the language typology reconstruction results.

2 Related work

A number of works in historical linguistics have applied methods from the field of bioinformatics, in particular algorithms for generating phylogenetic trees (Ringe et al., 2002; Nakhleh et al., 2005a,b; Ellison and Kirby, 2006; Boc et al., 2010). Most of them rely on lists of *cognates*, words in multiple languages with a common origin that share a similar meaning and a similar pronunciation (Dyen et al., 1992; Rexová et al., 2003). These works all rely on multilingual data, whereas we construct phylogenetic trees from texts in a single language.

The claim that translations exhibit unique properties is well established in translation studies literature (Toury, 1980; Frawley, 1984; Baker, 1993; Toury, 1995). Based on this assumption, several works use text classification techniques employing supervised, and recently also unsupervised, machine learning approaches, to distinguish between originals and translations (Baroni and Bernardini, 2006; Ilisei et al., 2010; Koppel and Ordan, 2011; Volansky et al., 2015; Rabinovich and Wintner, 2015; Avner et al., 2016). The features used in these studies reflect both universal and interference-related traits. Along the way, interference was proven to be a robust phenomenon, operating in every single sentence, even on the morpheme level (Avner et al., 2016). Interference can also be studied on pairs of source- and target languages and focus, for example, on word order (Eetemadi and Toutanova, 2014).

The powerful signal of interference is evident, e.g., by the finding that a classifier trained to distinguish between originals and translations from one language, exhibits lower accuracy when tested on translations from another language, and this accuracy deteriorates proportionally to the distance between the source and target languages (Koppel and Ordan, 2011). Consequently, it is possible to accurately distinguish among translations from various source languages (van Halteren, 2008).

A related task, identifying the native tongue of English language students based only on their writing in English, has been the subject of recent interest (Tetreault et al., 2013). The relations between this task and identification of the source language of translation has been emphazied, e.g., by Tsvetkov et al. (2013). English texts produced by native speakers of a variety of languages have been used to reconstruct phylogenetic trees, with varying degrees of success (Nagata and Whittaker, 2013; Berzak et al., 2014). In contrast to language learners, however, translators translate into their mother tongue, so the texts we studied were written by highly competent native speakers. Our work is the first to construct phylogenetic trees from translations.

3 Methodology

3.1 Dataset

This corpus-based study uses Europarl (Koehn, 2005), the proceedings of the European Parliament and their translations into all the official Eu-

ropean Union (EU) languages. Europarl is one of the most popular parallel resources in natural language processing, and has been used extensively in machine translation. We use a version of Europarl spanning the years 1999 through 2011, in which the direction of translation has been established through a comprehensive cross-lingual validation of the speakers' original language (Rabinovich et al., 2015).

All parliament speeches were translated[1] from the original language into all other EU languages (21 at the time) using English as an intermediate, *pivot* language. We thus refer to translations into English as *direct*, while translations into all other languages, via English as a third language, are *indirect*. We hypothesize that indirect translation will obscure the markers of the original language in the final translation. Nevertheless, we expect (weakened) fingerprints of the source language to be identifiable in the target despite the pivot, presumably resulting in somewhat poorer phylogenetic trees.

We focus on 17 source languages, grouped into 3 language families: Germanic, Romance, and Balto-Slavic.[2] These include translations to English and to French from Bulgarian (BG), Czech (CS), Danish (DA), Dutch (NL), English (EN), French (FR), German (DE), Italian (IT), Latvian (LV), Lithuanian (LT), Polish (PL), Portuguese (PT), Romanian (RO), Slovak (SK), Slovenian (SL), Spanish (ES), and Swedish (SV). We also included texts written originally in English and French.

All datasets were split on sentence boundary, cleaned (empty lines removed), tokenized, and annotated for part-of-speech (POS) using the Stanford tools (Manning et al., 2014). In all the tree reconstruction experiments, we sampled equal-sized chunks from each source language, using as much data as available for all languages. This yielded $27,000$ tokens from translations to English, and $30,000$ tokens from translations into French.

[1]The common practice is that one translates into one's native language; in particular, this practice is strictly imposed in the EU parliament where a translator must have perfect proficiency in the target language, meeting very high standards of accuracy.

[2]We excluded source languages with insufficient amounts of data, along with Greek, which is the only representative of the Hellenic family.

3.2 Features

Following standard practice (Volansky et al., 2015; Rabinovich and Wintner, 2015), we represented both original and translated texts as feature vectors, where the choice of features determines the extent to which we expect source-language interference to be present in the translation product. Crucially, the features abstract away from the contents of the texts and focus on their structure, reflecting, among other things, morphological and syntactic patterns. We use the following feature sets: 1. The top-1,000 most frequent POS trigrams, reflecting shallow syntactic structure. 2. Function words (FW), words known to reflect grammar of texts in numerous classification tasks, as they include non-content words such as articles, prepositions, etc. (Koppel and Ordan, 2011).[3] 3. Cohesive markers (Hinkel, 2001); these words and phrases are assumed to be over-represented in translated texts, where, for example, an implicit contrast in the original is made explicit in the target text with words such as *'but'* or *'however'*.[4] Note that the first two feature sets are strongly associated with interference, whereas the third is assumed to be universal and an instance of explicitation. We therefore expect trees based on the first two feature sets to be much better than those based on the third.

3.3 The Indo-European phylogenetic tree

The last few decades produced a large body of research on the evolution of individual languages and language families. While the existence of the Indo-European (IE) family of languages is an established fact, its history and origins are still a matter of much controversy (Pereltsvaig and Lewis, 2015). Furthermore, the actual subgroupings of languages within this family are not clear-cut (Ringe et al., 2002). Consequently, algorithms that attempt to reconstruct the IE languages tree face a serious evaluation challenge (Ringe et al., 2002; Rexová et al., 2003; Nakhleh et al., 2005a).

To evaluate the quality of the reconstructed trees, we define a metric to accurately assess their distance from the "true" tree. The tree that we use as ground truth (Serva and Petroni, 2008) has

[3]For French we used the list of FW available at https://code.google.com/archive/p/stop-words/.

[4]For French we used http://utilisateurs.linguist.univ-paris-diderot.fr/~croze/D/Lexconn.xml.

several advantages. First, it is similar to a well-accepted tree (Gray and Atkinson, 2003) (which is not insusceptible to criticism (Pereltsvaig and Lewis, 2015)). The differences between the two are mostly irrelevant for the group of languages that we address in this research. Second, it is a binary tree, facilitating comparison with the trees we produce, which are also binary branching. Third, its branches are decorated with the approximate year in which splitting occurred. This provides a way to induce the distance between two languages, modeled as lengths of paths in the tree, based on chronological information.

We projected the gold tree (Serva and Petroni, 2008) onto the set of 17 languages we considered in this work, preserving branch lengths. Figure 1 depicts the resulting gold-standard subtree.

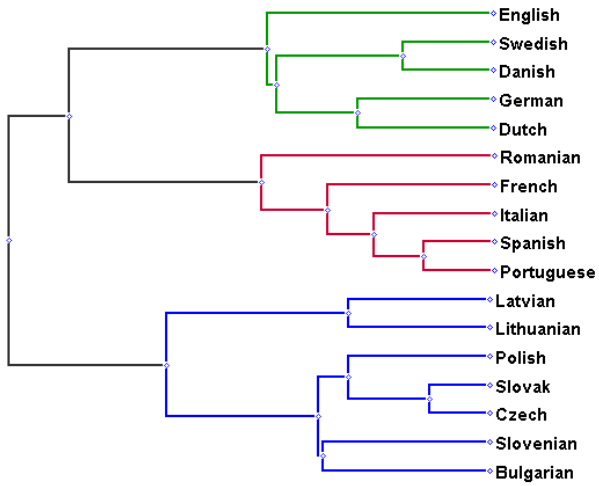


Figure 1: Gold standard tree, pruned

We reconstructed phylogenetic language trees by performing agglomerative (hierarchical) clustering of feature vectors extracted separately from English and French translations. We performed clustering using the variance minimization algorithm (Ward Jr, 1963) with Euclidean distance (the implementation available in the Python SciPy library). All feature values were normalized to a zero-one scale prior to clustering.

3.4 Evaluation methodology

To evaluate the quality of the trees we generate, we compute their similarity to the gold standard via two metrics: *unweighted*, assessing only structural (topological) similarity, and *weighted*, estimating similarity based on both structure and branching length.

Several methods have been proposed for evaluating the quality of phylogenetic language trees (Pompei et al., 2011; Wichmann and Grant, 2012; Nouri and Yangarber, 2016). A popular metric is the Robinson-Foulds (RF) methodology (Robinson and Foulds, 1981), which is based on the symmetric difference in the number of *bi-partitions*, the ways in which an edge can split the leaves of a tree into two sets. The distance between two trees is then defined as the number of splits induced by one of the trees, but not the other. Despite its popularity, the RF metric has well-known shortcomings; for example, relocating a single leaf can result in a tree maximally distant from the original one (Böcker et al., 2013). Additional methodologies for evaluating phylogenetic trees include *branch score distance* (Kuhner and Felsenstein, 1994), enhancing RF with branch lengths, *purity score* (Heller and Ghahramani, 2005), and *subtree score* (Teh et al., 2009). The latter two ignore branch lengths and only consider structural similarities for evaluation.

We opted for a simple yet powerful adaptation of the L2-norm to leaf-pair distance, inherently suitable for both unweighted and weighted evaluation. Given a tree of N leaves, l_i, $i \in [1..N]$, the *weighted distance* between two leaves l_i, l_j in a tree τ, denoted $D_\tau(l_i, l_j)$, is the sum of the weights of all edges on the shortest path between l_i and l_j. The *unweighted distance* sums up the *number* of the edges in this path (i.e., all weights are equal to 1). The distance $Dist(\tau, g)$ between a generated tree τ and the gold tree g is then calculated by summing the square differences between all leaf-pair distances (whether weighted or unweighted) in the two trees:

$$Dist(\tau, g) = \sum_{i,j \in [1..N]; i \neq j} (D_\tau(l_i, l_j) - D_g(l_i, l_j))^2$$

4 Detection of Translations and their Source Language

4.1 Identification of translation

We first reconfirmed that originals and translations are easily separable, extending results of supervised classification of O vs. T (where O refers to original English texts, and T to translated English) (Baroni and Bernardini, 2006; van Halteren, 2008; Volansky et al., 2015) to the 16 original languages considered in this work. We also conducted similar experiments with French originals and translations. We used 200 chunks of approximately 2K

tokens (respecting sentence boundaries) from both O and T, and normalized the values of lexical features by the number of tokens in each chunk. For classification, we used Platt's sequential minimal optimization algorithm (Keerthi et al., 2001; Hall et al., 2009) to train support vector machine classifiers with the default linear kernel. We evaluated the results with 10-fold cross-validation.

Table 1 presents the classification accuracy of (English and French) O vs. T using each feature set. In line with previous works (Ilisei et al., 2010; Volansky et al., 2015; Rabinovich and Wintner, 2015), the binary classification results are highly accurate, achieving over 95% accuracy using POS-trigrams and function words for both English and French, and above 85% using cohesive markers.

Feature	English	French
POS-trigrams	97.60	98.40
Function words	96.45	95.15
Cohesive markers	86.50	85.25

Table 1: Classification accuracy (%) of English and French O vs. T

4.2 Identification of source language

Identifying the source language of translated texts is a task in which machines clearly outperform humans (Baroni and Bernardini, 2006). Koppel and Ordan (2011) performed 5-way classification of texts translated from Italian, French, Spanish, German, and Finnish, achieving an accuracy of 92.7%. Furthermore, misclassified instances were more frequently assigned to genetically related languages.

We extended this experiment to 14 languages representing 3 language families (the number of languages was limited by the amount of data available). We extracted 100 chunks of 1,000 tokens each from each source language and classified the translated English (and, separately, French) texts into 14 classes using the best performing POS-trigrams feature set. Cross-validation evaluation yielded an accuracy of 75.61% on English translations (note that the baseline is $100/14 = 7.14\%$).

The corresponding confusion matrix, presented in Figure 2 (left), reveals interesting phenomena: much of the confusion resides within language families, framed by the bold line in the figure. For example, instances of Germanic languages are almost perfectly classified as Germanic, with only a few chunks assigned to other language families. The evident intra-family linguistic ties exposed by this experiment support the intuition that cross-linguistic transfer in translation is governed by typological properties of the source language. That is, translations from *related* sources tend to resemble each other to a greater extent than translations from more *distant* languages.

This observation is further supported by the evaluation of a three-way classification task, where the goal is to only identify the language family (Germanic, Romance, or Balto-Slavic): the accuracy of this task is 90.62%. Note also that the mis-classified instances of both Romance and Germanic languages are nearly never attributed to Balto-Slavic languages, since Germanic and Romance are much closer to each other than to Balto-Slavic.

Figure 2 (right) displays a similar confusion matrix, the only difference being that *French* translations are classified. We attribute the lower cross-validation accuracy (48.92%, reflected also by the lower number of correctly assigned instances on the matrix diagonal, compared to English) to the intervention of the pivot language in the translation process. Nevertheless, the confusion is still mainly constrained to intra-family boundaries.

5 Reconstruction of Phylogenetic Language Trees

5.1 Reconstructing language typology

Inspired by the results reported in Section 4.2, we generated phylogenetic language trees from both English and French texts translated from the other European languages. We hypothesized that interference from the source language was present in the translation product to an extent that would facilitate the construction of a tree sufficiently similar to the gold IE tree (Figure 1).

The best trees, those closest to the gold standard, were generated using POS-trigrams: these are the features that are most closely associated with source-language interference (see Section 3.2). Figure 3 depicts the trees produced from English and French translations using POS-trigrams. Both trees reasonably group individual languages into three language-family branches. In particular, they cluster the Germanic and Romance languages closer than the Balto-Slavic. Capturing the more subtle intra-family ties turned out to be

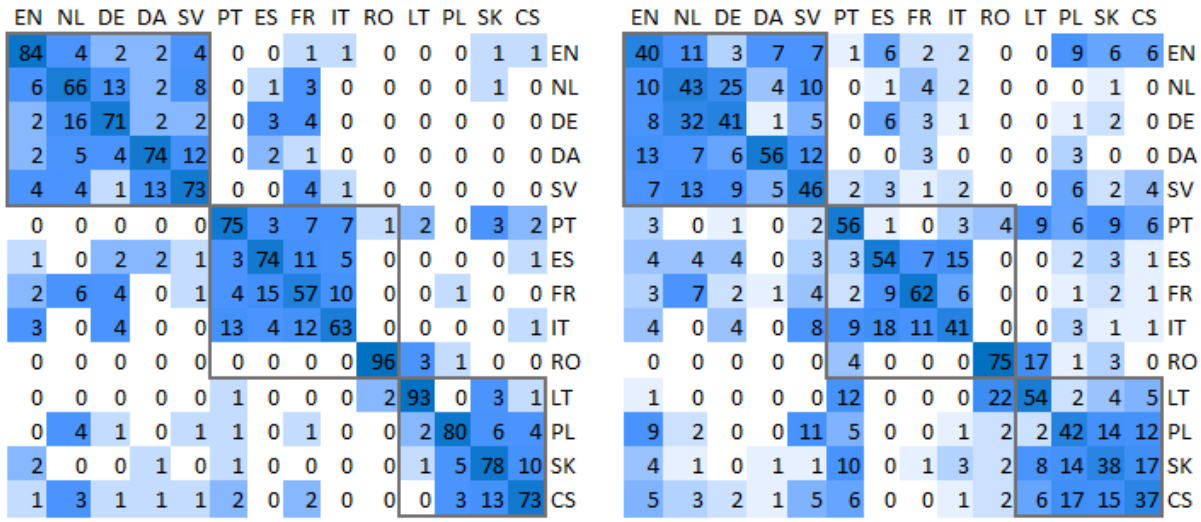


Figure 2: Confusion matrix of 14-way classification of English (left) and French (right) translations. The actual class is represented by rows and the predicted one by columns.

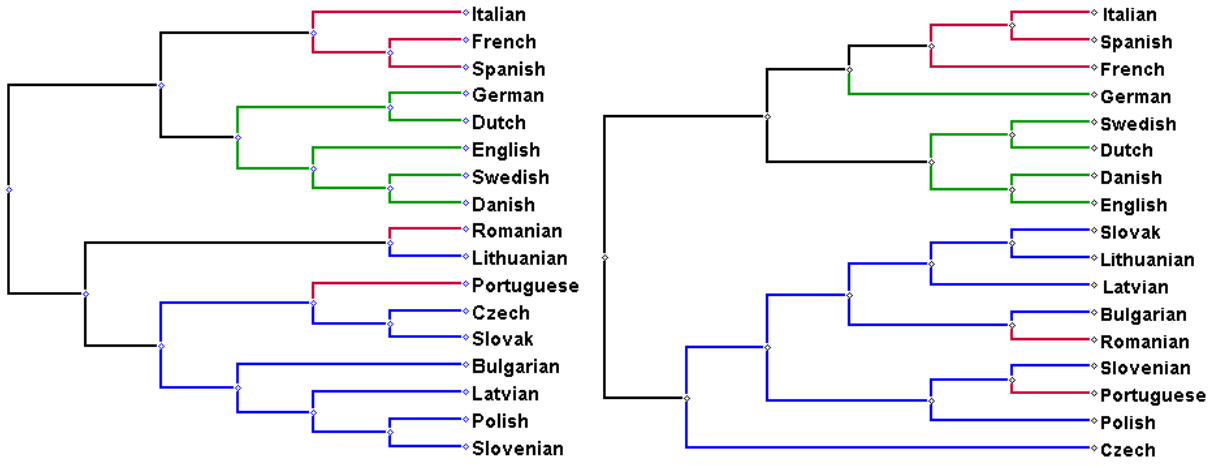


Figure 3: Phylogenetic language trees generated with English (left) and French (right) translations

more challenging, although English outperformed its French counterpart on this task by almost perfectly reconstructing the Germanic sub-tree.

We repeated the clustering experiments with various feature sets. For each feature set, we randomly sampled equally-sized subsets of the dataset (translated from each of the source languages), represented the data as feature vectors, generated a tree by clustering the feature vectors, and then computed the weighted and unweighted distances between the generated tree and the gold standard. We repeated this procedure 50 times for each feature set, and then averaged the resulting distances. We report this average and the standard deviation.[5]

5.2 Evaluation results

The *unweighted* evaluation results are listed in Table 2. For comparison, we also present the distance obtained for a random tree, generated by sampling a random distance matrix from the uniform $(0, 1)$ distribution. The reported random tree evaluation score is averaged over 1000 experiments. Similarly, we present *weighted* evaluation results in Table 3. All distances are normalized to a zero-one scale, where the bounds – zero and one – represent the identical and the most distant tree w.r.t. the gold standard, respectively.

The results reveal several interesting observations. First, as expected, POS-trigrams induce

[5]All the trees, both cladograms (with branches of equal length) and phylograms (with branch lengths proportional to the distance between two nodes), can be found at http://cl.haifa.ac.il/projects/translationese/acl2017_found-in-translation_trees.pdf

Target language	English		French	
Feature	AVG	STD	AVG	STD
POS-trigrams + FW	0.362	0.07	**0.367**	0.06
POS-trigrams	**0.353**	0.06	0.399	0.08
Function words	0.429	0.07	0.450	0.08
Cohesive markers	0.626	0.16	0.678	0.14
Random tree	0.724	0.07	0.724	0.07

Table 2: Unweighted evaluation of generated trees. AVG represents the average distance of a tree from the gold standard. The lowest distance in a column is boldfaced.

Target language	English		French	
Feature	AVG	STD	AVG	STD
POS-trigrams + FW	**0.278**	0.03	**0.348**	0.02
POS-trigrams	0.301	0.03	0.351	0.03
Function words	0.304	0.03	0.376	0.05
Cohesive markers	0.598	0.12	0.636	0.07
Random tree	0.676	0.10	0.676	0.10

Table 3: Weighted evaluation of generated trees. AVG represents the average distance of a tree from the gold standard. The lowest distance in a column is boldfaced.

trees closest to the gold standard among *distinct* feature sets. This corroborates our hypothesis that this feature set carries over interference of the source language to a considerable extent (see Section 1). Furthermore, function words achieve more moderate results, but still much better than random. This reflects the fact that these features carry over some grammatical constructs of the source language into the translation product.

Finally, in all cases, the least accurate tree, nearly random, is produced by cohesive markers; this is an evidence that this feature is source-language agnostic and reflects the universal effect of explicitation (see Section 3.2). While cohesive markers are a good indicator of translations, they reflect properties that are not indicative of the source language. The combination of POS-trigrams and FW yields the best tree in three out of four cases, implying that these feature sets capture different, complementary aspects of the source-language interference.

Surprisingly, reasonably good trees were also generated from French translations; yet, these trees are systematically worse than their English counterparts. The original signal of the source language is distorted twice: first via a Germanic language (English) and then via a Romance language (French). However, the signal is strong enough to yield a clear phylogenetic tree of the source languages. Interference is thus revealed to be an extremely powerful force, partially resistant to intermediate distortions.

6 Analysis

We demonstrated that source-language traces are dominant in translation products to an extent that facilitates reconstruction of the history of the source languages. We now inspect some of these phenomena in more detail to better understand the prominent characteristics of interference. For each phenomenon, we computed the frequencies of patterns that reflect it in texts translated to English from each individual language, and averaged the measures over each language family (Germanic, Romance, and Balto-Slavic). Figure 4 depicts the results.

6.1 Definite articles

Languages vary greatly in their use of articles. Like other Germanic languages, English has both definite (*'a'*) and indefinite (*'the'*) articles. However, many languages only have definite articles and some only have indefinite articles. Romance languages, and in particular the five Romance languages of our dataset, have definite articles that can sometimes be omitted, but not as commonly as in English. Balto-Slavic languages typically do not have any articles.

Mastering the use of articles in English is notoriously hard, leading to errors in non-native speakers (Han et al., 2006). For example, native speakers of Slavic languages tend to *over*use definite articles in German (Hirschmann et al., 2013). Similarly, we expect translations from Balto-Slavic languages to overuse *'the'*. We computed the frequencies of *'the'* in translations to English from each of the three language families. The results show a significant overuse of *'the'* in translations from Balto-Slavic languages, and some overuse in translations from Romance languages.

6.2 Possessive constructions

Languages also vary in the way they mark possession. English marks it in three ways: with the clitic *''s'* (*'the guest's room'*), with a prepositional phrase containing *'of'* (*'the room of the guest'*), and, like in other Germanic languages, with noun compounds (*'guest room'*). Compounds are considerably less frequent in Romance languages

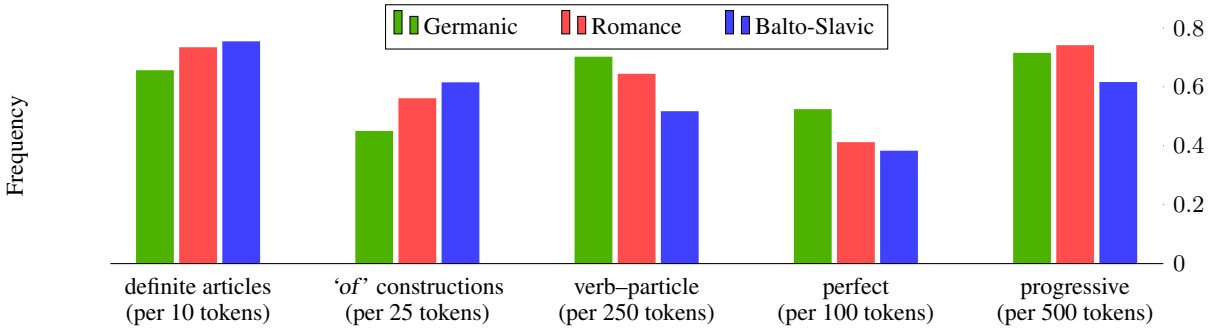


Figure 4: Frequencies reflecting various linguistic phenomena (Sections 6.1– 6.4) in English translations

(Swan and Smith, 2001); Balto-Slavic indicates possession using case-marking. Languages also vary with respect to whether or not possession is head-marked. In Balto-Slavic languages, the genitive case is head-marked, which reverses the order of the two nouns with respect to the common English *''s'* construction. Since copying word order, if possible across languages, is one of the major features of interference (Eetemadi and Toutanova, 2014), we anticipated that Balto-Slavic languages will exhibit the highest rate of noun-*'of'*-NP constructions. This would be followed by Romance languages, in which this construction is highly common, and then by Germanic languages, where noun compounds can often be copied as such. The results are consistent with our expectations.

6.3 Verb-particle constructions

Verb-particle constructions (e.g., *'turn down'*) consist of verbs that combine with a particle to create a new meaning (Dehé et al., 2002). Such constructions are much more common in Germanic languages (Iacobini and Masini, 2005), hence we expect to encounter their equivalents in English translations more frequently. We computed the frequencies of these constructions in the data; the results show a clear overuse of verb-particle constructions in translations from Germanic, and an underuse of such constructions in translations from Balto-Slavic.

6.4 Tense and aspect

Tense and aspect are expressed in different ways across languages. English, like other Germanic languages, uses a full system of aspectual distinctions, expressed via perfect and progressive forms (with the auxiliary verbs *'have'* or *'be'*). Balto-Slavic, in contrast, has no such system, and the distinction is marked lexically, by having two types of verbs. Romance languages are in between, with both lexical and grammatical distinctions. We computed the frequencies of perfect forms (defined as the auxiliary *'have'* followed by the past participle form), and the progressive forms (defined as the auxiliary *'be'* plus a present participle form). Indeed, Germanic overuses the perfect aspect significantly; the use of the progressive aspect also varies across language families, exhibiting the lowest frequency in translations from Balto-Slavic.

7 Conclusion

Translations may be considered distortions of the original text, but this distortion is far from random. It depicts a very clear picture, reflecting language typology to the extent that disregarding the sources altogether, a phylogenetic tree can be reconstructed from a monolingual corpus consisting of multiple translations. This holds for the product of highly professional translators, who conform to a common standard, and whose products are edited by native speakers, like themselves. It even holds after two phases of translations. We are presently trying to extend these results to translations in a different domain (literary texts) into a very different language (Hebrew).

Postulated universals in linguistics (Greenberg, 1963) were confronted with much contradicting evidence in recent years (Evans and Levinson, 2009), and the long quest for translation universals (Mauranen and Kujamäki, 2004) should now be viewed in light of our finding: more than anything else, translations are typified by interference. This does not undermine the force of translation universals: we demonstrated how explicitation, in the form of cohesive markers, can help identify translations. It may be possible to define classi-

fiers implementing other universal facets of translation, e.g., simplification, which will yield good separation between O and T. However, explicitation fails in the reproduction of language typology, whereas interference-based features produce trees of considerable quality.

Remarkably, translations to contemporary English and French capture part of the millennium-old history of the source languages from which the translations were made. Our trees reflect some of the historical connections among the languages, but of course they are related in other ways, too (whether incidental, areal, etc.). This may explain the case of Romanian in our reconstructed trees: it has been isolated for many years from other Romance languages and was under heavy influence from Balto-Slavic languages.

Very little research has been done in historical linguistics on how translations impact the evolvement of languages. The major trends relate to loan translations (Jahr, 1999), or the impact of canonical texts, such as Luther's translation of the Bible to German (Russ, 1994) or the case of the King James translation to English (Crystal, 2010). It has been attested that for certain languages, up to 30% of published materials are mediated through translation (Pym and Chrupała, 2005). Given the fingerprints left on target language texts, translations very likely play a role in language change. We leave this as a direction for future research.

Acknowledgements

We wish to thank the three ACL anonymous reviewers for their constructive feedback. We are grateful to Sergiu Nisioi and Oren Weimann for their advice and helpful suggestions. We are also thankful to Yonatan Belinkov and Michael Katz for insightful and valuable comments.

References

Ehud Alexander Avner, Noam Ordan, and Shuly Wintner. 2016. Identifying translationese at the word and sub-word level. *Digital Scholarship in the Humanities* 31(1):30–54. http://dx.doi.org/10.1093/llc/fqu047.

Mona Baker. 1993. Corpus linguistics and translation studies: Implications and applications. In Mona Baker, Gill Francis, and Elena Tognini-Bonelli, editors, *Text and technology: in honour of John Sinclair*, John Benjamins, Amsterdam, pages 233–252.

Marco Baroni and Silvia Bernardini. 2006. A new approach to the study of Translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing* 21(3):259–274.

Yevgeni Berzak, Roi Reichart, and Boris Katz. 2014. Reconstructing native language typology from foreign language usage. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. pages 21–29. http://aclweb.org/anthology/W/W14/W14-1603.pdf.

Shoshana Blum-Kulka. 1986. Shifts of cohesion and coherence in translation. In Juliane House and Shoshana Blum-Kulka, editors, *Interlingual and intercultural communication Discourse and cognition in translation and second language acquisition studies*, Gunter Narr Verlag, volume 35, pages 17–35.

Shoshana Blum-Kulka and Eddie A. Levenston. 1983. Universals of lexical simplification. In Claus Faerch and Gabriele Kasper, editors, *Strategies in Interlanguage Communication*, Longman, pages 119–139.

Alix Boc, Anna Maria Di Sciullo, and Vladimir Makarenkov. 2010. Classification of the Indo-European languages using a phylogenetic network approach. In Hermann Locarek-Junge and Claus Weihs, editors, *Classification as a Tool for Research: Proceedings of the 11th IFCS Biennial Conference and 33rd Annual Conference of the Gesellschaft für Klassifikation e.V., Dresden, March 13-18, 2009*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 647–655.

Sebastian Böcker, Stefan Canzar, and Gunnar W Klau. 2013. The generalized Robinson-Foulds metric. In *International Workshop on Algorithms in Bioinformatics*. Springer, pages 156–169.

David Crystal. 2010. *Begat: The King James Bible and the English Language*. Oxford University Press.

Nicole Dehé, Ray Jackendoff, Andrew McIntyre, and Silke Urban, editors. 2002. *Verb-particle Explorations*. Interface explorations. Mouton de Gruyter.

Isidore Dyen, Joseph B. Kruskal, and Paul Black. 1992. An Indoeuropean classification. a lexicostatistical experiment. *Transactions of the American Philosophical Society* 82(5):iii–132.

Sauleh Eetemadi and Kristina Toutanova. 2014. Asymmetric features of human generated translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 159–164. http://www.aclweb.org/anthology/D14-1018.

T. Mark Ellison and Simon Kirby. 2006. Measuring language divergence by intra-lexical comparison. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th*

Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 273–280. https://doi.org/10.3115/1220175.1220210.

Nicholas Evans and Stephen Levinson. 2009. The myth of language universals: Language diversity and its importance for cognitive science. *Behavioral and Brain Sciences* 32(5):429–494.

William Frawley. 1984. Prolegomenon to a theory of translation. In William Frawley, editor, *Translation. Literary, Linguistic and Philosophical Perspectives*, University of Delaware Press, Newark, pages 159–175.

Martin Gellerstam. 1986. Translationese in Swedish novels translated from English. In Lars Wollin and Hans Lindquist, editors, *Translation Studies in Scandinavia*, CWK Gleerup, Lund, pages 88–95.

Russell D. Gray and Quentin D. Atkinson. 2003. Language-tree divergence times support the Anatolian theory of Indo-European origin. *Nature* 426:435–439.

Joseph H. Greenberg, editor. 1963. *Universals of Human Language*. MIT Press, Cambridge, Mass.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Explorations* 11(1):10–18. https://doi.org/10.1145/1656274.1656278.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering* 12(02):115–129.

Katherine A Heller and Zoubin Ghahramani. 2005. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*. ACM, pages 297–304.

Eli Hinkel. 2001. Matters of cohesion in L2 academic texts. *Applied Language Learning* 12(2):111–132.

Hagen Hirschmann, Anke Lüdeling, Ines Rehbein, Marc Reznicek, and Amir Zeldes. 2013. Underuse of syntactic categories in Falko. a case study on modification. In Sylviane Granger, Gaëtanelle Gilquin, and Fanny Meunier, editors, *20 Years of Learner Corpus Research. Looking Back, Moving Ahead.*, Presses Universitaires de Louvain, Louvain la Neuve, pages 223–234.

Claudio Iacobini and Francesca Masini. 2005. Verb-particle constructions and prefixed verbs in Italian: typology, diachrony and semantics. In *Mediterranean Morphology Meetings*. volume 5, pages 157–184.

Iustina Ilisei, Diana Inkpen, Gloria Corpas Pastor, and Ruslan Mitkov. 2010. Identification of translationese: A machine learning approach. In Alexander F. Gelbukh, editor, *Proceedings of CICLing-2010: 11th International Conference on Computational Linguistics and Intelligent Text Processing*. Springer, volume 6008 of *Lecture Notes in Computer Science*, pages 503–511. http://dx.doi.org/10.1007/978-3-642-12116-6.

Ernst Håkon Jahr. 1999. *Language change: advances in historical sociolinguistics*, volume 114. Walter de Gruyter.

S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. 2001. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation* 13(3):637–649.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. MT Summit.

Moshe Koppel and Noam Ordan. 2011. Translationese and its dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 1318–1326. http://www.aclweb.org/anthology/P11-1132.

Mary K Kuhner and Joseph Felsenstein. 1994. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution* 11(3):459–468.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Anna Mauranen and Pekka Kujamäki, editors. 2004. *Translation universals: Do they exist?*. John Benjamins.

Ryo Nagata and Edward W. D. Whittaker. 2013. Reconstructing an Indo-European family tree from non-native English texts. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. pages 1137–1147. http://aclweb.org/anthology/P/P13/P13-1112.pdf.

Luay Nakhleh, Don Ringe, and Tandy Warnow. 2005a. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. *Language* 81(2):382–420.

Luay Nakhleh, Tandy Warnow, Don Ringe, and Steven N. Evans. 2005b. A comparison of phylogenetic reconstruction methods on an Indo-European dataset. *Transactions of the Philological Society* 103(2):171–192. https://doi.org/10.1111/j.1467-968X.2005.00149.x.

Javad Nouri and Roman Yangarber. 2016. Modeling language evolution with codes that utilize context and phonetic features. *CoNLL 2016* page 136.

Lin Øverås. 1998. In search of the third code: An investigation of norms in literary translation. *Meta* 43(4):557–570.

Asya Pereltsvaig and Martin W. Lewis. 2015. *The Indo-European Controversy*. Cambridge University Press, Cambridge.

Simone Pompei, Vittorio Loreto, and Francesca Tria. 2011. On the accuracy of language trees. *PloS one* 6(6):e20109.

Anthony Pym. 2008. On Toury's laws of how translators translate. *BENJAMINS TRANSLATION LIBRARY* 75:311.

Anthony Pym and Grzegorz Chrupała. 2005. The quantitative analysis of translation flows in the age of an international language. In Albert Branchadell and Lovell M. West, editors, *Less Translated Languages*, John Benjamins, Amsterdam, pages 27–38.

Ella Rabinovich and Shuly Wintner. 2015. Unsupervised identification of translationese. *Transactions of the Association for Computational Linguistics* 3:419–432.

Ella Rabinovich, Shuly Wintner, and Ofek Luis Lewinsohn. 2015. The Haifa corpus of translationese. Unpublished manuscript. http://arxiv.org/abs/1509.03611.

Kateřina Rexová, Daniel Frynta, and Jan Zrzavỳ. 2003. Cladistic analysis of languages: Indo-European classification based on lexicostatistical data. *Cladistics* 19(2):120–127.

Kateřina Rexová, Daniel Frynta, and Jan Zrzavý. 2003. Cladistic analysis of languages: Indo-European classification based on lexicostatistical data. *Cladistics-the International Journal of the Willi Hennig Society* 19(2):120–127.

Don Ringe, Tandy Warnow, and Ann Taylor. 2002. Indo-European and computational cladistics. *Transactions of the Philological Society* 100(1):59–129. https://doi.org/10.1111/1467-968X.00091.

David F Robinson and Leslie R Foulds. 1981. Comparison of phylogenetic trees. *Mathematical biosciences* 53(1):131–147.

Charles VJ Russ. 1994. *The German language today: A linguistic introduction*. Psychology Press.

Maurizio Serva and Filippo Petroni. 2008. Indo-European languages tree by Levenshtein distance. *Europhysics Letters* 81(6):68005. http://stacks.iop.org/0295-5075/81/i=6/a=68005.

George Steiner. 1975. *After Babel*. University Press.

Michael Swan and Bernard Smith. 2001. *Learner English*. Cambridge University Press, Cambridge, second edition.

Yee Whye Teh, Hal Daumé III, and Daniel Roy. 2009. Bayesian agglomerative clustering with coalescents. *arXiv preprint arXiv:0907.0781* .

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics.

Gideon Toury. 1980. *In Search of a Theory of Translation*. The Porter Institute for Poetics and Semiotics, Tel Aviv University, Tel Aviv.

Gideon Toury. 1995. *Descriptive Translation Studies and beyond*. John Benjamins, Amsterdam / Philadelphia.

Yulia Tsvetkov, Naama Twitto, Nathan Schneider, Noam Ordan, Manaal Faruqui, Victor Chahuneau, Shuly Wintner, and Chris Dyer. 2013. Identifying the L1 of non-native writers: the CMU-Haifa system. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, pages 279–287. http://www.aclweb.org/anthology/W13-1736.

Hans van Halteren. 2008. Source language markers in EUROPARL translations. In Donia Scott and Hans Uszkoreit, editors, *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*. pages 937–944. http://www.aclweb.org/anthology/C08-1118.

Ria Vanderauwerea. 1985. *Dutch novels translated into English: the transformation of a 'minority' literature*. Rodopi, Amsterdam.

Lawrence Venuti. 2008. *The translator's invisibility: A history of translation*. Routledge.

Vered Volansky, Noam Ordan, and Shuly Wintner. 2015. On the features of translationese. *Digital Scholarship in the Humanities* 30(1):98–118.

Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* 58(301):236–244.

Søren Wichmann and Anthony P Grant. 2012. *Quantitative approaches to linguistic diversity: commemorating the centenary of the birth of Morris Swadesh*, volume 46. John Benjamins Publishing.

Predicting Native Language from Gaze

Yevgeni Berzak
MIT CSAIL
berzak@mit.edu

Chie Nakamura
MIT Linguistics
chienak@mit.edu

Suzanne Flynn
MIT Linguistics
sflynn@mit.edu

Boris Katz
MIT CSAIL
boris@mit.edu

Abstract

A fundamental question in language learning concerns the role of a speaker's first language in second language acquisition. We present a novel methodology for studying this question: analysis of eye-movement patterns in second language reading of free-form text. Using this methodology, we demonstrate for the first time that the native language of English learners can be predicted from their gaze fixations when reading English. We provide analysis of classifier uncertainty and learned features, which indicates that differences in English reading are likely to be rooted in linguistic divergences across native languages. The presented framework complements production studies and offers new ground for advancing research on multilingualism.[1]

1 Introduction

The influence of a speaker's native language on learning and performance in a foreign language, also known as cross-linguistic transfer, has been studied for several decades in linguistics and psychology (Odlin, 1989; Martohardjono and Flynn, 1995; Jarvis and Pavlenko, 2008; Berkes and Flynn, 2012; Alonso, 2015). The growing availably of learner corpora has also sparked interest in cross-linguistic influence phenomena in NLP, where studies have explored the task of Native Language Identification (NLI) (Tetreault et al., 2013), as well as analysis of textual features in relation to the author's native language (Jarvis and Crossley, 2012; Swanson and Charniak, 2013; Malmasi and Dras, 2014). Despite these advances, the extent and nature of first language influence in second language processing remains far from being established. Crucially, most prior work on this topic focused on production, while little is currently known about cross-linguistic influence in language comprehension.

In this work, we present a novel framework for studying cross-linguistic influence in language comprehension using *eyetracking for reading* and *free-form native English text*. We collect and analyze English newswire reading data from 182 participants, including 145 English as Second Language (ESL) learners from four different native language backgrounds: Chinese, Japanese, Portuguese and Spanish, as well as 37 native English speakers. Each participant reads 156 English sentences, half of which are shared across all participants, and the remaining half are individual to each participant. All the sentences are manually annotated with part-of-speech (POS) tags and syntactic dependency trees.

We then introduce the task of *Native Language Identification from Reading (NLIR)*, which requires predicting a subject's native language from gaze while reading text in a second language. Focusing on ESL participants and using a log-linear classifier with word fixation times normalized for reading speed as features, we obtain 71.03 NLIR accuracy in the shared sentences regime. We further demonstrate that NLIR can be generalized effectively to the individual sentences regime, in which each subject reads a different set of sentences, by grouping fixations according to linguistically motivated clustering criteria. In this regime, we obtain an NLIR accuracy of 51.03.

Further on, we provide classification and feature analyses, suggesting that the signal underlying NLIR is likely to be related to *linguistic* characteristics of the respective native languages. First, drawing on previous work on ESL production, we

[1] The experimental data collected in this study will be made publicly available.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 541–551
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1050

observe that classifier uncertainty in NLIR correlates with global linguistic similarities across native languages. In other words, the more similar are the languages, the more similar are the reading patterns of their native speakers in English. Second, we perform feature analysis across native and non-native English speakers, and discuss structural and lexical factors that could potentially drive some of the non-native reading patterns in each of our native languages. Taken together, our results provide evidence for a systematic influence of native language properties on reading, and by extension, on online processing and comprehension in a second language.

To summarize, we introduce a novel framework for studying cross-linguistic influence in language learning by using eyetracking for reading free-form English text. We demonstrate the utility of this framework in the following ways. First, we obtain the first NLIR results, addressing both the shared and the individual textual input scenarios. We further show that reading preserves linguistic similarities across native languages of ESL readers, and perform feature analysis, highlighting key distinctive reading patterns in each native language. The proposed framework complements and extends production studies, and can inform linguistic inquiry on cross-linguistic influence.

This paper is structured as follows. In section 2 we present the data and our experimental setup. Section 3 describes our approach to NLIR and summarizes the classification results. We analyze cross-linguistic influence in reading in section 4. In section 4.1 we examine NLIR classification uncertainty in relation to linguistic similarities between native languages. In section 4.2 we discuss several key fixation features associated with different native languages. Section 5 surveys related work, and section 6 concludes.

2 Experimental Setup

Participants

We recruited 182 adult participants. Of those, 37 are native English speakers and 145 are ESL learners from four native language backgrounds: Chinese, Japanese, Portuguese and Spanish. All the participants in the experiment are native speakers of only one language. The ESL speakers were tested for English proficiency using the grammar and listening sections of the Michigan English test (MET), which consist of 50 multiple choice questions. The English proficiency score was calculated as the number of correctly answered questions on these modules. The majority of the participants scored in the intermediate-advanced proficiency range. Table 1 presents the number of participants and the mean English proficiency score for each native language group. Additionally, we collected metadata on gender, age, level of education, duration of English studies and usage, time spent in English speaking countries and proficiency in any additional language spoken.

	# Participants	English Score
Chinese	36	42.0
Japanese	36	40.3
Portuguese	36	41.1
Spanish	37	42.4
English	37	NA

Table 1: Number of participants and mean MET English score by native language group.

Reading Materials

We utilize 14,274 randomly selected sentences from the Wall Street Journal part of the Penn Treebank (WSJ-PTB) (Marcus et al., 1993). To support reading convenience and measurement precision, the maximal sentence length was set to 100 characters, leading to an average sentence length of 11.4 words. Word boundaries are defined as whitespaces. From this sentence pool, 78 sentences (900 words) were presented to all participants (henceforth *shared* sentences) and the remaining 14,196 sentences were split into 182 individual batches of 78 sentences (henceforth *individual* sentences, averaging 880 words per batch).

All the sentences include syntactic annotations from the Universal Dependency Treebank project (UDT) (McDonald et al., 2013). The annotations include PTB POS tags (Santorini, 1990), Google universal POS tags (Petrov et al., 2012) and dependency trees. The dependency annotations of the UDT are converted automatically from the manual phrase structure tree annotations of the WSJ-PTB.

Gaze Data Collection

Each participant read 157 sentences. The first sentence was presented to familiarize participants with the experimental setup and was discarded during analysis. The following 156 sentences consisted of 78 shared and 78 individual sen-

tences. The shared and the individual sentences were mixed randomly and presented to all participants in the same order. The experiment was divided into three parts, consisting of 52 sentences each. Participants were allowed to take a short break between experimental parts.

Each sentence was presented on a blank screen as a one-liner. The text appeared in Times font, with font size 23. To encourage attentive reading, upon completion of sentence reading participants answered a simple yes/no question about its content, and were subsequently informed if they answered the question correctly. Both the sentences and the questions were triggered by a 300ms gaze on a fixation target (fixation circle for sentences and the letter "Q" for questions) which appeared on a blank screen and was co-located with the beginning of the text in the following screen.

Throughout the experiment, participants held a joystick with buttons for indicating completion of sentence reading and answering the comprehension questions. Eye-movement of participants' dominant eye was recorded using a desktop mount Eyelink 1000 eyetracker, at a sampling rate of 1000Hz. Further details on the experimental setup are provided in appendix A.

3 Native Language Identification from Reading

Our first goal is to determine whether the native language of ESL learners can be decoded from their gaze patterns while reading English text. We address this question in two regimes, corresponding to our division of reading input into shared and individual sentences. In the *shared regime*, all the participants read the same set of sentences. Normalizing over the reading input, this regime facilitates focusing on differences in reading behavior across readers. In the *individual regime*, we use the individual batches from our data to address the more challenging variant of the NLIR task in which the reading material given to each participant is different.

3.1 Features

We seek to utilize features that can provide robust, simple and interpretable characterizations of reading patterns. To this end, we use speed normalized *fixation duration* measures over word sequences.

Fixation Measures

We utilize three measures of word fixation duration:

- *First Fixation duration (FF)* Duration of the first fixation on a word.
- *First Pass duration (FP)* Time spent from first entering a word to first leaving it (including re-fixations within the word).
- *Total Fixation duration (TF)* The sum of all fixation times on a word.

We experiment with fixations over unigram, bigram and trigram sequences $seq_{i,k} = w_i, ..., w_{i+k-1}, k \in \{1, 2, 3\}$, where for each metric $M \in \{FF, FP, TF\}$ the fixation time for a sequence $M_{seq_{i,k}}$ is defined as the sum of fixations on individual tokens M_w in the sequence[2].

$$M_{seq_{i,k}} = \sum_{w' \in seq_{i,k}} M_{w'} \quad (1)$$

Importantly, we control for variation in reading speeds across subjects by normalizing each subjects's sequence fixation times. For each metric M and sequence $seq_{i,k}$ we normalize the sequence fixation time $M_{seq_{i,k}}$ relative to the subject's sequence fixation times in the textual context of the sequence. The context C is defined as the sentence in which the sequence appears for the *Words in Fixed Context* feature-set and the entire textual input for the *Syntactic* and *Information* clusters feature-sets (see definitions of feature-sets below). The normalization term $S_{M,C,k}$ is accordingly defined as the metric's fixation time per sequence of length k in the context:

$$S_{M,C,k} = \frac{1}{|C|} \sum_{seq_k \in C} M_{seq_k} \quad (2)$$

We then obtain a normalized fixation time $Mnorm_{seq_{i,k}}$ as:

$$Mnorm_{seq_{i,k}} = \frac{M_{seq_{i,k}}}{S_{M,C,k}} \quad (3)$$

[2]Note that for bigrams and trigrams, one could also measure FF and FP for interest regions spanning the sequence, instead, or in addition to summing these fixation times over individual tokens.

Feature Types

We use the above presented speed normalized fixation metrics to extract three feature-sets, *Words in Fixed Context (WFC)*, *Syntactic Clusters* (SC) and *Information Clusters (IC)*. WFC is a token-level feature-set that presupposes a fixed textual input for all participants. It is thus applicable only in the shared sentences regime. SC and IC are type-level features which provide abstractions over sequences of words. Crucially, they can also be applied when participants read different sentences.

- **Words in Fixed Context (WFC)** The WFC features capture fixation times on word sequences in a specific sentence. This feature-set consists of FF, FP and TF times for each of the 900 unigram, 822 bigram, and 744 trigram word sequences comprising the shared sentences. The fixation times of each metric are normalized for each participant relative to their fixations on sequences of the same length in the surrounding sentence. As noted above, the WFC feature-set is not applicable in the individual regime, as it requires identical sentences for all participants.

- **Syntactic Clusters (SC)** CS features are average globally normalized FF, FP and TF times for word sequences clustered by our three types of syntactic labels: universal POS, PTB POS, and syntactic relation labels. An example of such a feature is the average of speed-normalized TF times spent on the PTB POS bigram sequence DT NN. We take into account labels that appear at least once in the reading input of all participants. On the four non-native languages, considering all three label types, we obtain 104 unigram, 636 bigram and 1,310 trigram SC features per fixation metric in the shared regime, and 56 unigram, 95 bigram and 43 trigram SC features per fixation metric in the individual regime.

- **Information Clusters (IC)** We also obtain average FF, FP and TF for words clustered according to their *length*, measured in number of characters. Word length was previously shown to be a strong predictor of *information content* (Piantadosi et al., 2011). As such, it provides an alternative abstraction to the syntactic clusters, combining both syntactic and lexical information. As with SC features, we take into account features that appear at least once in the textual input of all participants. For our set of non-native languages, we obtain for each fixation metric 15 unigram, 21 bigram and 23 trigram IC features in the shared regime, and 12 unigram, 18 bigram and 18 trigram IC features in the individual regime. Notably, this feature-set is very compact, and differently from the syntactic clusters, does not rely on the availability of external annotations.

In each feature-set, we perform a final preprocessing step for each individual feature, in which we derive a zero mean unit variance scaler from the training set feature values, and apply it to transform both the training and the test values of the feature to Z scores.

3.2 Model

The experiments are carried out using a log-linear model:

$$p(y|x;\theta) = \frac{\exp(\theta \cdot f(x,y))}{\sum_{y' \in Y} \exp(\theta \cdot f(x,y'))} \quad (4)$$

where y is the reader's native language, x is the reading input and θ are the model parameters. The classifier is trained with gradient descent using L-BFGS (Byrd et al., 1995).

3.3 Experimental Results

In table 2 we report 10-fold cross-validation results on NLIR in the shared and the individual experimental regimes for native speakers of Chinese, Japanese, Portuguese and Spanish. We introduce two baselines against which we compare the performance of our feature-sets. The *majority* baseline selects the native language with the largest number of participants. The *random clusters* baseline clusters words into groups randomly, with the number of groups set to the number of syntactic categories in our data.

In the shared regime, WFC fixations yield the highest classification rates, substantially outperforming the cluster feature-sets and the two baselines. The strongest result using this feature-set, 71.03, is obtained by combining unigram, bigram and trigram fixation times. In addition to this outcome, we note that training binary classifiers in this setup yields accuracies ranging from 68.49 for the language pair Portuguese and Spanish, to 93.15 for Spanish and Japanese. These results confirm the effectiveness of the shared input

	Shared Sentences Regime			Individual Sentences Regime		
Majority Class	25.52			25.52		
Random Clusters	22.76			22.07		
	unigrams	+bigrams	+trigrams	unigrams	+bigrams	+trigrams
Information Clusters (IC)	41.38	44.14	46.21	38.62	32.41	32.41
Syntactic Clusters (SC)	45.52	57.24	58.62	48.97	43.45	48.28
SC+IC	51.72	57.24	60.0	**51.03**	46.21	49.66
Words in Fixed Context (WFC)	64.14	68.28	**71.03**	NA		

Table 2: Native Language Identification from Reading results with 10-fold cross-validation for native speakers of Chinese, Japanese, Portuguese and Spanish. In the *Shared* regime all the participants read the same 78 sentences. In the *Individual* regime each participant reads a different set of 78 sentences.

regime for performing reliable NLIR, and suggest a strong native language signal in non-native reading fixation times.

SC features yield accuracies of 45.52 to 58.62 on the shared sentences, while IC features exhibit weaker performance in this regime, with accuracies of 41.38 to 46.21. Both results are well above chance, but lower than WFC fixations due to the information loss imposed by the clustering step. Crucially, both feature-sets remain effective in the individual input regime, with 43.45 to 48.97 accuracy for SC features and 32.41 to 38.62 accuracy for IC features. The strongest result in the individual regime is 51.03, obtained by concatenating IC and SC features over unigrams. We also note that using this setup in a binary classification scheme yields results ranging from chance level 49.31 for Portuguese versus Spanish, to 84.93 on Spanish versus Japanese.

Generally, we observe that adding bigram and trigram fixations in the shared regime leads to performance improvements compared to using unigram features only. This trend does not hold for the individual sentences, presumably due to a combination of feature sparsity and context variation in this regime. We also note that IC and SC features tend to perform better together than in separation, suggesting that the information encoded using these feature-sets is to some extent complementary.

The generalization power of our cluster based feature-sets has both practical and theoretical consequences. Practically, they provide useful abstractions for performing NLIR over arbitrary textual input. That is, they enable performing this task using *any* textual input during both training and testing phases. Theoretically, the effectiveness of linguistically motivated features in discerning native languages suggests that linguistic factors play an important role in the ESL reading process. The analysis presented in the following sections will further explore this hypothesis.

4 Analysis of Cross-Linguistic Influence in ESL Reading

As mentioned in the previous section, the ability to perform NLIR in general, and the effectiveness of linguistically motivated features in particular, suggest that linguistic factors in the native and second languages are pertinent to ESL reading. In this section we explore this hypothesis further, by analyzing classifier uncertainty and the features learned in the NLIR task.

4.1 Preservation of Linguistic Similarity

Previous work in NLP suggested a link between textual patterns in ESL production and linguistic similarities of the respective native languages (Nagata and Whittaker, 2013; Nagata, 2014; Berzak et al., 2014, 2015). In particular, Berzak et al. (2014) has demonstrated that NLI classification uncertainty correlates with similarities between languages with respect to their typological features. Here, we extend this framework and examine if preservation of native language similarities in ESL production is paralleled in reading.

Similarly to Berzak et al. (2014) we define the classification uncertainty for a pair of native languages y and y' in our data collection D, as the average probability assigned by the NLIR classifier to one language given the other being the true native language. This approach provides a robust measure of classification confusion that does not rely on the actual performance of the classifier. We interpret the classifier uncertainty as a similarity measure between the respective languages and de-

note it as English Reading Similarity ERS.

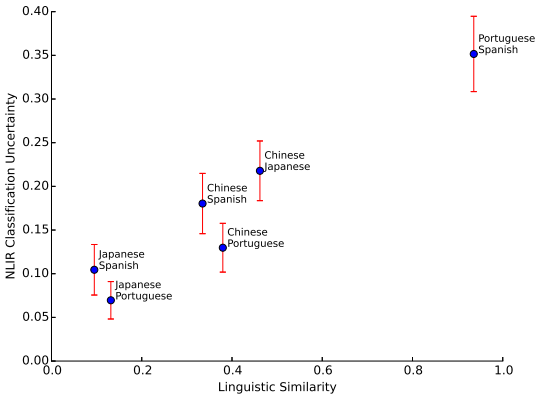
$$ERS_{y,y'} = \frac{\sum_{(x,y)\in D_y} p(y'|x;\theta) + \sum_{(x,y')\in D_{y'}} p(y|x;\theta)}{|D_y| + |D_{y'}|} \quad (5)$$

We compare these reading similarities to the linguistic similarities between our native languages. To approximate these similarities, we utilize feature vectors from the URIEL Typological Compendium (Littel et al., 2016) extracted using the *lang2vec* tool (Littell et al., 2017). URIEL aggregates, fuses and normalizes typological, phylogenetic and geographical information about the world's languages.

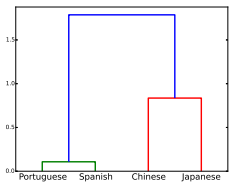
We obtain all the 103 available morpho-syntactic features in URIEL, which are derived from the World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013), Syntactic Structures of the World's Languages (SSWL) (Collins and Kayne, 2009) and Ethnologue (Lewis et al., 2015). Missing feature values are completed with a KNN classifier. We also extract URIEL's 3,718 language family features derived from Glottolog (Hammarström et al., 2015). Each of these features represents membership in a branch of Glottolog's world language tree. Truncating features with the same value for all our languages, we remain with 76 features, consisting of 49 syntactic features and 27 family tree features. The linguistic similarity LS between a pair of languages y and y' is then determined by the cosine similarity of their URIEL feature vectors.

$$LS_{y,y'} = \frac{v_y \cdot v_{y'}}{\|v_y\| \|v_{y'}\|} \quad (6)$$

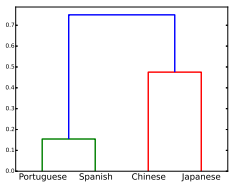
Figure 1 presents the URIEL based linguistic similarities for our set of non-native languages against the average NLIR classification uncertainties on the cross-validation test samples. The results presented in this figure are based on the unigram IC+SC feature-set in the individual sentences regime. We also provide a graphical illustration of the language similarities for each measure, using the Ward clustering algorithm (Ward Jr, 1963). We observe a correlation between the two measures which is also reflected in similar hierarchies in the two language trees. Thus, linguistically motived features in English reveal linguistic similarities across native languages. This outcome supports the hypothesis that English reading differences across native languages are related to linguistic factors.



(a) Linguistic similarities against mean NLIR classification uncertainty. Error bars denote standard error.



(b) Linguistic tree



(c) English reading tree

Figure 1: (a) Linguistic versus English reading language similarities. The horizontal axis represents typological and phylogenetic similarity between languages, obtained by vectorizing linguistic features form URIEL, and measuring their cosine similarity. The vertical axis represents the average uncertainty of the NLIR classifier in distinguishing ESL readers of each language pair. (b) Ward hierarchical clustering of linguistic similarities between languages. (c) Ward hierarchical clustering of NLIR average pairwise classification uncertainties.

We note that while comparable results are obtained for the IC and SC feature-sets, together and in separation in the shared regime, WFC features in the shared regime do not exhibit a clear uncertainty distinction when comparing across the pairs Japanese and Spanish, Japanese and Portuguese, Chinese and Spanish, and Chinese and Portuguese. Instead, this feature-set yields very low uncertainty, and correspondingly very high performance ranging from 90.41 to 93.15, for all four language pairs.

4.2 Feature Analysis

Our framework enables not only native language classification, but also exploratory analysis of native language specific reading patterns in English. The basic question that we examine in this respect is on which features do readers of different native language groups spend more versus less time. We also discuss several potential relations of the observed reading time differences to usage patterns and grammatical errors committed by speakers of our four native languages in production. We obtain this information by extracting grammatical error counts from the CLC FCE corpus (Yannakoudakis et al., 2011), and from the ngram frequency analysis in Nagata and Whittaker (2013).

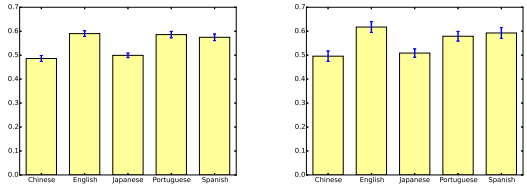
In order to obtain a common benchmark for reading time comparisons across non-native speakers, in this analysis we also consider our group of native English speakers. In this context, we train four binary classifiers that discern each of the non-native groups from native English speakers based on TF times over unigram PTB POS tags in the shared regime. The features with the strongest positive and negative weights learned by these classifiers are presented in table 3. These features serve as a reference point for selecting the case studies discussed below.

Interestingly, some of the reading features that are most predictive of each native language lend themselves to linguistic interpretation with respect to *structural* factors. For example, in Japanese and Chinese we observe shorter reading times for determiners (DT), which do not exist in these languages. Figure 2a presents the mean TF times for determiners in all five native languages, suggesting that native speakers of Portuguese and Spanish, which do have determiners, do not exhibit reduced reading times on this structure compared to natives. In ESL production, missing determiner errors are the most frequent error for native speakers of Japanese and third most common error for native speakers of Chinese.

In figure 2b we present the mean TF reading times for pronouns (PRP), where we also see shorter reading times by natives of Japanese and Chinese as compared to English natives. In both languages pronouns can be omitted both in object and subject positions. Portuguese and Spanish, in which pronoun omission is restricted to the subject position present similar albeit weaker tendency.

	Negative (Fast)	Positive (Slow)
Chinese	**DT**	JJR
	PRP	**NN**
Japanese	**DT**	**NN**
	CD	VBD
Portuguese	NNS	**NN-POS**
	PRP	VBZ
Spanish	NNS	MD
	PRP	RB

Table 3: PTB POS features with the strongest weights learned in non-native versus native classification for each native language in the shared regime. Feature types presented in figure 2 are highlighted in bold.



(a) Determiners (DT) (b) Pronouns (PRP)

(c) Possessives (NN+POS) (d) Nouns (NN)

Figure 2: Mean speed-normalized Total Fixation duration for Determiners (DT), Pronouns (PRP), singular noun possessives (NN+POS), and singular nouns (NN) appearing in the shared sentences. Error bars denote standard error.

In figure 2c we further observe that differently from natives of Chinese and Japanese, native speakers of Portuguese and Spanish spend more time on NN+POS in head final possessives such as "the *public's* confidence". While similar constructions exist in Chinese and Japanese, the NN+POS combination is expressed in Portuguese and Spanish as a head initial NN *of NN*. This form exists in English (e.g. "the confidence of the public") and is preferred by speakers of these languages in ESL writing (Nagata and Whittaker, 2013). As an additional baseline for this construction, we provide the TF times for NN in figure 2d. There, relative to English natives, we observe longer reading times for Japanese and Chinese and comparable times for Portuguese and Spanish.

The reading times of NN in figure 2d also give

rise to a second, potentially competing interpretation of differences in ESL reading times, which highlights *lexical* rather than structural factors. According to this interpretation, increased reading times of nouns are the result of substantially smaller lexical sharing with English by Chinese and Japanese as compared to Spanish and Portuguese. Given the utilized speed normalization, lexical effects on nouns could in principle account for reduced reading times on determiners and pronouns. Conversely, structural influence leading to reduced reading times on determiners and pronouns could explain longer dwelling on nouns. A third possibility consistent with the observed reading patterns would allow for both structural and lexical effects to impact second language reading. Importantly, in each of these scenarios, ESL reading patterns are related to linguistic factors of the reader's native language.

We note that the presented analysis is preliminary in nature, and warrants further study in future research. In particular, reading times and classifier learned features may in some cases differ between the shared and the individual regimes. In the examples presented above, similar results are obtained in the individual sentences regime for DT, PRP and NN. The trend for the NN+POS construction, however, diminishes in that setup with similar reading times for all languages. On the other hand, one of the strongest features for predicting Portuguese and Spanish in the individual regime are longer reading times for prepositions (IN), an outcome that holds in the shared regime only relative to Chinese and Japanese, but not relative to native speakers of English.

Despite these caveats, our results suggest that reading patterns can potentially be related to linguistic factors of the reader's native language. This analysis can be extended in various ways, such as inclusion of additional feature types and fixation metrics, as well as utilization of other comparative methodologies. Combined with evidence from language production, this line of investigation can be instrumental for informing linguistic theory of cross-linguistic influence.

5 Related Work

Eyetracking and second language reading Second language reading has been studied using eyetracking, with much of the work focusing on processing of syntactic ambiguities and analysis of specific target word classes such as cognates (Dussias, 2010; Roberts and Siyanova-Chanturia, 2013). In contrast to our work, such studies typically use controlled, rather than free-form sentences. Investigation of global metrics in free-form second language reading was introduced only recently by Cop et al. (2015). This study compared ESL and native reading of a novel by native speakers of Dutch, observing longer sentence reading times, more fixations and shorter saccades in ESL reading. Differently from this study, our work focuses on comparison of reading patterns between different native languages. We also analyze a related, but different metric, namely speed normalized fixation durations on word sequences.

Eyetracking for NLP tasks Recent work in NLP has demonstrated that reading gaze can serve as a valuable supervision signal for standard NLP tasks. Prominent examples of such work include POS tagging (Barrett and Søgaard, 2015a; Barrett et al., 2016), syntactic parsing (Barrett and Søgaard, 2015b) and sentence compression (Klerke et al., 2016). Our work also tackles a traditional NLP task with free-form text, but differs from this line of research in that it addresses this task only in comprehension. Furthermore, while these studies use gaze recordings of native readers, our work focuses on non-native readers.

NLI in production NLI was first introduced in Koppel et al. (2005) and has been drawing considerable attention in NLP, including a recent shared-task challenge with 29 participating teams (Tetreault et al., 2013). NLI has also been driving much of the work on identification of native language related features in writing (Tsur and Rappoport, 2007; Jarvis and Crossley, 2012; Brooke and Hirst, 2012; Tetreault et al., 2012; Swanson and Charniak, 2013, 2014; Malmasi and Dras, 2014; Bykh and Meurers, 2016). Several studies have also linked usage patterns and grammatical errors in production to linguistic properties of the writer's native language (Nagata and Whittaker, 2013; Nagata, 2014; Berzak et al., 2014, 2015). Our work departs from NLI in writing and introduces NLI and related feature analysis in reading.

6 Conclusion and Outlook

We present a novel framework for studying cross-linguistic influence in multilingualism by measuring gaze fixations during reading of free-form En-

glish text. We demonstrate for the first time that this signal can be used to determine a reader's native language. The effectiveness of linguistically motivated criteria for fixation clustering and our subsequent analysis suggest that the ESL reading process is affected by linguistic factors. Specifically, we show that linguistic similarities between native languages are reflected in similarities in ESL reading. We also identify several key features that characterize reading in different native languages, and discuss their potential connection to structural and lexical properties of the native langauge. The presented results demonstrate that eyetracking data can be instrumental for developing predictive and explanatory models of second language reading.

While this work is focused on NLIR from fixations, our general framework can be used to address additional aspects of reading, such as analysis of saccades and gaze trajectories. In future work, we also plan to explore the role of native and second language writing system characteristics in second language reading. More broadly, our methodology introduces parallels with production studies in NLP, creating new opportunities for integration of data, methodologies and tasks between production and comprehension. Furthermore, it holds promise for formulating language learning theory that is supported by empirical findings in naturalistic setups across language processing domains.

Acknowledgements

We thank Amelia Smith, Emily Weng, Run Chen and Lila Jansen for contributions to stimuli preparation and data collection. We also thank Andrei Barbu, Guy Ben-Yosef, Yen-Ling Kuo, Roger Levy, Jonathan Malmaud, Karthik Narasimhan and the anonymous reviewers for valuable feedback on this work. This material is based upon work supported by the Center for Brains, Minds, and Machines (CBMM), funded by NSF STC award CCF-1231216.

References

Rosa Alonso Alonso. 2015. *Crosslinguistic Influence in Second Language Acquisition*, volume 95. Multilingual Matters.

Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. 2016. Weakly supervised part-of-speech tagging using eye-tracking data. In *ACL*. volume 2, pages 579–584.

Maria Barrett and Anders Søgaard. 2015a. Reading behavior predicts syntactic categories. In *CoNLL*. pages 345–349.

Maria Barrett and Anders Søgaard. 2015b. Using reading behavior to predict grammatical functions. In *Proceedings of the Sixth Workshop on Cognitive Aspects of Computational Language Learning*. pages 1–5.

Éva Berkes and Suzanne Flynn. 2012. Multilingualism: New perspectives on syntactic development. *The Handbook of Bilingualism and Multilingualism, Second Edition* pages 137–167.

Yevgeni Berzak, Roi Reichart, and Boris Katz. 2014. Reconstructing native language typology from foreign language usage. In *Eighteenth Conference on Computational Natural Language Learning (CoNLL)*.

Yevgeni Berzak, Roi Reichart, and Boris Katz. 2015. Contrastive analysis with predictive power: Typology driven estimation of grammatical error distributions in esl. In *Conference on Computational Natural Language Learning (CoNLL)*.

Julian Brooke and Graeme Hirst. 2012. Measuring interlanguage: Native language identification with l1-influence metrics. In *LREC*. pages 779–784.

Serhiy Bykh and Detmar Meurers. 2016. Advancing linguistic features and insights by label-informed feature grouping: An exploration in the context of native language identification. In *COLING*.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing* 16(5):1190–1208.

Chris Collins and Richard Kayne. 2009. *Syntactic Structures of the world's languages*. http://sswl.railsplayground.net.

Uschi Cop, Denis Drieghe, and Wouter Duyck. 2015. Eye movement patterns in natural reading: A comparison of monolingual and bilingual reading of a novel. *PLOS ONE* 10(8):1–38.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig. http://wals.info/.

Paola E Dussias. 2010. Uses of eye-tracking data in second language sentence processing research. *Annual Review of Applied Linguistics* 30:149–166.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2015. Glottolog 2.6. *Leipzig: Max Planck Institute for Evolutionary Anthropology*. http://glottolog.org.

Scott Jarvis and Scott A Crossley. 2012. *Approaching Language Transfer Through Text Classification: Explorations in the Detection-based Approach*, volume 64. Multilingual Matters.

Scott Jarvis and Aneta Pavlenko. 2008. *Crosslinguistic influence in language and cognition*. Routledge.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. *NAACL-HLT* .

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Determining an author's native language by mining a text for errors. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, pages 624–628.

Paul M. Lewis, Gary F. Simons, and Charles D. Fennig, editors. 2015. *Ethnologue: Languages of the World*. SIL International, Dallas, Texas. http://www.ethnologue.com.

Patrick Littel, David Mortensen, and Lori Levin, editors. 2016. *URIEL Typological Database*. Pittsburgh: Carnegie Mellon University. http://www.cs.cmu.edu/ dmortens/uriel.html.

Patrick Littell, David Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. *EACL 2017* page 8.

Shervin Malmasi and Mark Dras. 2014. Language transfer hypotheses with linear svm weights. In *EMNLP*. pages 1385–1390.

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

Gita Martohardjono and Suzanne Flynn. 1995. Language transfer: what do we really mean. In L. Eubank, L. Selinker, and M. Sharwood Smith, editors, *The current state of Interlanguage: studies in honor of William E. Rutherford*, John Benjamins: The Netherlands, pages 205–219.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL*. pages 92–97.

Ryo Nagata. 2014. Language family relationship preserved in non-native english. In *COLING*. pages 1940–1949.

Ryo Nagata and Edward W. D. Whittaker. 2013. Reconstructing an indo-european family tree from non-native english texts. In *ACL*. pages 1137–1147.

Terence Odlin. 1989. *Language transfer: Cross-linguistic influence in language learning*. Cambridge University Press.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC*.

Steven T Piantadosi, Harry Tily, and Edward Gibson. 2011. Word lengths are optimized for efficient communication. *Proceedings of the National Academy of Sciences* 108(9):3526–3529.

Leah Roberts and Anna Siyanova-Chanturia. 2013. Using eye-tracking to investigate topics in l2 acquisition and l2 processing. *Studies in Second Language Acquisition* 35(02):213–235.

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). *Technical Reports (CIS)* .

Ben Swanson and Eugene Charniak. 2013. Extracting the native language signal for second language acquisition. In *HLT-NAACL*. pages 85–94.

Ben Swanson and Eugene Charniak. 2014. Data driven language transfer hypotheses. *EACL* page 169.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Citeseer, pages 48–57.

Joel R Tetreault, Daniel Blanchard, Aoife Cahill, and Martin Chodorow. 2012. Native tongues, lost and found: Resources and empirical evaluations in native language identification. In *COLING*. pages 2585–2602.

Oren Tsur and Ari Rappoport. 2007. Using classifier features for studying the effect of native language on the choice of written second language words. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*. Association for Computational Linguistics, pages 9–16.

Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* 58(301):236–244.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *ACL*. pages 180–189.

A Supplemental Material

Eyetracking Setup We use a 44.5x30cm screen with 1024x768px resolution to present the reading materials, and a desktop mount Eyelink 1000 eyetracker (1000Hz) to record gaze. The screen, eyetracker camera and chinrest are horizontally aligned on a table surface. The screen center (x=512, y=384) is 79cm away from the center of

the forehead bar, and 13cm below it. The eye-tracker camera knob is 65cm away from forehead bar. Throughout the experiment participants hold a joystick with a button for indicating sentence completion, and two buttons for answering yes/no questions. We record gaze of the participant's dominant eye.

Text Parameters All the textual material in the experiment is presented using Times font, normal style, with font size 23. In our setup, this corresponds to 0.36 degrees (11.3px) average lower case letter width, and 0.49 degrees (15.7px) average upper case letter width. We chose a non-monospace font, as such fonts are generally more common in reading. They are also more compact compared to monospace fonts, allowing to substantially increase the upper limit for sentence length.

Calibration We use 3H line calibration with point repetition on the central horizontal line (y=384), using 16px outer circle, 6px inner circle, fixation points. At least three calibrations are performed during the experiment, one at the beginning of each experimental section. We also recalibrate upon failure to produce a 300ms fixation on any fixation trigger preceding a sentence or a question within 4 seconds after its appearance. The mean validation error for calibrations across subjects is 0.146 degrees (std 0.038).

MORSE: Semantic-ally Drive-n MORpheme SEgment-er

Tarek Sakakini
University of Illinois
Urbana, IL 61820
sakakini@illinois.edu

Suma Bhat
University of Illinois
Urbana, IL 61820
spbhat2@illinois.edu

Pramod Viswanath
University of Illinois
Urbana, IL 61820
pramodv@illinois.edu

Abstract

In this paper we present a novel framework for morpheme segmentation which uses the morpho-syntactic regularities preserved by word representations, in addition to orthographic features, to segment words into morphemes. This framework is the first to consider vocabulary-wide syntactico-semantic information for this task. We also analyze the deficiencies of available benchmarking datasets and introduce our own dataset that was created on the basis of compositionality. We validate our algorithm across different datasets and languages and present new state-of-the-art results.

1 Introduction

Morpheme segmentation is a core natural language processing (NLP) task used as an integral component in related-fields such as information retrieval (IR) (Zieman and Bleich, 1997; Kurimo et al., 2007), automatic speech recognition (ASR) (Bilmes and Kirchhoff, 2003; Kurimo et al., 2006), and machine translation (MT) (Lee, 2004; Virpioja et al., 2007). Most previous works have relied solely on orthographic features (Harris, 1970; Goldsmith, 2000; Creutz and Lagus, 2002, 2005, 2007), neglecting the underlying semantic information. This has led to an over-segmentation of words because a change of the surface form pattern is a necessary but insufficient indication of a morphological change. For example, the surface form of "freshman", hints that it should be segmented to "fresh-man", although "freshman" does not describe semantically the compositional meaning of "fresh" and "man".

To compensate for this lack of semantic knowledge, previous works (Schone and Jurafsky, 2000; Baroni et al., 2002; Narasimhan et al., 2015) have incorporated semantic knowledge *locally* by checking the semantic relatedness of possibly morphologically related pair of words. Narasimhan et al. (2015) check for semantic relatedness using cosine similarity in word representations (Mikolov et al., 2013a; Pennington et al., 2014). A limitation of such an approach is the inherent "sample noise" in specific word representations (exacerbated in the case of rare words). Moreover, limitation to local comparison enforces modeling morphological relations via semantic relatedness, although it has been shown that difference vectors model morphological relations more accurately (Mikolov et al., 2013b). To address this issue, we introduce a new framework (MORSE), the first to bring semantics into morpheme segmentation both on a local and a vocabulary-wide level. That is, when checking for the morphological relation between two words, we not only check for the semantic relatedness of the pair at hand (*local*), but also check if the difference vectors of pairs showing similar orthographic change are consistent (*vocabulary-wide*).

In summary, MORSE clusters pairs of words which only vary by an affix; for example, pairs such as ("quick", "quickly") and ("hopeful", "hopefully") get clustered together. To verify the cluster of a specific affix from a semantic corpus-wide standpoint, we check for the consistency of the difference vectors (Mikolov et al., 2013b). To evaluate it from an orthographic corpus-wide perspective, we check for the size of each cluster of an affix. To evaluate each pair in a cluster locally from a semantic standpoint, we check if a pair of words in a valid affix cluster are morphologically related by checking if its difference vector is consistent with other members in the cluster and if the words in the pair are semantically related (i.e. close in the vector space). The reason for local

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 552–561
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1051

evaluations is exemplified by ("on","only") which belongs to the cluster of a valid affix ("ly"), although they are not (obviously) morphologically related. We would expect such a pair to fail the last two local evaluation methods.

Our proposed segmentation algorithm is evaluated using benchmarking datasets from the Morpho Challenge (MC) for multiple languages and a newly introduced dataset for English which compensates for lack of discriminating capabilities in the MC dataset. Experiments reveal that our proposed framework not only outperforms the widely used approach, but also performs better than published state-of-the-art results.

The central contribution of this work is a novel framework that performs morpheme segmentation resulting in new state-of-the-art results. To the best of our knowledge this is the first unsupervised approach to consider the vocabulary-wide semantic knowledge of words and their affixes in addition to relying on their surface forms. Moreover we point out the deficiencies in the MC datasets with respect to the compositionality of morphemes and introduce our own dataset free of these deficiencies.

2 Related Work

Extensive work has been done in morphology learning, with tasks such as morphological analysis (Baayen et al., 1993), morphological reinflection (Cotterell et al., 2016), and morpheme segmentation. Given the less complex nature of morpheme segmentation in comparison to the other tasks, most systems developed for morpheme segmentation have been unsupervised or minimally supervised (mostly for parameter tuning).

Unsupervised morpheme segmentation traces back to (Harris, 1970), which falls under the framework of Letter Successor Variety (LSV) which builds on the hypothesis that predictability of successor letters is high within morphemes and low otherwise. The most dominant pieces of work on unsupervised morpheme segmentation, Morfessor (Creutz and Lagus, 2002, 2005, 2007) and Linguistica (Goldsmith, 2000) adopt the Minimum Description Length (MDL) principle (Rissanen, 1998): they aim to minimize describing the lexicon of morphs as well as minimizing the description of an input corpus. Morfessor has a widely used API and has inspired a large body of following work (Kohonen et al., 2010; Grönroos et al., 2014).

The unsupervised original implementation was later adapted (Kohonen et al., 2010; Grönroos et al., 2014) to allow for minimal supervision. Another work on minimally supervised morpheme segmentation is (Sirts and Goldwater, 2013) which relies on Adaptor Grammars (AGs) (Johnson et al., 2006). AGs learn latent tree structures over an input corpus using a nonparametric Bayesian model (Sirts and Goldwater, 2013).

(Lafferty et al., 2001) use Conditional Random Fields (CRF) for morpheme segmentation. In this supervised method, the morpheme segmentation task is modeled as a sequence-to-sequence learning problem, whereby the sequence of labels defines the boundaries of morphemes (Ruokolainen et al., 2013, 2014). In contrast to the previously mentioned generative approaches of MDL and AG, this method takes a discriminative approach and allows for the inclusion of a larger set of features. In this approach, CRF learns a conditional probability of a segmentation given a word (Ruokolainen et al., 2013, 2014).

All these morpheme segmenters rely solely on orthographic features of morphemes. Semantics were initially introduced to morpheme segmenters by (Schone and Jurafsky, 2000), using LSA to generate word representations and then evaluate if two words are morphologically related based on semantic relatedness, as well as deterministic orthographic methods. Similarly, (Baroni et al., 2002) use edit distance and mutual information as metrics for semantic and orthographic validity of a morphological relation between two words. Recent work in (Narasimhan et al., 2015), inspired by the log-linear model in (Poon et al., 2009) incorporates semantic relatedness into the model via word representations. Other systems such as (Üstün and Can, 2016) rely solely on evaluating two words from a semantic standpoint by the use of a two-layer neural network.

MORSE introduces semantic information into its morpheme segmenters via distributed word representations while also relying on orthographic features. Inspired by the work of (Soricut and Och, 2015), instead of merely evaluating semantic relatedness, we are the first to evaluate the morphological relationship via the difference vector of morphologically related words. Comparing the difference vectors of multiple pairs across the corpus following the same morphological relation, gives

MORSE a vocabulary-wide evaluation of morphological relations learned.

3 System

The key limitation of previous frameworks that rely solely on orthographic features is the resulting over-segmentation. As an example, MDL-based frameworks segment "sing" to "s-ing" due to the high frequency of the morphemes: "s" and "ing". Our framework combines semantic relatedness with orthographic relatedness to eliminate such error. For the example mentioned, MORSE validates morphemes such as "s" and "ing" from an orthographic perspective, yet invalidates the relation between "s" and "sing" from a local and vocabulary-wide semantic perspective. Hence, MORSE will segment "jumping" as "jump-ing", and perform no segmentations on "sing".

To bring in semantic understanding into MORSE, we rely on word representations (Mikolov et al., 2013a; Pennington et al., 2014). These word representations capture the semantics of the vocabulary through statistics over the context in which they appear. Moreover, morpho-syntactic regularities have been shown over these word representations, whereby pairs of words sharing the same relationship exhibit equivalent difference vectors (Mikolov et al., 2013b). For example, it is expected in the vector space of word representations that $\vec{w}_{\text{jumping}} - \vec{w}_{\text{jump}} \approx \vec{w}_{\text{playing}} - \vec{w}_{\text{play}}$, but $\vec{w}_{\text{sing}} - \vec{w}_{\text{s}} \not\approx \vec{w}_{\text{playing}} - \vec{w}_{\text{play}}$.

As a high level description, we first learn all possible affix transformations (morphological rules) in the language from pairs of words from an orthographic standpoint. For example, the pair ("jump", "jumping") corresponds to the valid affix transformation $\phi \xrightarrow{\text{suffix}}$ "ing" (where ϕ represents the empty string), and the pair ("slow", "slogan") corresponds to the invalid rule "w" $\xrightarrow{\text{suffix}}$ "gan". Then we invalidate the rules, such as "w" $\xrightarrow{\text{suffix}}$ "gan", that do not conform to the linear relation in the vector space. We also invalidate pairs of words which, due to randomness, are orthographically related via a valid rule although they are not morphologically related, such as ("on", "only").

Now we formalize the objects we learn in MORSE and the scores (orthographic and semantic) used for validation. This constitutes the training stage. Finally, we formalize the inference stage, where we use these objects and scores to perform morpheme segmentation.

3.1 Training Stage

Objects:

- Rule set **R** made of all possible affix transformations in a language. **R** is populated via the following definition: $\mathbf{R}_{\text{suffix}} = \{\text{aff}_1 \xrightarrow{\text{suffix}} \text{aff}_2 \colon \exists\ (w_1, w_2) \in \text{V}^2, \text{stem}(w_1) = \text{stem}(w_2), w_1 = \text{stem}(w_1) + \text{aff}_1, w_2 = \text{stem}(w_2) + \text{aff}_2\}$, $\mathbf{R}_{\text{prefix}}$ is defined similarly for prefixes, and $\mathbf{R} = \mathbf{R}_{\text{suffix}} \cup \mathbf{R}_{\text{prefix}}$. An example **R** would be equal to $\{\phi \xrightarrow{\text{suffix}}$ "ly", $\phi \xrightarrow{\text{prefix}}$ "un", "ing" $\xrightarrow{\text{suffix}}$ "ed",$\dots\}$.

- Support set $\mathbf{SS}_r$ for a rule $r \in \mathbf{R}$ consists of all pairs of words related via r on a surface level. $\mathbf{SS}_r$ is populated via the following definition: $\mathbf{SS}_r = \{(w_1, w_2)\colon w_1, w_2 \in \text{V}, w_1 \xrightarrow{r} w_2\}$. An example support set of the rule "ing" $\xrightarrow{\text{suffix}}$ "ed" would be {("playing", "played"), ("crafting", "crafted"),$\dots$}.

Scores:

- $\mathbf{score}_{\text{r\_orth}}(r)$ is a vocabulary-wide orthographic confidence score for rule $r \in$ R. It reflects the validity of an affix transformation in a language from an orthographic perspective. This score is evaluated as $\mathbf{score}_{\text{r\_orth}}(r) = |\text{SS}_r|$.
- $\mathbf{score}_{\text{r\_sem}}(r)$ is a vocabulary-wide semantic confidence score for rule $r \in$ R. It reflects the validity of an affix transformation in a language from a semantic perspective. This score is evaluated as: $\mathbf{score}_{\text{r\_sem}}(r) = |\text{cluster}_{\text{r}}|/|\text{SS}_r|^2$ where $\text{cluster}_r = \{((w_1, w_2), (w_3, w_4))\colon (w_1, w_2), (w_3, w_4) \in \text{SS}_r, \vec{w}_1 - \vec{w}_2 \approx \vec{w}_3 - \vec{w}_4\}$. We consider $\vec{w}_1 - \vec{w}_2 \approx \vec{w}_3 - \vec{w}_4$ if $\cos(\vec{w}_4, \vec{w}_2 - \vec{w}_1 + \vec{w}_3) > 0.1$.
- $\mathbf{score}_{\text{w\_sem}}((w_1, w_2) \in \text{SS}_r)$ is a vocabulary-wide semantic confidence score for a pair of words (w_1, w_2). The pair of words is related via r on an orthographic level, but the score reflects the validity of the morphological relation via r on a semantic level. This score is evaluated as: $\mathbf{score}_{\text{w\_sem}}((w_1, w_2) \in \text{SS}_r) = |\{(w_3, w_4)\colon (w_3, w_4) \in \text{SS}_r, \vec{w}_1 - \vec{w}_2 \approx \vec{w}_3 - \vec{w}_4\}|/|\text{SS}_r|$. In other words, it is the fraction of pairs of words in the support set that exhibit a similar linear relation as (w_1, w_2) in the vector space.

- **score**$_{\text{loc\_sem}}((w_1, w_2) \in \text{SS}_r)$ is a local semantic confidence score for a pair of words (w_1, w_2). The pair of words is related via r on an orthographic level, but the score reflects the semantic relatedness between the pair. The score is evaluated as: **score**$_{\text{loc\_sem}}((w_1, w_2) \in \text{SS}_r) = \cos(\vec{w_1}, \vec{w_2})$.

3.2 Inference Stage

In this stage we perform morpheme segmentation using the knowledge gained from the first stage. We begin with some notation: let $\text{R}_{\text{add}} = \{r : r \in \text{R}, r = \text{aff}_1 \xrightarrow{r} \text{aff}_2, \text{aff}_1 = \phi, \text{aff}_2 \neq \phi\}$, $\text{R}_{\text{rep}} = \{r : r \in \text{R}, r = \text{aff}_1 \xrightarrow{r} \text{aff}_2, \text{aff}_1 \neq \phi, \text{aff}_2 \neq \phi\}$. In other words, we divide the rules to those where an affix is added (R_{add}) and to those where an affix is replaced (R_{rep}).

Given a word w to segment, we search for r^*, the solution to the following optimization problem[1]. The search space is limited to the rules that include w in their support set, a fairly small search space and the corresponding computation readily tractable:

$$\max_{\mathbf{r}} \quad \sum_{t_1} \text{score}_{t_1}((w', w) \in \text{SS}_{\mathbf{r}}) + \sum_{t_2} \text{score}_{t_2}(\mathbf{r})$$

$$s.t. \quad \mathbf{r} \in \text{R}_{\text{add}}$$
$$\text{score}_{\text{r\_sem}}(\mathbf{r}) > \text{t}_{\text{r\_sem}}$$
$$\text{score}_{\text{r\_orth}}(\mathbf{r}) > \text{t}_{\text{r\_orth}}$$
$$\text{score}_{\text{w\_sem}}((w', w) \in \text{SS}_{\mathbf{r}}) > \text{t}_{\text{w\_sem}}$$
$$\text{score}_{\text{loc\_sem}}((w', w) \in \text{SS}_{\mathbf{r}}) > \text{t}_{\text{loc\_sem}}$$

Where t_1 = {w_sem, loc_sem}, t_2 = {r_sem, r_orth}, and $\text{t}_{\text{r\_sem}}$, $\text{t}_{\text{r\_orth}}$, $\text{t}_{\text{w\_sem}}$, $\text{t}_{\text{loc\_sem}}$ are hyperparameters of the system. Now given $\text{r}^* = \phi \xrightarrow{\text{suffix}}$ suf, w' is defined as $w' \xrightarrow{\text{r}^*} w$. Thus the algorithm segments $w \rightarrow w'$-suf. We treat prefixes similarly. Next, the algorithm iterates over w'. Figure 1 shows the segmentation process of the word "unhealthy" based on the sequentially retrieved r^*.

The reason we restrict our rule set to R_{add} in the optimization problem is to avoid rules such as "er" $\xrightarrow{\text{suffix}}$ "ing" like in ("player", "playing") leading to false segmentations such as "playing" → "playering". Yet we cannot completely restrict our search to R_{add} due to rules such as "y" → "ies" in words like ("sky", "skies"). To be able to segment words such as "skies", we'd have to consider rules in R_{rep} but only after searching in R_{add}. Thus if the first optimization problem was unfeasible, we repeat it while replacing R_{add} with R_{rep}. The program terminates when both optimization problems are infeasible.

[1] r and w uniquely identify w', and thus the search space is defined only over r.

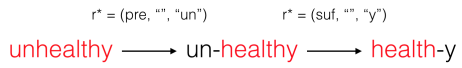


Figure 1: Illustration of the iterative process of segmentation in MORSE

4 Experiments

We conduct a variety of experiments to assess the performance of MORSE, and compare it with prior works. First, the performance is assessed intrinsically on the task of morpheme segmentation and against the most widely used morpheme segmenter: Morfessor 2.0. We evaluate the performance across three languages of varying morphology levels: English, Turkish, Finnish, with Finnish being the richest in morphology and English being the poorest. Second, we show the inadequacies of benchmarking gold datasets for this task and describe a new dataset that we create to address the inadequacy. Third, in order to highlight the effect of including semantic information, we compare MORSE against Morfessor on a set of words which should not be segmented from a semantic perspective although orthographically they seem to be segmentable (such as "freshman").

In all of our experiments (unless specified otherwise), we report precision and recall (and corresponding F1 scores) with locations of morpheme boundaries being considered positives and the rest of the locations considered negatives. It should be noted that we disregard starting and ending positions of words, since they form trivial boundaries (Virpioja et al., 2011).

4.1 Setup

Both systems, Morfessor and MORSE, were trained on the same monolingual corpus: Wikipedia[2] (as of September 20, 2016) to control for affecting factors within the experiment. For each language considered, the respective Wikipedia dump was preprocessed using an available code[3]. We use Word2Vec (Mikolov

[2] https://dumps.wikimedia.org

[3] https://github.com/bwbaugh/wikipedia-extractor

Dataset	En	Fi	Tr
Tuning Data	1000	1000	971
Test Data	686	760	809

Table 1: Morpho Challenge 2010 Dataset Sizes.

et al., 2013a) to train word representations of 300 dimensions and based on a context window of size 5. Also, for computational efficiency, MORSE was limited to a vocabulary of size 1M, a restriction not enforced on Morfessor.

MORSE's hyperparameters are tuned based on a tuning set of gold morpheme segmentations. We have publicly released the source code of a pre-trained MORSE[4] as described in this paper.

4.2 Morpho Challenge Dataset

As our first intrinsic experiment, we consider the Morpho Challenge (MC) gold segmentations available online[5]. For every language, two datasets are supplied: training and development. For the purpose of our experiments, all systems use the development dataset as a test dataset, and the training dataset is used for tuning MORSE's hyperparameters. MC dataset sizes are reported in Table 1.

4.3 Semantically Driven Dataset

There are a variety of weaknesses in the MC dataset, specifically related to whether the segmentation is semantically appropriate or not. We introduce a new semantically driven dataset (SD17) for morpheme segmentation along with the methodology used for creation; this new dataset is publicly available in the canonical[6] and non-canonical[7] versions (Cotterell and Vieira, 2016).

Non-compositional segmentation: One of the key requirements of morpheme segmentation is the compositionality of the meaning of the word from the meaning of its morphemes. This requirement is violated on multiple occasions in the MC dataset. One example from Table 2 is segmenting the word "business" into "busi-ness", which falsely assumes that "business" means the act of being busy. Such a segmentation might be consistent with the historic origin of the word, but with radical semantic changes over time, the segmentation no longer semantically represents the compositionality of the words' components (Wijaya and Yeniterzi, 2011). Not only does such a weakness contribute to false segmentations, but it also favors segmentation methods following the MDL principle.

Word	Gold Segmentation
freshman	fresh man
airline	air line
business'	busi ness '
ahead	a head
adultery	adult ery

Table 2: Examples of gold morpheme segmentations from the Morpho Challenge 2010 dataset deemed invalid from a compositionality viewpoint.

Trivial instances: The second weakness in the MC dataset is due to abundance of trivial instances. These instances lack discriminating capability since all methods can easily predict them (Baker, 2001). These instances are comprised of genetive cases (such as teacher's) as well as hyphenated words (such as turning-point). For genetive cases, segmenting at the apostrophe leads to perfect precision and recall, and thus such instances are deemed trivial. In the case of hyphenated words, segmenting at the hyphen is a correct segmentation with a very high probability. In the MC tuning dataset, in 43 times out of 46, the hyphen was a correct indication of segmentation.

Other issues exist in the Morpho Challenge dataset although less abundantly. There are instances of **wrong segmentations** possibly due to human error. One example of such instance is "turning-point" segmented to "turning - point" instead of "turn ing - point". Another issue, which is hard to avoid, is **ambiguity** of segmentation boundaries. Take for example the word "strafed", the segmentations "straf-ed" and "strafe-d" are equally justified. In such situations, the MC dataset favors complete affixes rather than complete lemmas. This also favors MDL-based segmenters. We note that the MC dataset also provides segmentations in a canonical version such as "strafe-ed", yet for the sake of a fair comparison with Morfessor and all previously evaluated systems on the MC dataset, we consider only the former version of segmentations.

[4] https://goo.gl/w4r7vP
[5] http://research.ics.aalto.fi/events/morphochallenge2010
[6] https://goo.gl/MgKfG1
[7] https://goo.gl/0vTXVt

	English			Turkish			Finnish		
	P	R	F1	P	R	F1	P	R	F1
Morfessor	74.46	56.66	64.35	40.81	25.00	31.01	**43.09**	**28.16**	**34.06**
MORSE	**81.98**	**61.57**	**70.32**	**49.90**	**30.78**	**38.07**	36.26	9.44	14.98

Table 3: Performance of MORSE on the MC dataset across three languages: English, Turkish, Finnish.

Due to these reasons, we create a new dataset SD17 for English gold morpheme segmentations with compositionality guiding the annotations. We select 2000 words randomly from the 10K most frequent words in the English Wikipedia dump and have them annotated by two proficient English speakers. The segmentation criterion was to segment the word to the largest extent possible while preserving its compositionality from the segments. The inter-annotator agreement reached 91% on a word level. Based on post annotation discussions, annotators agreed on 99% of the words, and words not agreed on were eliminated along with words containing non-alpha characters to avoid trivial instances.

SD17 is used to evaluate the performance of both Morfessor and MORSE. We claim that the performance on SD17 is a better indication of the performance of a morpheme segmenter. By the use of SD17 we expect to gain insights on the extent to which morpheme segmentation is a function of semantics in addition to orthography.

4.4 Handling Compositionality

We have hypothesized that following the MDL principle (such as Morfessor) leads to over-segmentation. This over-segmentation happens specifically when the meaning of the word does not follow from the meaning of its morphemes. Examples include words such as "red head", "duck face", "how ever", "s ing". A subset of these words are defined by linguists as exocentric compounds (Bauer, 2008). MORSE does not suffer from this issue owing to its use of a semantic model.

We use a collection of 100 English words which appear to be segmentable but actually are not (example: "however"). Such a collection will highlight a system's capability of distinguishing frequent letter sequences from the semantic contribution of this letter sequence in a word. We make this collection publicly available[8].

[8] https://goo.gl/EFbacj

	En	Tr	Fi
Candidate Rules	27.5M	14.9M	10.8M
Candidate Rel. Pairs	53.3M	25.1M	18.6M

Table 4: Number of candidate rules and candidate related word pairs detected per language.

5 Results

We compare MORSE with Morfessor, and place the performance alongside the state-of-the-art published results.

5.1 Morpho Challenge Dataset

As demonstrated in Table 3, MORSE performs better than Mofessor on English and Turkish, and worse on Finnish. Considering English first, using MORSE instead of Morfessor, resulted in a 6% absolute increase in F1 scores. This supports our claim for the need of semantic cues in morpheme segmentation, and also validates the method used in this paper. Since English is a less systematic language in terms of the orthographic structure of words, semantic cues are of greater need, and hence a system which relies on semantic cues is expected to perform better; indeed this is the case. Similarly, MORSE performs better on Turkish with a 7% absolute margin in terms of F1 score. On the other hand, Morfessor surpasses MORSE in performance on Finnish by a large margin as well, especially in terms of recall.

5.1.1 Discussion

We hypothesize that the richness of morphology in Finnish led to suboptimal performance of MORSE. This is because richness in morphology leads to word level sparsity which directly leads to: (1) Degradation of quality of word representations (2) Increased vocabulary size exacerbating the issue of limited vocabulary (recall MORSE was limited to a vocabulary of 1M). In a language with productive morphology, limiting its vocabulary results in a lower chance of finding morphologically related word pairs. This negatively im-

pacts the training stage of MORSE which relies on the availability of such pairs. In order to detect the suffix "ly" from the word "cheerfully" MORSE needs to come across "cheerful" as well. Coming across "cheerful" is now a lower probability event due to high sparsity. This is not as much of an issue for Morfessor under the MDL principle, since it might detect "ly" just by coming across multiple words ending with "ly" even without encountering the base forms of those words. We show how the detection of rules is affected by considering the number of candidate rules detected as well as the number of candidate morphologically related word pairs detected. As shown in Table 4, the number of detected candidate rules and candidate related words decreases with the increase in morphology in a language. This confirms our hypothesis; we note that this issue can be directly attributed to the limited vocabulary size in MORSE. With the increase in processing power, and thus larger vocabulary coverage, MORSE is expected to perform better.

5.2 Semantically Driven Dataset

The performance of MORSE and Morfessor on SD17 is shown in Table 5. The use of MC data (which does not adhere to the compositionality principle) to tune MORSE to be evaluated on SD17 (which does adhere to the compositionality principle) is not optimal. Thus, we evaluate MORSE on SD17 using 5-fold cross validation, where 80% of the dataset is used to tune and 20% is used to evaluate. Precision, Recall, and F1 scores are averaged and reported in Table 5 using the label MORSE-CV.

Based on the results in Table 5, we make the following observations. Comparing MORSE-CV to MORSE reflects the fundamental difference between SD17 and MC datasets. Knowing the basis of construction of SD17 and the fundamental weaknesses in MC datasets, we attribute the performance increase to the lack of compositionality in MC dataset. Comparing MORSE-CV to Morfessor, we observe a significant jump in performance (an increase of 24%). In comparison, the increase on the MC dataset (6%) shows that the Morpho Challenge dataset underestimates the performance gap between Morfessor and MORSE due its inherent weaknesses.

Since MORSE is equipped with the capability to retrieve full morphemes even when not present in full orthographically, a capability that Morfessor lacks, we evaluated both systems on the canonical version of SD17. The results are reported in Table 6. We notice that evaluating on the canonical form of SD17 gives a further edge for MORSE over Morfessor. For evaluation on the canonical version of SD17, we switch to morpheme-level evaluation instead of boundary-level as a more suitable method for Morfessor. Morpheme-level evaluation is distinguished from boundary-level evaluation in that we evaluate the detection of morphemes instead of the boundary locations in the segmented word.

	P	R	F1
Morfessor	65.95	51.13	57.60
MORSE	75.35	**83.60**	79.26
MORSE-CV	**84.6**	78.36	**81.29**

Table 5: Performance of MORSE against Morfessor on the non-canonical version of SD17

	P	R	F1
Morfessor	65.61	50.87	57.31
MORSE	79.70	82.37	81.01
MORSE-CV	**85.08**	**82.90**	**83.96**

Table 6: Performance of MORSE against Morfessor on the canonical version of SD17

We next compare MORSE against published state-of-the-art results[9]. As one can see in Table 7 MORSE significantly performs better than published state-of-the-art results, most notably (Narasimhan et al., 2015) referred to as LLSM in the Table. Comparison is also made against the top results in the latest Morpho Challenge: Morfessor S+W and Morfessor S+W+L (Kohonen et al., 2010), and Base Inference (Lignos, 2010).

	P	R	F1
MORSE	**84.6**	**78.36**	**81.29**
LLSM	80.70	72.20	76.2
Morfessor S+W	65.62	69.28	67.40
Morfessor S+W+L	67.87	66.43	67.14
Base Inference	80.77	53.76	64.55

Table 7: Performance of MORSE against published state-of-the-art results

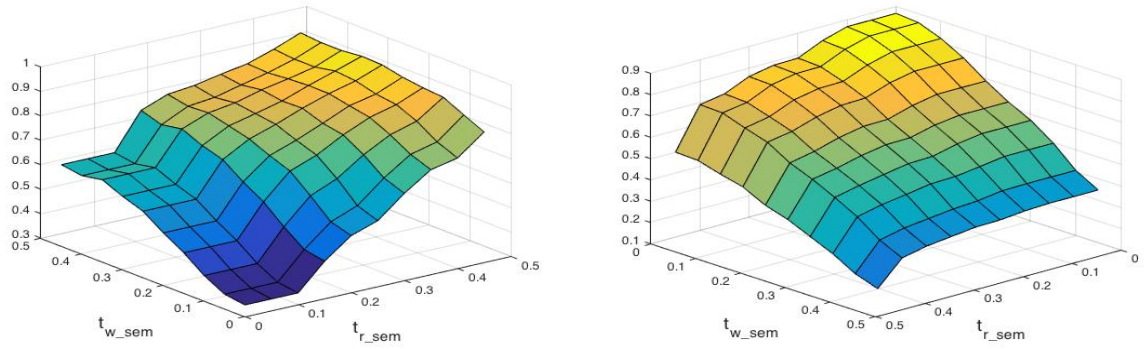


Figure 2: Precision (left) and Recall (right) of MORSE as a function of the hyperparameters: t_{r_sem}, t_{w_sem}

5.3 Handling Compositionality

We compare the performance of MORSE and Morfessor on a set of words made up of morphemes which don't compose the meaning of the word. Since all the boundaries in this dataset are negative, to evaluate both systems (with MORSE tuned on SD17), we only report the number of segments generated. The more segments a system generates, the worse is its performance.

We find that MORSE generates **7** false morphemes whereas Morfessor generates **43** false morphemes. This shows MORSE's robustness to such examples through its semantic knowledge and validates our claim that Morfessor oversegments on such examples.

6 Discussion

One of the benefits of MORSE against other frameworks such as MDL is its ability to identify the lemma within the segmentation. The lemma would be the last non-segmented word in the iterative process of segmentation. Hence, an advantage of our framework is its easy adaptability into a lemmatizer and even a stemmer.

Another key aspect which is not present in some of the competitive systems is the need for a small tuning dataset. This is a point in favor of completely unsupervised systems such as Morfessor. On the other hand, these hyperparameters could allow for flexibility. Figure 2 shows how precision and recall changes as a function of the hyperparameter selection[10]. As one would expect, increasing the hyperparameters, in general, leads to a stricter search space and thus increases precision and decreases recall. Putting these results in perspective, the user of MORSE is given the capability of controlling for precision and recall based on the needs of the downstream task.

Moreover, to check for the level of dependency of MORSE on a set of gold morpheme segmentations for tuning, we check for the variation in performance with respect to size of tuning data. For the purpose of this experiment we take an 80-20 split of SD17 and vary the size of the tuning set. We notice that the performance (81.90% F1) reaches a steady state at 20% ($\approx$ 300 gold segmentations) of the tuning data. This reflects the minimal dependency on a tuning dataset.

Regarding the training stage, homomorphs are treated as one rule and allomorphs are treated as separate rules. For example, ("tall", "taller") and ("fast", "faster") are wrongly considered to have the same morphological relation, besides ("cat", "cats") and ("butterfly", "butterflies") are wrongly considered to have different morphological relations. The separate clustering of the different forms of a homomorph leads to the underestimation of the respective orthographic scores. Moreover, the clustering of allomorphs together would lead to the underestimation of the semantic score of the rule as well as the underestimation of the vocabulary-wide semantic score of word pairs in the support set of this rule. This does not significantly affect the performance of MORSE, since the tuned thresholds are able to distinguish between the low scores of an invalid rule and the mediocre underestimated scores of allomorphs and homomorphs.

As for the inference stage of MORSE, the greedy inference approach limits its performance. In other words, a wrong segmentation at the be-

[9]The five published state-of-the-art results are on different datasets

[10]Only a subset of the hyperparameters is used for display purposes

ginning will propagate and result in consequent wrong segmentations. Also, MORSE's limitation to concatenative morphology decreases its efficacy on languages that include non-concatenative morphology. This opens the stage for further research on a more optimal inference stage and a more global modeling of orthographic morphological transformations.

7 Conclusions and Future Work

In this paper, we have presented MORSE, a first morpheme segmenter to consider semantic structure at this scale (local and vocabulary-wide). We show its superiority over state-of-the-art algorithms using intrinsic evaluation on a variety of languages. We also pinpointed the weaknesses in current benchmarking datasets, and presented a new dataset free of these weaknesses. With a relative increase in performance reaching 24% absolute increase over Morfessor, this work proves the significance of semantic cues as well as validates a new state-of-the-art morpheme segmenter. For future work, we plan to address the limitations of MORSE: minimal supervision, greedy inference, and concatenative orthographic model. Moreover, we plan to computationally optimize the training stage for the sake of wider adoption by the community.

Acknowledgements

This work is supported in part by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM Cognitive Horizons Network.

References

RH Baayen, R Piepenbrock, and H Van Rijn. 1993. The CELEX lexical database [cd-rom] Philadelphia: University of Pennsylvania. *Linguistic Data Consortium* .

Frank B Baker. 2001. *The basics of item response theory*. ERIC.

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning-Volume 6*. Association for Computational Linguistics, pages 48–57.

Laurie Bauer. 2008. Exocentric compounds. *Morphology* 18(1):51–74.

Jeff A Bilmes and Katrin Kirchhoff. 2003. Factored language models and generalized parallel backoff. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*. Association for Computational Linguistics, pages 4–6.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 2016 Meeting of SIGMORPHON*. Association for Computational Linguistics, Berlin, Germany.

Ryan Cotterell and Tim Vieira. 2016. A joint model of orthography and morphological segmentation. In *Proceedings of NAACL-HLT*. pages 664–669.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning-Volume 6*. Association for Computational Linguistics, pages 21–30.

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)* 4(1):3.

John Goldsmith. 2000. Linguistica: An automatic morphological analyzer. In *Proceedings of 36th meeting of the Chicago Linguistic Society*.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor Flatcat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *COLING*. pages 1177–1185.

Zellig S Harris. 1970. From phoneme to morpheme. In *Papers in Structural and Transformational Linguistics*, Springer, pages 32–67.

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in neural information processing systems*. pages 641–648.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*. Association for Computational Linguistics, pages 78–86.

Mikko Kurimo, Mathias Creutz, and Ville T Turunen. 2007. Unsupervised morpheme analysis evaluation by IR experiments-Morpho Challenge 2007. In *CLEF (Working Notes)*.

Mikko Kurimo, Mathias Creutz, Matti Varjokallio, Ebru Arisoy, and Murat Saraçlar. 2006. Unsupervised segmentation of words into morphemes–challenge 2005: An introduction and evaluation report. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth International Conference on Machine Learning, ICML*. volume 1, pages 282–289.

Young-Suk Lee. 2004. Morphological analysis for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*. Association for Computational Linguistics, pages 57–60.

Constantine Lignos. 2010. Learning from unseen data. In *Proceedings of the Morpho Challenge 2010 Workshop*. pages 35–38.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *HLT-NAACL*. volume 13, pages 746–751.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics* 3.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 209–217.

Jorma Rissanen. 1998. *Stochastic complexity in statistical inquiry*, volume 15. World scientific.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *CoNLL*. pages 29–37.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *EACL*. pages 84–89.

Patrick Schone and Daniel Jurafsky. 2000. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational Natural Language Learning-Volume 7*. Association for Computational Linguistics, pages 67–72.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *Transactions of the Association for Computational Linguistics* 1:255–266.

Radu Soricut and Franz Josef Och. 2015. Unsupervised morphology induction using word embeddings. In *HLT-NAACL*. pages 1627–1637.

Ahmet Üstün and Burcu Can. 2016. Unsupervised morphological segmentation using neural word embeddings. In *International Conference on Statistical Language and Speech Processing*. Springer, pages 43–53.

Sami Virpioja, Ville T Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *TAL* 52(2):45–90.

Sami Virpioja, Jaakko J Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. *Machine Translation Summit XI* 2007:491–498.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proceedings of the 2011 International Workshop on DETecting and Exploiting Cultural diversiTy on the Social Web*. ACM, New York, NY, USA, DETECT '11, pages 35–40. https://doi.org/10.1145/2064448.2064475.

Yuri L Zieman and Howard L Bleich. 1997. Conceptual mapping of user's queries to medical subject headings. In *Proceedings of the AMIA Annual Fall Symposium*. American Medical Informatics Association, page 519.

Deep Pyramid Convolutional Neural Networks for Text Categorization

Rie Johnson
RJ Research Consulting
Tarrytown, NY, USA
riejohnson@gmail.com

Tong Zhang
Tencent AI Lab
Shenzhen, China
bradymzhang@tencent.com

Abstract

This paper proposes a *low-complexity* word-level deep convolutional neural network (CNN) architecture for text categorization that can efficiently represent *long-range* associations in text. In the literature, several deep and complex neural networks have been proposed for this task, assuming availability of relatively large amounts of training data. However, the associated computational complexity increases as the networks go deeper, which poses serious challenges in practical applications. Moreover, it was shown recently that shallow word-level CNNs are more accurate and much faster than the state-of-the-art very deep nets such as character-level CNNs even in the setting of large training data. Motivated by these findings, we carefully studied deepening of word-level CNNs to capture global representations of text, and found a simple network architecture with which the best accuracy can be obtained by increasing the network depth without increasing computational cost by much. We call it *deep pyramid CNN*. The proposed model with 15 weight layers outperforms the previous best models on six benchmark datasets for sentiment classification and topic categorization.

1 Introduction

Text categorization is an important task whose applications include spam detection, sentiment classification, and topic classification. In recent years, neural networks that can make use of word order have been shown to be effective for text categorization. While simple and shallow *convolutional neural networks (CNNs)* (Kim, 2014; Johnson and Zhang, 2015a) were proposed for this task earlier, more recently, deep and more complex neural networks have also been studied, assuming availability of relatively large amounts of training data (e.g., one million documents). Examples are deep character-level CNNs (Zhang et al., 2015; Conneau et al., 2016), a complex combination of CNNs and *recurrent neural networks (RNNs)* (Tang et al., 2015), and RNNs in a word-sentence hierarchy (Yang et al., 2016).

A CNN is a feedforward network with convolution layers interleaved with pooling layers. Essentially, a convolution layer converts to a vector every small patch of data (either the original data such as text or image or the output of the previous layer) at every location (e.g., 3-word windows around every word), which can be processed in parallel. By contrast, an RNN has connections that form a cycle. In its typical application to text, a recurrent unit takes words one by one as well as its own output on the previous word, which is parallel-processing unfriendly. While both CNNs and RNNs can take advantage of word order, the simple nature and parallel-processing friendliness of CNNs make them attractive particularly when large training data causes computational challenges.

There have been several recent studies of CNN for text categorization in the large training data setting. For example, in (Conneau et al., 2016), very deep 32-layer character-level CNNs were shown to outperform deep 9-layer character-level CNNs of (Zhang et al., 2015). However, in (Johnson and Zhang, 2016), very shallow 1-layer word-level CNNs were shown to be more accurate and much faster than the very deep character-level CNNs of (Conneau et al., 2016). Although character-level approaches have merit in not having to deal with millions of distinct words, shallow word-level CNNs turned out to be superior even

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 562–570
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1052

when used with only a manageable number (30K) of the most frequent words. This demonstrates the basic fact – knowledge of *word* leads to a powerful representation. These results motivate us to pursue an effective and efficient design of *deep word-level* CNNs for text categorization. Note, however, that it is not as simple as merely replacing characters with words in character-level CNNs; doing so rather degraded accuracy in (Zhang et al., 2015).

We carefully studied deepening of word-level CNNs in the large-data setting and found a deep but low-complexity network architecture with which the best accuracy can be obtained by increasing the depth but not the order of computation time – the total computation time is bounded by a constant. We call it *deep pyramid CNN (DPCNN)*, as the computation time per layer decreases exponentially in a '*pyramid* shape'. After converting discrete text to continuous representation, the DPCNN architecture simply alternates a convolution block and a downsampling layer over and over[1], leading to a deep network in which internal data size (as well as per-layer computation) shrinks in a *pyramid* shape. The network depth can be treated as a meta-parameter. The computational complexity of this network is bounded to be no more than twice that of one convolution block. At the same time, as described later, the 'pyramid' enables efficient discovery of long-range associations in the text (and so more global information), as the network is deepened. This is why DPCNN can achieve better accuracy than the shallow CNN mentioned above (hereafter *ShallowCNN*), which can use only short-range associations. Moreover, DPCNN can be regarded as a deep extension of ShallowCNN, which we proposed in (Johnson and Zhang, 2015b) and later tested with large datasets in (Johnson and Zhang, 2016).

We show that DPCNN with 15 weight layers outperforms the previous best models on six benchmark datasets for sentiment classification and topic classification.

2 Word-level deep pyramid CNN (DPCNN) for text categorization

Overview of DPCNN: DPCNN is illustrated in Figure 1a. The first layer performs *text region embedding*, which generalizes commonly used *word embedding* to the embedding of text regions covering one or more words. It is followed by stacking of *convolution blocks* (two convolution layers and a shortcut) interleaved with pooling layers with stride 2 for downsampling. The final pooling layer aggregates internal data for each document into one vector. We use max pooling for all pooling layers. The key features of DPCNN are as follows.

- Downsampling *without* increasing the *number of feature maps* (dimensionality of layer output, 250 in Figure 1a). Downsampling enables efficient representation of long-range associations (and so more global information) in the text. By keeping the same number of feature maps, every 2-stride downsampling reduces the per-block computation by half and thus the total computation time is bounded by a constant.

- Shortcut connections with pre-activation and identity mapping (He et al., 2016) for enabling training of deep networks.

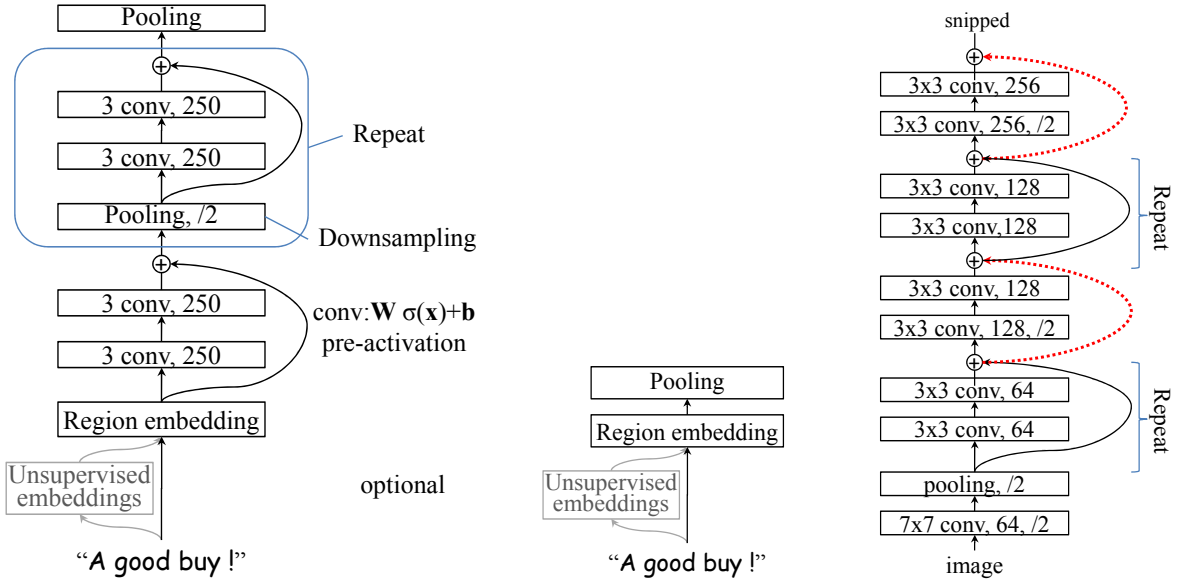
- Text region embedding enhanced with *unsupervised embeddings* (embeddings trained in an unsupervised manner) (Johnson and Zhang, 2015b) for improving accuracy.

2.1 Network architecture

Downsampling with the number of feature maps fixed After each convolution block, we perform max-pooling with size 3 and stride 2. That is, the pooling layer produces a new internal representation of a document by taking the component-wise maximum over 3 contiguous internal vectors, representing 3 overlapping text regions, but it does this only for every other possible triplet (stride 2) instead of all the possible triplets (stride 1). This 2-stride downsampling reduces the size of the internal representation of each document by half.

A number of models (Simonyan and Zisserman, 2015; He et al., 2015, 2016; Conneau et al., 2016) increase the number of feature maps whenever downsampling is performed, causing the total computational complexity to be a function of the depth. In contrast, we fix the number of feature maps, as we found that increasing the number of feature maps only does harm – increasing computation time substantially without accuracy improvement, as shown later in the experiments.

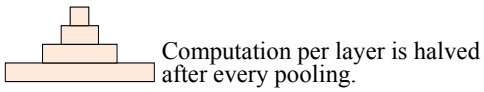
[1]Previous deep CNNs (either on image or text) tend to be more complex and irregular, having occasional increase of the number of feature maps.



(a) Our proposed model DPCNN (b) cf. ShallowCNN [JZ15b] (c) cf. ResNet for image [HZRS15]

Figure 1: (a) Our proposed model DPCNN. (b,c) Previous models for comparison. $\oplus$ indicates addition. The dotted red shortcuts in (c) perform dimension matching. DPCNN is dimension-matching free.

With the number of feature maps fixed, the computation time for each convolution layer is halved (as the data size is halved) whenever 2-stride downsampling is performed, thus, forming a '*pyramid*'.



Therefore, with DPCNNs, the total computation time is bounded by a constant – twice the computation time of a single block, which makes our deep pyramid networks computationally attractive.

In addition, downsampling with stride 2 essentially doubles the effective coverage (i.e., coverage in the original document) of the convolution kernel; therefore, after going through downsampling L times, associations among words within a distance in the order of 2^L can be represented. Thus, deep pyramid CNN is computationally efficient for representing long-range associations and so more global information.

Shortcut connections with pre-activation To enable training of deep networks, we use additive shortcut connections with identity mapping, which can be written as $\mathbf{z} + f(\mathbf{z})$ where f represents the skipped layers (He et al., 2016). In DPCNN, the skipped layers $f(\mathbf{z})$ are two convolution layers with *pre-activation*. Here, pre-activation refers to activation being done *before* weighting instead of *after* as is typically done. That is, in the convolution layer of DPCNN, $\mathbf{W}\sigma(\mathbf{x}) + \mathbf{b}$ is computed at every location of each document where a column vector $\mathbf{x}$ represents a small region (overlapping with each other) of input at each location, $\sigma(\cdot)$ is a component-wise nonlinear activation, and weights $\mathbf{W}$ and biases $\mathbf{b}$ (unique to each layer) are the parameters to be trained. The number of $\mathbf{W}$'s rows is the *number of feature maps* (also called the *number of filters* (He et al., 2015)) of this layer. We set activation $\sigma(\cdot)$ to the rectifier $\sigma(x) = \max(x, 0)$. In our implementation, we fixed the number of feature maps to 250 and the kernel size (the size of the small region covered by $\mathbf{x}$) to 3, as shown in Figure 1a.

With pre-activation, it is the results of linear weighting ($\mathbf{W}\sigma(\mathbf{x}) + \mathbf{b}$) that travel through the shortcut, and what is added to them at a $\oplus$ (Figure 1a) is also the results of linear weighting, instead of the results of nonlinear activation ($\sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$). Intuitively, such 'linearity' eases training of deep networks, similar to the role of *constant error carousels* in LSTM (Hochreiter and Schmidhuber, 1997). We empirically observed that pre-activation indeed outperformed 'post-activation', which is in line with the image results (He et al., 2016).

No need for dimension matching Although the shortcut with pre-activation was adopted from the *improved ResNet* of (He et al., 2016), our model is simpler than ResNet (Figure 1c), as all the

shortcuts are exactly simple *identity mapping* (i.e., passing data exactly as it is) without any complication for dimension matching. When a shortcut meets the 'main street', the data from two paths need to have the same dimensionality so that they can be added; therefore, if a shortcut skips a layer that changes the dimensionality, e.g., by downsampling or by use of a different number of feature maps, then a shortcut must perform dimension matching. Dimension matching for increased number of feature maps, in particular, is typically done by projection, introducing more weight parameters to be trained. We eliminate the complication of dimension matching by not letting any shortcut skip a downsampling layer, and by fixing the number of feature maps throughout the network. The latter also substantially saves computation time as mentioned above, and we will show later in our experiments that on our tasks, we do not sacrifice anything for such a substantial efficiency gain.

2.2 Text region embedding

A CNN for text categorization typically starts with converting each word in the text to a word vector (word embedding). We take a more general viewpoint as in (Johnson and Zhang, 2015b) and consider *text region embedding* – embedding of a region of text covering one or more words.

Basic region embedding We start with the basic setting where there is no unsupervised embedding. In the region embedding layer we compute $\mathbf{W}\mathbf{x} + \mathbf{b}$ for each word of a document where input $\mathbf{x}$ represents a k-word region (i.e., window) around the word in some straightforward manner, and weights $\mathbf{W}$ and bias $\mathbf{b}$ are trained with the parameters of other layers. Activation is delayed to the pre-activation of the next layer. Now let v be the size of vocabulary, and let us consider the following three types of straightforward representation of a k-word region for $\mathbf{x}$: (1) *sequential input*: the kv-dimensional concatenation of k one-hot vectors; (2) *bow input*: a v-dimensional bag-of-word (bow) vector; and (3) *bag-of-n-gram input*: e.g., a bag of word uni, bi, and trigrams contained in the region. Setting the region size $k = 1$, they all become word embedding.

A region embedding layer with the sequential input is equivalent to a convolution layer applied to a sequence of one-hot vectors representing a document, and this viewpoint was taken to describe the first layer of ShallowCNN in (Johnson and Zhang, 2015a,b). From the region embedding viewpoint, ShallowCNN is DPCNN's special case in which a region embedding layer is directly followed by the final pooling layer (Figure 1b).

A region embedding layer with region size $k > 1$ seeks to capture more complex concepts than single words in *one* weight layer, whereas a network with word embedding uses *multiple* weight layers to do this, e.g., word embedding followed by a convolution layer. In general, having fewer layers has a practical advantage of easier optimization. Beyond that, the optimum input type and the optimum region size can only be determined empirically. Our preliminary experiments indicated that when used with DPCNN (but not ShallowCNN), the sequential input has no advantage over the bow input – comparable accuracy with k times more weight parameters; therefore, we excluded the sequential input from our experiments[2]. The n-gram input turned out to be prone to overfitting in the supervised setting, likely due to its high representation power, but it is very useful as the input to unsupervised embeddings, which we discuss next.

Enhancing region embedding with unsupervised embeddings In (Johnson and Zhang, 2015b, 2016), it was shown that accuracy was substantially improved by extending ShallowCNN with unsupervised embeddings obtained by *tv-embedding* training ('tv' stands for *two views*). We found that accuracy of DPCNN can also be improved in this manner. Below we briefly review tv-embedding training and then describe how we use the resulting unsupervised embeddings with DPCNN.

The tv-embedding training requires two views. For text categorization, we define a region of text as view-1 and its adjacent regions as view-2. Then using unlabeled data, we train a neural network of one hidden layer with an artificial task of predicting view-2 from view-1. The obtained hidden layer, which is an embedding function that takes view-1 as input, serves as an unsupervised embedding function in the model for text categorization. In (Johnson and Zhang, 2015b), we showed theoretical conditions on views and labels under which

[2]This differs from ShallowCNN where the sequential input is often superior to bow input. We conjecture that when bow input is used in DPCNN, convolution layers following region embedding make up for the loss of local word order caused by bow input, as they use word order.

	AG	Sogou	Dbpedia	Yelp.p	Yelp.f	Yahoo	Ama.f	Ama.p
# of training documents	120K	450K	560K	560K	650K	1.4M	3M	3.6M
# of test documents	7.6K	60K	70K	38K	50K	60K	650K	400K
# of classes	4	5	14	2	5	10	5	2
Average #words	45	578	55	153	155	112	93	91

Table 1: Data. Note that Yelp.f is a balanced subset of Yelp 2015. The results on these two datasets are not comparable.

unsupervised embeddings obtained this way are useful for classification.

For use with DPCNN, we train several unsupervised embeddings in this manner, which differ from one another in the region size and the vector representations of view-1 (input region) so that we can benefit from diversity. The region embedding layer of DPCNN computes $\mathbf{W}\mathbf{x} + \sum_{u \in U} \mathbf{W}^{(u)}\mathbf{z}^{(u)} + \mathbf{b}$, where $\mathbf{x}$ is the discrete input as in the basic region embedding, and $\mathbf{z}^{(u)}$ is the output of an unsupervised embedding function indexed by u. We will show below that use of unsupervised embeddings in this way consistently improves the accuracy of DPCNN.

3 Experiments

We report the experiments with DPCNNs in comparison with previous models and alternatives. The code is publicly available on the internet.

3.1 Experimental setup

Data and data preprocessing To facilitate comparisons with previous results, we used the eight datasets compiled by Zhang et al. (2015), summarized in Table 1. AG and Sogou are news. Dbpedia is an ontology. Yahoo consists of questions and answers from the 'Yahoo! Answers' website. Yelp and Amazon ('Ama') are reviews where '.p' (polarity) in the names indicates that labels are binary (positive/negative), and '.f' (full) indicates that labels are the number of stars. Sogou is in Romanized Chinese, and the others are in English. Classes are balanced on all the datasets. Data preprocessing was done as in (Johnson and Zhang, 2016). That is, upper-case letters were converted to lower-case letters. Unlike (Kim, 2014; Zhang et al., 2015; Conneau et al., 2016), variable-sized documents were handled as variable-sized without any shortening or padding; however, the vocabulary size was limited to 30K words. For example, as also mentioned in (Johnson and Zhang, 2016), the complete vocabulary of the Ama.p training set contains 1.3M words. A vocabulary of 30K words is only a small portion of it, but it covers about 98% of the text and produced good accuracy as reported below.

Training protocol We held out 10K documents from the training data for use as a validation set on each dataset, and meta-parameter tuning was done based on the performance on the validation set.

To minimize a log loss with softmax, mini-batch SGD with momentum 0.9 was conducted for n epochs (n was fixed to 50 for AG, 30 for Yelp.f/p and Dbpedia, and 15 for the rest) while the learning rate was set to η for the first $\frac{4}{5}n$ epochs and then 0.1η for the rest[3]. The initial learning rate η was considered to be a meta-parameter. The mini-batch size was fixed to 100. Regularization was done by weight decay with the parameter 0.0001 and by optional dropout (Hinton et al., 2012) with 0.5 applied to the input to the top layer. In some cases overfitting was observed, and so we performed early stopping, based on the validation performance, after reducing the learning rate to 0.1η. Weights were initialized by the Gaussian distribution with zero mean and standard deviation 0.01. The discrete input to the region embedding layer was fixed to the bow input, and the region size was chosen from {1,3,5}, while fixing output dimensionality to 250 (same as convolution layers).

Details of unsupervised embedding training To facilitate comparison with ShallowCNN, we matched our unsupervised embedding setting exactly with that of (Johnson and Zhang, 2016). That is, we trained the same four types of tv-embeddings, which are embeddings of 5- and 9-word regions, each of which represents the input regions by either 30K-dim bow or 200K-dim

[3]This learning rate scheduling method was used also in (Johnson and Zhang, 2015a,b, 2016). It was meant to reduce learning rate when error plateaus, as is often done on image tasks, e.g., (He et al., 2015), though for simplicity, the timing of reduction was fixed for each dataset.

	Models	Deep	Unsup. embed.	Yelp.p	Yelp.f	Yahoo	Ama.f	Ama.p
1	DPCNN + unsupervised embed.	✓	tv	**2.64**	**30.58**	**23.90**	**34.81**	**3.32**
2	ShallowCNN + unsup. embed. [JZ16]		tv	*2.90*	*32.39*	24.85	*36.24*	*3.79*
3	Hierarchical attention net [YYDHSH16]	✓	w2v	–	–	*24.2*	36.4	–
4	[CSBL16]'s char-level CNN: *best*	✓		4.28	35.28	26.57	37.00	4.28
5	fastText bigrams (Joulin et al., 2016)			4.3	36.1	27.7	39.8	5.4
6	[ZZL15]'s char-level CNN: *best*	✓		4.88	37.95	28.80	40.43	4.93
7	[ZZL15]'s word-level CNN: *best*	✓	(w2v)	4.60	39.58	28.84	42.39	5.51
8	[ZZL15]'s linear model: *best*			4.36	40.14	28.96	44.74	7.98

Table 2: Error rates (%) on larger datasets in comparison with previous models. The previous results are roughly sorted in the order of error rates (best to worst). The best results and the second best are shown in bold and italic, respectively. 'tv' stands for tv-embeddings. 'w2v' stands for word2vec. '(w2v)' in row 7 indicates that the best results among those with and without word2vec pretraining are shown. Note that '*best*' in rows 4&6–8 indicates that we are giving an 'unfair' advantage to these models by choosing the *best test error rate* among a number of variations presented in the respective papers.
[JZ16]: Johnson and Zhang (2016), [YYDHSH16]: Yang et al. (2016), [CSBL16]: Conneau et al. (2016), [ZZL15]: Zhang et al. (2015)

bags of {1,2,3}-grams, retaining only the most frequent 30K words or 200K {1,2,3}-grams. Training was done on the labeled data (disregarding the labels), setting the training objectives to the prediction of adjacent regions of the same size as the input region (i.e., 5 or 9). Weighted square loss $\sum_{i,j} \alpha_{i,j}(\mathbf{z}_i[j] - \mathbf{p}_i[j])^2$ was minimized where i goes through instances, $\mathbf{z}$ represents the target regions by bow, $\mathbf{p}$ is the model output, and the weights $\alpha_{i,j}$ were set to achieve the negative sampling effect. The dimensionality of unsupervised embeddings was set to 300 unless otherwise specified. Unsupervised embeddings were fixed during the supervised training – no fine-tuning.

3.2 Results

In the results below, the *depth* of DPCNN was fixed to 15 unless otherwise specified. Making it deeper did not substantially improve or degrade accuracy. Note that we count as *depth* the number of hidden weight layers including the region embedding layer but excluding unsupervised embeddings, therefore, 15 means 7 convolution blocks of 2 layers plus 1 layer for region embedding.

3.2.1 Main results

Large data results We first report the error rates of our full model (DPCNN with 15 weight layers plus unsupervised embeddings) on the larger five datasets (Table 2). To put it into perspective, we also show the previous results in the literature. The previous results are roughly sorted in the order of error rates from best to worst. On all the five datasets, DPCNN outperforms all of the previous results, which validates the effectiveness of our approach.

DPCNN can be regarded as a deep extension of ShallowCNN (row 2), sharing region embedding enhancement with diverse unsupervised embeddings. Note that ShallowCNN enhanced with unsupervised embeddings (row 2) was originally proposed in (Johnson and Zhang, 2015b) as a semi-supervised extension of (Johnson and Zhang, 2015a), and then it was tested on the large datasets in (Johnson and Zhang, 2016). The performance improvements of DPCNN over ShallowCNN indicates that the added depth is indeed useful, capturing more global information. Yang et al. (2016)'s hierarchical attention network (row 3) consists of RNNs in the word level and the sentence level. It is more complex than DPCNN due to the use of RNNs and linguistic knowledge for sentence segmentation. Similarly, Tang et al. (2015) proposed to use CNN or LSTM to represent each sentence in documents and then use RNNs. Although we do not have direct comparison with Tang et al.'s model, Yang et al. (2016) reports that their model outperformed Tang et al.'s model. Conneau et al. (2016) and Zhang et al. (2015) proposed deep character-level CNNs (row 4&6). Their models underperform our DPCNN with relatively large differences in spite of their deepness. Our mod-

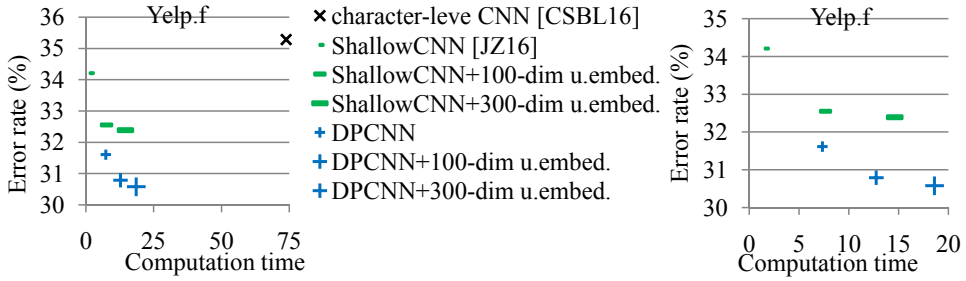


Figure 2: Error rates and computation time. DPCNN, ShallowCNN, and Conneau et al. (2016)'s character-level CNN. The x-axis is the time in seconds spent for categorizing 10K documents using our implementation on Tesla M2070. The right figure is a close-up of $x \in [0, 20]$ of the left figure. Though shown on one particular dataset Yelp.f, the trend is the same on the other four large datasets.

els are word-level and therefore use the knowledge of word boundaries which character-level models have no access to. While this is arguably not an apple-to-apple comparison, since word boundaries can be obtained for free in many languages, we view our model as much more useful in practice. Row 7 shows the performance of deep word-level CNN from (Zhang et al., 2015), which was designed to match their character-level models in complexity. Its relatively poor performance shows that it is not easy to design a high-performance deep word-level CNN.

Computation time In Figure 2, we plot error rates in relation to the computation time – the time spent for categorizing 10K documents using our implementation on a GPU. The right figure is a close-up of $x \in [0, 20]$ of the left figure. It stands out in the left figure that the character-level CNN of (Conneau et al., 2016) is much slower than DPCNNs. This is partly because it increases the number of feature maps with downsampling (i.e., no *pyramid*) while it is deeper (32 weight layers), and partly because it deals with characters – there are more characters than words in each document. DPCNNs are more accurate than ShallowCNNs at the expense of more computation time due to the depth (15 layers vs. 1 layer). Nevertheless, their computation time is comparable – the points of both fit in the same range $[0, 20]$. The efficiency of DPCNNs is due to the exponential decrease of per-layer computation due to downsampling with the number of feature maps being fixed.

Comparison with non-pyramid variants Furthermore, we tested the following two 'non-pyramid' models for comparison. The first model doubles the number of feature maps at every other downsampling so that per-layer computation is kept approximately constant[4]. The second model performs no downsampling. Otherwise, these two models are the same as DPCNN. We show in Figure 3 the error rates of these two variations (labeled as 'Increase #feature maps' and 'No downsampling', respectively) in comparison with DPCNN. The x-axis is the computation time, measured by the seconds spent for categorizing 10K documents. For all types, the models of depth 11 and 15 are shown. Clearly, DPCNN is more accurate and computes faster than the others. Figure 3 is on Yelp.f, and we observed the same performance trend on the other four large datasets.

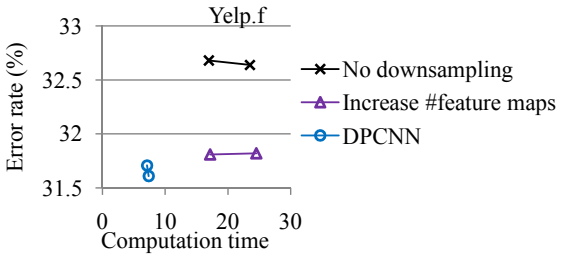


Figure 3: Comparison with non-pyramid models. Models of depth 11 and 15 are shown. No unsupervised embeddings.

Small data results Now we turn to the results on the three smaller datasets in Table 3. Again, the previous models are roughly sorted from best to worst. For these small datasets, the DPCNN performances with 100-dim unsupervised embed-

[4]Note that if we double the number of feature maps, it would increase the computation cost of the next layer by 4 times as it doubles the dimensionality of both input and output. On image, downsampling with stride 2 cancels it out as it makes data 4 times smaller by shrinking both horizontally and vertically, but text is one dimensional, and so downsampling with stride 2 merely halves data. That is why we doubled the number of feature maps at every other downsampling instead of at every downsampling to avoid exponential increase of computation time.

	Models	Deep	Unsup. embed.	AG	Sogou	Dbpedia
1	DPCNN + unsupervised embed.	✓	tv	*6.87*	**1.84**	*0.88*
2	ShallowCNN + unsup. embed. [JZ16]		tv	**6.57**	*1.89*	**0.84**
3	[ZZL15]'s linear model: *best*			7.64	2.81	1.31
4	[CSBL16]'s deep char-level CNN: *best*	✓		8.67	3.18	1.29
5	fastText bigrams (Joulin et al., 2016)			7.5	3.2	1.4
6	[ZZL15]'s word-level CNN : *best*	✓	(w2v)	8.55	4.39	1.37
7	[ZZL15]'s deep char-level CNN: *best*	✓		9.51	4.88	1.55

Table 3: Error rates (%) on smaller datasets in comparison with previous models. The previous results are roughly sorted in the order of error rates (best to worst). Notation follows that of Table 2.

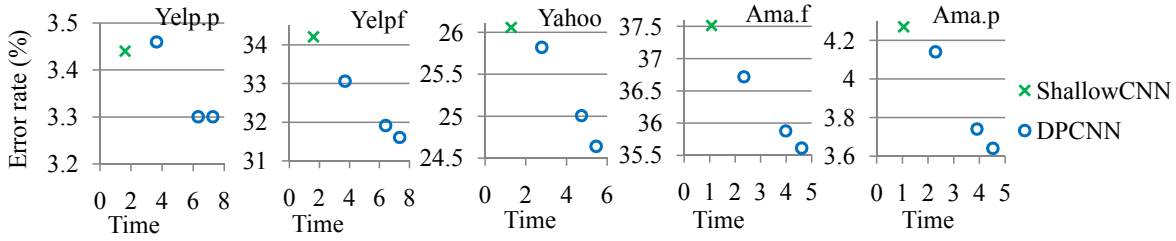


Figure 4: Error rates of DPCNNs with various depths (3, 7, and 15). The x-axis is computation time. No unsupervised embeddings.

dings are shown, which turned out to be as good as those with 300-dim unsupervised embeddings. One difference from the large dataset results is that the strength of shallow models stands out. ShallowCNN (row 2) rivals DPCNN (row 1), and Zhang et al.'s best linear model (row 3) moved up from the worst performer to the third best performer. The results are in line with the general fact that more complex models require more training data, and with the paucity of training data, simpler models can outperform more complex ones.

3.2.2 Empirical studies

We present some empirical results to validate the design choices. For this purpose, the larger five datasets were used to avoid the paucity of training data.

Depth Figure 4 shows error rates of DPCNNs with 3, 7, and 15 weight layers (blue circles from left to right). For comparison, the ShallowCNN results (green 'x') from (Johnson and Zhang, 2016) are also shown. The x-axis represents the computation time (seconds for categorizing 10K documents on a GPU). For simplicity, the results without unsupervised embeddings are shown for all. The error rate improves as the depth increases. The results confirm the effectiveness of our strategy of deepening the network.

Unsupervised embeddings To study the effectiveness of unsupervised embeddings, we experimented with variations of DPCNN that differ only in whether/how to use unsupervised embeddings (Table 4). First, we compare DPCNNs with and without unsupervised embeddings. The model with unsupervised embeddings (row 1, copied from Table 2 for easy comparison) clearly outperforms the one without them (row 4), which confirms the effectiveness of the use of unsupervised embeddings. Second, in the proposed model (row 1), a region embedding layer receives two types of input, the output of unsupervised embedding functions and the high-dimensional discrete input such as a bow vector. Row 2 shows the results obtained by using unsupervised embeddings to produce sole input (i.e., no discrete vectors provided to the region embedding layer). Degradations of error rates are up to 0.32%, small but consistent. Since the discrete input add almost no computation cost due to its sparseness, its use is desirable. Third, a number of previous studies used unsupervised word embedding to initialize word embedding in neural networks and then fine-tune it as training proceeds (pretraining). The model in row 3 does this with DPCNN using word2vec (Mikolov et al., 2013). The word2vec training was done on the training data (ignoring the labels),

	Unsupervised embeddings	Yelp.p	Yelp.f	Yahoo	Ama.f	Ama.p
1	tv-embed. (additional input)	**2.64**	**30.58**	**23.90**	**34.81**	**3.32**
2	tv-embed. (sole input)	2.68	30.66	24.09	35.13	3.45
3	word2vec (pretraining)	2.93	32.08	24.11	35.30	3.65
4	–	3.30	31.61	24.64	35.61	3.64

Table 4: Error rates (%) of DPCNN variations that differ in use of unsupervised embeddings. The rows are roughly sorted from best to worst.

same as tv-embedding training. This model (row 3) underperformed our proposed model (row 1). We attribute the superiority of the proposed model to its use of richer information than a word embedding. These results support our approach.

4 Conclusion

This paper tackled the problem of designing high-performance deep word-level CNNs for text categorization in the large training data setting. We proposed a deep pyramid CNN model which has low computational complexity, and can efficiently represent long-range associations in text and so more global information. It was shown to outperform the previous best models on six benchmark datasets.

References

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun. 2016. Very deep convolutional networks for natural language processing. *arXiv:1606.01781v1 (6 June 2016 version)* .

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv:1512.03385* .

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. *arXiv:1603.05027* .

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580* .

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

Rie Johnson and Tong Zhang. 2015a. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL HLT)*.

Rie Johnson and Tong Zhang. 2015b. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*.

Rie Johnson and Tong Zhang. 2016. Convolutional neural networks for text categorization: Shallow word-level vs. deep character-level. *arXiv:1609.00718* .

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv:1607.01795v3 (9 Aug 2016 version)* .

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*. pages 1746–1751.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL HLT)*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*.

Improved Neural Relation Detection for Knowledge Base Question Answering

Mo Yu[†] **Wenpeng Yin**[⋆] **Kazi Saidul Hasan**[‡] **Cicero dos Santos**[†]
Bing Xiang[‡] **Bowen Zhou**[†]
[†]AI Foundations, IBM Research, USA
[⋆]Center for Information and Language Processing, LMU Munich
[‡]IBM Watson, USA
{yum,kshasan,cicerons,bingxia,zhou}@us.ibm.com, wenpeng@cis.lmu.de

Abstract

Relation detection is a core component of many NLP applications including Knowledge Base Question Answering (KBQA). In this paper, we propose a hierarchical recurrent neural network enhanced by residual learning which detects KB relations given an input question. Our method uses deep residual bidirectional LSTMs to compare questions and relation names via different levels of abstraction. Additionally, we propose a simple KBQA system that integrates entity linking and our proposed relation detector to make the two components enhance each other. Our experimental results show that our approach not only achieves outstanding relation detection performance, but more importantly, it helps our KBQA system achieve state-of-the-art accuracy for both single-relation (SimpleQuestions) and multi-relation (WebQSP) QA benchmarks.

1 Introduction

Knowledge Base Question Answering (KBQA) systems answer questions by obtaining information from KB tuples (Berant et al., 2013; Yao et al., 2014; Bordes et al., 2015; Bast and Haussmann, 2015; Yih et al., 2015; Xu et al., 2016). For an input question, these systems typically generate a KB query, which can be executed to retrieve the answers from a KB. Figure 1 illustrates the process used to parse two sample questions in a KBQA system: (a) a single-relation question, which can be answered with a single <*head-entity, relation, tail-entity*> KB tuple (Fader et al., 2013; Yih et al., 2014; Bordes et al., 2015); and (b) a more complex case, where some constraints need to be handled for multiple entities in the question. The KBQA system in the figure performs two key tasks: (1) *entity linking*, which links n-grams in questions to KB entities, and (2) *relation detection*, which identifies the KB relation(s) a question refers to.

The main focus of this work is to improve the *relation detection* subtask and further explore how it can contribute to the KBQA system. Although general relation detection[1] methods are well studied in the NLP community, such studies usually do not take the end task of KBQA into consideration. As a result, there is a significant gap between general relation detection studies and KB-specific relation detection. First, in most general relation detection tasks, the number of target relations is limited, normally smaller than 100. In contrast, in KBQA even a small KB, like Freebase2M (Bordes et al., 2015), contains more than 6,000 relation types. Second, relation detection for KBQA often becomes a zero-shot learning task, since some test instances may have unseen relations in the training data. For example, the SimpleQuestions (Bordes et al., 2015) data set has 14% of the golden test relations not observed in golden training tuples. Third, as shown in Figure 1(b), for some KBQA tasks like WebQuestions (Berant et al., 2013), we need to predict a chain of relations instead of a single relation. This increases the number of target relation types and the sizes of candidate relation pools, further increasing the difficulty of KB relation detection. Owing to these reasons, KB relation detection is significantly more challenging compared to general relation detection tasks.

This paper improves KB relation detection to cope with the problems mentioned above. First, in order to deal with the unseen relations, we propose to break the relation names into word sequences for question-relation matching. Second, noticing

[1]In the information extraction field such tasks are usually called *relation extraction* or *relation classification*.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 571–581
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1053

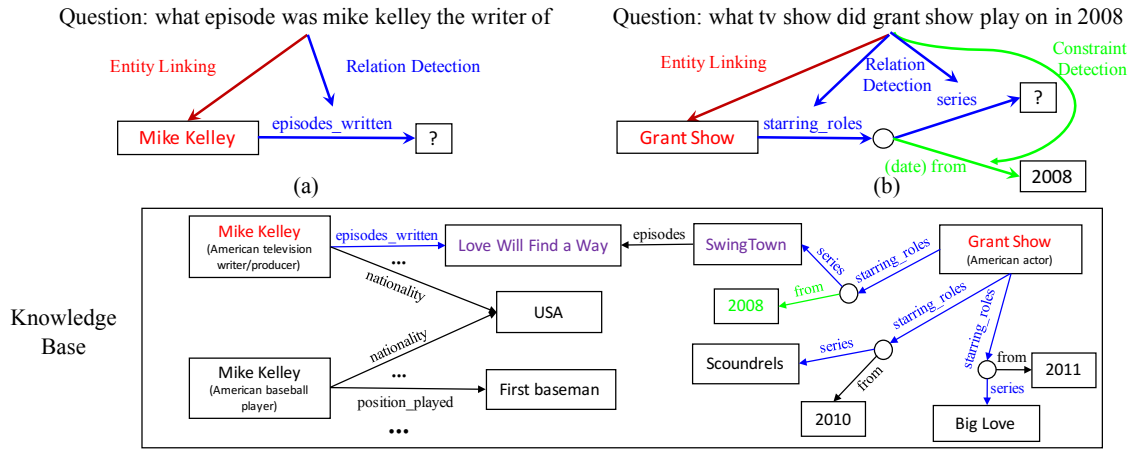


Figure 1: KBQA examples and its three key components. (a) A single relation example. We first identify the topic entity with ***entity linking*** and then detect the relation asked by the question with ***relation detection*** (from all relations connecting the topic entity). Based on the detected entity and relation, we form a query to search the KB for the correct answer "*Love Will Find a Way*". (b) A more complex question containing two entities. By using "*Grant Show*" as the topic entity, we could detect a chain of relations "*starring_roles-series*" pointing to the answer. An additional ***constraint detection*** takes the other entity "*2008*" as a constraint, to filter the correct answer "*SwingTown*" from all candidates found by the topic entity and relation.

that original relation names can sometimes help to match longer question contexts, we propose to build both relation-level and word-level relation representations. Third, we use deep bidirectional LSTMs (*BiLSTM*s) to learn different levels of question representations in order to match the different levels of relation information. Finally, we propose a residual learning method for sequence matching, which makes the model training easier and results in more abstract (deeper) question representations, thus improves hierarchical matching.

In order to assess how the proposed *improved relation detection* could benefit the KBQA end task, we also propose a simple KBQA implementation composed of *two-step relation detection*. Given an input question and a set of candidate entities retrieved by an entity linker based on the question, our proposed relation detection model plays a key role in the KBQA process: (1) Re-ranking the entity candidates according to whether they connect to high confident relations detected from the *raw question text* by the relation detection model. This step is important to deal with the ambiguities normally present in entity linking results. (2) Finding the core relation (chains) for each *topic entity*[2] selection from a much smaller candidate entity set after re-ranking. The above steps are followed by an optional constraint detection step, when the question cannot be answered by single relations (e.g., multiple entities in the question). Finally the highest scored query from the above steps is used to query the KB for answers.

Our main contributions include: (i) An improved relation detection model by hierarchical matching between questions and relations with residual learning; (ii) We demonstrate that the improved relation detector enables our simple KBQA system to achieve state-of-the-art results on both single-relation and multi-relation KBQA tasks.

2 Related Work

Relation Extraction Relation extraction (RE) is an important sub-field of information extraction. General research in this field usually works on a (small) pre-defined relation set, where given a text paragraph and two target entities, the goal is to determine whether the text indicates any types of relations between the entities or not. As a result RE is usually formulated as a **classification task**. Traditional RE methods rely on large amount of hand-crafted features (Zhou et al., 2005; Rink and Harabagiu, 2010; Sun et al., 2011). Recent research benefits a lot from the advancement of deep learning: from word embeddings (Nguyen and Grishman, 2014; Gormley et al., 2015) to deep networks like CNNs and LSTMs (Zeng et al., 2014; dos Santos et al., 2015; Vu et al., 2016) and attention models (Zhou et al., 2016; Wang et al., 2016).

The above research assumes there is a fixed (closed) set of relation types, thus no zero-shot learning capability is required. The number of relations is usually not large: The widely used ACE2005 has 11/32 coarse/fine-grained relations; SemEval2010 Task8 has 19 relations; TAC-

[2]Following Yih et al. (2015), here *topic entity* refers to the root of the (directed) query tree; and *core-chain* is the directed path of relation from root to the answer node.

KBP2015 has 74 relations although it considers open-domain Wikipedia relations. All are much fewer than thousands of relations in KBQA. As a result, few work in this field focuses on dealing with large number of relations or unseen relations. Yu et al. (2016) proposed to use relation embeddings in a low-rank tensor method. However their relation embeddings are still trained in supervised way and the number of relations is not large in the experiments.

Relation Detection in KBQA Systems Relation detection for KBQA also starts with feature-rich approaches (Yao and Van Durme, 2014; Bast and Haussmann, 2015) towards usages of deep networks (Yih et al., 2015; Xu et al., 2016; Dai et al., 2016) and attention models (Yin et al., 2016; Golub and He, 2016). Many of the above relation detection research could naturally support large relation vocabulary and open relation sets (especially for QA with OpenIE KB like ParaLex (Fader et al., 2013)), in order to fit the goal of open-domain question answering.

Different KBQA data sets have different levels of requirement about the above open-domain capacity. For example, most of the gold test relations in WebQuestions can be observed during training, thus some prior work on this task adopted the close domain assumption like in the general RE research. While for data sets like SimpleQuestions and ParaLex, the capacity to support large relation sets and unseen relations becomes more necessary. To the end, there are two main solutions: (1) use pre-trained relation embeddings (e.g. from TransE (Bordes et al., 2013)), like (Dai et al., 2016); (2) factorize the relation names to sequences and formulate relation detection as a **sequence matching and ranking** task. Such factorization works because that the relation names usually comprise meaningful word sequences. For example, Yin et al. (2016) split relations to word sequences for single-relation detection. Liang et al. (2016) also achieve good performance on WebQSP with word-level relation representation in an end-to-end neural programmer model. Yih et al. (2015) use character tri-grams as inputs on both question and relation sides. Golub and He (2016) propose a generative framework for single-relation KBQA which predicts relation with a character-level sequence-to-sequence model.

Another difference between relation detection in KBQA and general RE is that general RE research assumes that the two argument entities are both available. Thus it usually benefits from features (Nguyen and Grishman, 2014; Gormley et al., 2015) or attention mechanisms (Wang et al., 2016) based on the entity information (e.g. entity types or entity embeddings). For relation detection in KBQA, such information is mostly missing because: (1) one question usually contains single argument (the topic entity) and (2) one KB entity could have multiple types (type vocabulary size larger than 1,500). This makes KB entity typing itself a difficult problem so no previous used entity information in the relation detection model.[3]

3 Background: Different Granularity in KB Relations

Previous research (Yih et al., 2015; Yin et al., 2016) formulates KB relation detection as a sequence matching problem. However, while the questions are natural word sequences, how to represent relations as sequences remains a challenging problem. Here we give an overview of two types of relation sequence representations commonly used in previous work.

(1) Relation Name as a Single Token (*relation-level*). In this case, each relation name is treated as a unique token. The problem with this approach is that it suffers from the low relation coverage due to limited amount of training data, thus cannot generalize well to large number of open-domain relations. For example, in Figure 1, when treating relation names as single tokens, it will be difficult to match the questions to relation names "*episodes_written*" and "*starring_roles*" if these names do not appear in training data – their relation embeddings $\mathbf{h}^r$s will be random vectors thus are not comparable to question embeddings $\mathbf{h}^q$s.

(2) Relation as Word Sequence (*word-level*). In this case, the relation is treated as a sequence of words from the tokenized relation name. It has better generalization, but suffers from the lack of global information from the original relation names. For example in Figure 1(b), when doing only word-level matching, it is difficult to rank the target relation "*starring_roles*" higher compared to the incorrect relation "*plays_produced*". This is because the incorrect relation contains word "*plays*", which is more similar to the question

[3] Such entity information has been used in KBQA systems as features for the final answer re-rankers.

	Relation Token	Question 1	Question 2
		what tv episodes were <e> the writer of	what episode was written by <e>
relation-level	episodes_written	*tv episodes were <e> the writer of*	*episode was written by <e>*
word-level	episodes	*tv episodes*	*episode*
	written	*the writer of*	*written*

Table 1: An example of KB relation (*episodes_written*) with two types of relation tokens (relation names and words), and two questions asking this relation. The topic entity is replaced with token $<e>$ which could give the position information to the deep networks. The italics show the evidence phrase for each relation token in the question.

(containing word "*play*") in the embedding space. On the other hand, if the target relation co-occurs with questions related to "*tv appearance*" in training, by treating the whole relation as a token (i.e. relation id), we could better learn the correspondence between this token and phrases like "*tv show*" and "*play on*".

The two types of relation representation contain different levels of abstraction. As shown in Table 1, the word-level focuses more on local information (words and short phrases), and the relation-level focus more on global information (long phrases and skip-grams) but suffer from data sparsity. Since both these levels of granularity have their own pros and cons, we propose a hierarchical matching approach for KB relation detection: for a candidate relation, our approach matches the input question to both word-level and relation-level representations to get the final ranking score. Section 4 gives the details of our proposed approach.

4 Improved KB Relation Detection

This section describes our hierarchical sequence matching with residual learning approach for relation detection. In order to match the question to different aspects of a relation (with different abstraction levels), we deal with three problems as follows on learning question/relation representations.

4.1 Relation Representations from Different Granularity

We provide our model with both types of relation representation: word-level and relation-level. Therefore, the input relation becomes $\mathbf{r} = \{r_1^{word}, \cdots, r_{M_1}^{word}\} \cup \{r_1^{rel}, \cdots, r_{M_2}^{rel}\}$, where the first M_1 tokens are words (e.g. $\{$*episode, written*$\}$), and the last M_2 tokens are relation names, e.g., $\{$*episode_written*$\}$ or $\{$*starring_roles, series*$\}$ (when the target is a chain like in Figure 1(b)). We transform each token above to its word embedding then use two BiLSTMs (with shared parameters) to get their hidden representations $[\mathbf{B}_{1:M_1}^{word} : \mathbf{B}_{1:M_2}^{rel}]$ (each row vector $\boldsymbol{\beta}_i$ is the concatenation between forward/backward representations at i). We initialize the relation sequence LSTMs with the final state representations of the word sequence, as a back-off for unseen relations. We apply *one* max-pooling on these two sets of vectors and get the final relation representation $\mathbf{h}^r$.

4.2 Different Abstractions of Questions Representations

From Table 1, we can see that different parts of a relation could match different contexts of question texts. Usually relation names could match longer phrases in the question and relation words could match short phrases. Yet different words might match phrases of different lengths.

As a result, we hope the question representations could also comprise vectors that summarize various lengths of phrase information (different levels of abstraction), in order to match relation representations of different granularity. We deal with this problem by applying deep BiLSTMs on questions. The first-layer of BiLSTM works on the word embeddings of question words $\mathbf{q} = \{q_1, \cdots, q_N\}$ and gets hidden representations $\mathbf{\Gamma}_{1:N}^{(1)} = [\boldsymbol{\gamma}_1^{(1)}; \cdots; \boldsymbol{\gamma}_N^{(1)}]$. The second-layer BiLSTM works on $\mathbf{\Gamma}_{1:N}^{(1)}$ to get the second set of hidden representations $\mathbf{\Gamma}_{1:N}^{(2)}$. Since the second BiLSTM starts with the hidden vectors from the first layer, intuitively it could learn more general and abstract information compared to the first layer.

Note that the first(second)-layer of question representations does not necessarily correspond to the word(relation)-level relation representations, instead either layer of question representations could potentially match to either level of relation representations. This raises the difficulty of matching between different levels of relation/question representations; the following section gives our proposal to deal with such problem.

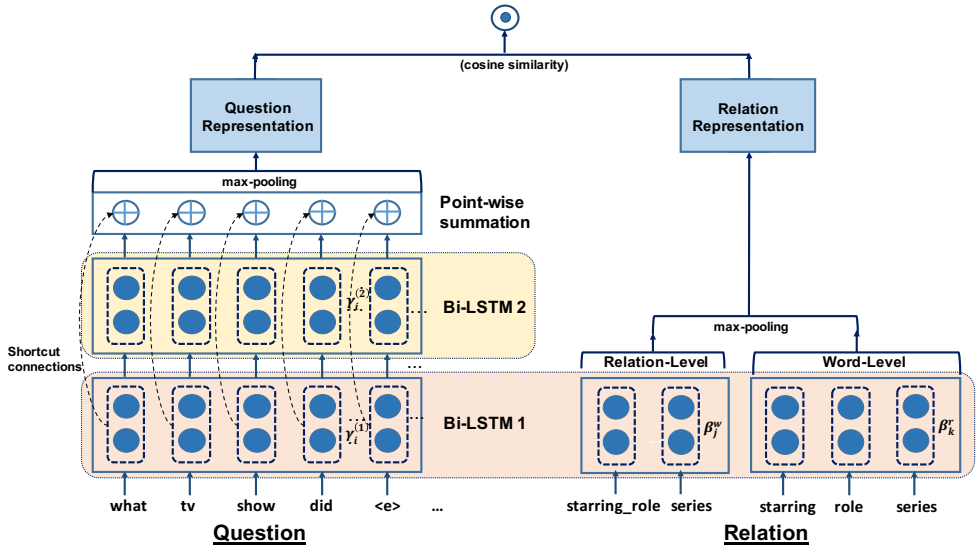


Figure 2: The proposed Hierarchical Residual BiLSTM (HR-BiLSTM) model for relation detection. Note that without the dotted arrows of shortcut connections between two layers, the model will only compute the similarity between the second-layer of questions representations and the relation, thus is not doing hierarchical matching.

4.3 Hierarchical Matching between Relation and Question

Now we have question contexts of different lengths encoded in $\mathbf{\Gamma}^{(1)}_{1:N}$ and $\mathbf{\Gamma}^{(2)}_{1:N}$. Unlike the standard usage of deep BiLSTMs that employs the representations in the final layer for prediction, here we expect that two layers of question representations can be complementary to each other and both should be compared to the relation representation space (*Hierarchical Matching*). This is important for our task since each relation token can correspond to phrases of different lengths, mainly because of syntactic variations. For example in Table 1, the relation word *written* could be matched to either the same single word in the question or a much longer phrase *be the writer of*.

We could perform the above hierarchical matching by computing the similarity between each layer of $\mathbf{\Gamma}$ and $\mathbf{h}^r$ separately and doing the (weighted) sum between the two scores. However this does not give significant improvement (see Table 2). Our analysis in Section 6.2 shows that this naive method suffers from the training difficulty, evidenced by that the converged training loss of this model is much higher than that of a single-layer baseline model. This is mainly because (1) Deep BiLSTMs do not guarantee that the two-levels of question hidden representations are comparable, the training usually falls to local optima where one layer has good matching scores and the other always has weight close to 0. (2) The training of deeper architectures itself is more difficult.

To overcome the above difficulties, we adopt the idea from Residual Networks (He et al., 2016) for hierarchical matching by adding shortcut connections between two BiLSTM layers. We proposed two ways of such *Hierarchical Residual Matching*: (1) Connecting each $\boldsymbol{\gamma}^{(1)}_i$ and $\boldsymbol{\gamma}^{(2)}_i$, resulting in a $\boldsymbol{\gamma}'_i = \boldsymbol{\gamma}^{(1)}_i + \boldsymbol{\gamma}^{(2)}_i$ for each position i. Then the final question representation $\mathbf{h}^q$ becomes a max-pooling over all $\boldsymbol{\gamma}'_i$s, $1 \leq i \leq N$. (2) Applying max-pooling on $\mathbf{\Gamma}^{(1)}_{1:N}$ and $\mathbf{\Gamma}^{(2)}_{1:N}$ to get $\mathbf{h}^{(1)}_{max}$ and $\mathbf{h}^{(2)}_{max}$, respectively, then setting $\mathbf{h}^q = \mathbf{h}^{(1)}_{max} + \mathbf{h}^{(2)}_{max}$. Finally we compute the matching score of $\mathbf{r}$ given $\mathbf{q}$ as $s_{rel}(\mathbf{r}; \mathbf{q}) = cos(\mathbf{h}^r, \mathbf{h}^q)$.

Intuitively, the proposed method should benefit from hierarchical training since the second layer is fitting the residues from the first layer of matching, so the two layers of representations are more likely to be complementary to each other. This also ensures the vector spaces of two layers are comparable and makes the second-layer training easier.

During training we adopt a ranking loss to maximizing the margin between the gold relation $\mathbf{r}^+$ and other relations $\mathbf{r}^-$ in the candidate pool R.

$$l_{\text{rel}} = \max\{0, \gamma - s_{\text{rel}}(\mathbf{r}^+; \mathbf{q}) + s_{\text{rel}}(\mathbf{r}^-; \mathbf{q})\}$$

where γ is a constant parameter. Fig 2 summarizes the above *Hierarchical Residual BiLSTM* (***HR-BiLSTM***) model.

Remark: Another way of hierarchical matching consists in relying on **attention mechanism**, e.g. (Parikh et al., 2016), to find the correspondence between different levels of representations. This performs below the HR-BiLSTM (see Table 2).

5 KBQA Enhanced by Relation Detection

This section describes our KBQA pipeline system. We make minimal efforts beyond the training of the relation detection model, making the whole system easy to build.

Following previous work (Yih et al., 2015; Xu et al., 2016), our KBQA system takes an existing entity linker to produce the top-K linked entities, $EL_K(q)$, for a question q ("*initial entity linking*"). Then we generate the KB queries for q following the four steps illustrated in Algorithm 1.

Algorithm 1: KBQA with two-step relation detection

Input : Question q, Knowledge Base KB, the initial top-K entity candidates $EL_K(q)$
Output: Top query tuple $(\hat{e}, \hat{r}, \{(c, r_c)\})$

1. **Entity Re-Ranking** (*first-step relation detection*): Use the *raw question text* as input for a relation detector to score all relations in the KB that are associated to the entities in $EL_K(q)$; use the relation scores to re-rank $EL_K(q)$ and generate a shorter list $EL'_{K'}(q)$ containing the top-K' entity candidates (Section 5.1)
2. **Relation Detection**: Detect relation(s) using the *reformatted question text* in which the topic entity is replaced by a special token <*e*> (Section 5.2)
3. **Query Generation**: Combine the scores from step 1 and 2, and select the top pair $(\hat{e}, \hat{r})$ (Section 5.3)
4. **Constraint Detection** (optional): Compute similarity between q and any neighbor entity c of the entities along $\hat{r}$ (connecting by a relation r_c) , add the high scoring c and r_c to the query (Section 5.4).

Compared to previous approaches, the main difference is that we have an additional *entity re-ranking* step after the *initial entity linking*. We have this step because we have observed that entity linking sometimes becomes a bottleneck in KBQA systems. For example, on SimpleQuestions the best reported linker could only get 72.7% top-1 accuracy on identifying topic entities. This is usually due to the ambiguities of entity names, e.g. in Fig 1(a), there are *TV writer* and *baseball player* "*Mike Kelley*", which is impossible to distinguish with only entity name matching.

Having observed that different entity candidates usually connect to different relations, here we propose to help entity disambiguation in the *initial entity linking* with relations detected in questions.

Sections 5.1 and 5.2 elaborate how our relation detection help to re-rank entities in the initial entity linking, and then those re-ranked entities enable more accurate relation detection. The KBQA end task, as a result, benefits from this process.

5.1 Entity Re-Ranking

In this step, we use the *raw question text* as input for a relation detector to score all relations in the KB with connections to at least one of the entity candidates in $EL_K(q)$. We call this step *relation detection on entity set* since it does not work on a single topic entity as the usual settings. We use the HR-BiLSTM as described in Sec. 4. For each question q, after generating a score $s_{rel}(r; q)$ for each relation using HR-BiLSTM, we use the top l best scoring relations (R_q^l) to re-rank the original entity candidates. Concretely, for each entity e and its associated relations R_e, given the original entity linker score s_{linker}, and the score of the most confident relation $r \in R_q^l \cap R_e$, we sum these two scores to re-rank the entities:

$$s_{\text{rerank}}(e; q) = \alpha \cdot s_{\text{linker}}(e; q) + (1 - \alpha) \cdot \max_{r \in R_q^l \cap R_e} s_{\text{rel}}(r; q).$$

Finally, we select top $K' < K$ entities according to score s_{rerank} to form the re-ranked list $EL'_{K'}(q)$.

We use the same example in Fig 1(a) to illustrate the idea. Given the input question in the example, a relation detector is very likely to assign high scores to relations such as "*episodes_written*", "*author_of*" and "*profession*". Then, according to the connections of entity candidates in KB, we find that the TV writer "*Mike Kelley*" will be scored higher than the baseball player "*Mike Kelley*", because the former has the relations "*episodes_written*" and "*profession*". This method can be viewed as exploiting entity-relation collocation for entity linking.

5.2 Relation Detection

In this step, for each candidate entity $e \in EL'_K(q)$, we use the question text as the input to a relation detector to score all the relations $r \in R_e$ that are associated to the entity e in the KB.[4] Because we have a single topic entity input in this step, we do the following question reformatting: we replace the the candidate e's entity mention in

[4]Note that the number of entities and the number of relation candidates will be much smaller than those in the previous step.

q with a token "<e>". This helps the model better distinguish the relative position of each word compared to the entity. We use the HR-BiLSTM model to predict the score of each relation $r \in R_e$: $s_{rel}(r; e, q)$.

5.3 Query Generation

Finally, the system outputs the <entity, relation (or core-chain)> pair $(\hat{e}, \hat{r})$ according to:

$$s(\hat{e}, \hat{r}; q) = \max_{e \in EL'_{K'}(q), r \in R_e} (\beta \cdot s_{\text{rerank}}(e; q) + (1 - \beta) \cdot s_{\text{rel}}(r; e, q)),$$

where β is a hyperparameter to be tuned.

5.4 Constraint Detection

Similar to (Yih et al., 2015), we adopt an additional constraint detection step based on text matching. Our method can be viewed as entity-linking on a KB sub-graph. It contains two steps: (1) **Sub-graph generation**: given the top scored query generated by the previous 3 steps[5], for each node v (answer node or the CVT node like in Figure 1(b)), we collect all the nodes c connecting to v (with relation r_c) with any relation, and generate a sub-graph associated to the original query. (2) **Entity-linking on sub-graph nodes**: we compute a matching score between each n-gram in the input question (without overlapping the topic entity) and entity name of c (except for the node in the original query) by taking into account the maximum overlapping sequence of characters between them (see Appendix A for details and B for special rules dealing with date/answer type constraints). If the matching score is larger than a threshold θ (tuned on training set), we will add the constraint entity c (and r_c) to the query by attaching it to the corresponding node v on the core-chain.

6 Experiments

6.1 Task Introduction & Settings

We use the SimpleQuestions (Bordes et al., 2015) and WebQSP (Yih et al., 2016) datasets. Each question in these datasets is labeled with the gold semantic parse. Hence we can directly evaluate relation detection performance independently as well as evaluate on the KBQA end task.

SimpleQuestions (SQ): It is a single-relation KBQA task. The KB we use consists of a Freebase subset with 2M entities (FB2M) (Bordes et al., 2015), in order to compare with previous research. Yin et al. (2016) also evaluated their relation extractor on this data set and released their proposed question-relation pairs, so we run our relation detection model on their data set. For the KBQA evaluation, we also start with their entity linking results[6]. Therefore, our results can be compared with their reported results on both tasks.

WebQSP (WQ): A multi-relation KBQA task. We use the entire Freebase KB for evaluation purposes. Following Yih et al. (2016), we use S-MART (Yang and Chang, 2015) entity-linking outputs.[7] In order to evaluate the relation detection models, we create a new relation detection task from the WebQSP data set.[8] For each question and its labeled semantic parse: (1) we first select the topic entity from the parse; and then (2) select all the relations and relation chains (length ≤ 2) connected to the topic entity, and set the core-chain labeled in the parse as the positive label and all the others as the negative examples.

We tune the following hyper-parameters on development sets: (1) the size of hidden states for LSTMs ($\{50, 100, 200, 400\}$)[9]; (2) learning rate ($\{0.1, 0.5, 1.0, 2.0\}$); (3) whether the shortcut connections are between hidden states or between max-pooling results (see Section 4.3); and (4) the number of training epochs.

For both the relation detection experiments and the second-step relation detection in KBQA, we have *entity replacement* first (see Section 5.2 and Figure 1). All word vectors are initialized with 300-d pretrained word embeddings (Mikolov et al., 2013). The embeddings of relation names are randomly initialized, since existing pre-trained relation embeddings (e.g. TransE) usually support limited sets of relation names. We leave the usage of pre-trained relation embeddings to future work.

6.2 Relation Detection Results

Table 2 shows the results on two relation detection tasks. The AMPCNN result is from (Yin et al., 2016), which yielded state-of-the-art scores by outperforming several attention-based meth-

[5]Starting with the top-1 query suffers more from error propagation. However we still achieve state-of-the-art on WebQSP in Sec.6, showing the advantage of our relation detection model. We leave in future work beam-search and feature extraction on beam for final answer re-ranking like in previous research.

[6]The two resources have been downloaded from https://github.com/Gorov/SimpleQuestions-EntityLinking

[7]https://github.com/scottyih/STAGG

[8]The dataset is available at https://github.com/Gorov/SimpleQuestions-EntityLinking.

[9]For CNNs we double the size for fair comparison.

Model	Relation Input Views	Accuracy	
		SimpleQuestions	WebQSP
AMPCNN (Yin et al., 2016)	words	91.3	-
BiCNN (Yih et al., 2015)	char-3-gram	90.0	77.74
BiLSTM w/ words	words	91.2	79.32
BiLSTM w/ relation names	rel_names	88.9	78.96
Hier-Res-BiLSTM (HR-BiLSTM)	words + rel_names	**93.3**	**82.53**
w/o rel_name	words	91.3	81.69
w/o rel_words	rel_names	88.8	79.68
w/o residual learning (weighted sum on two layers)	words + rel_names	92.5	80.65
replacing residual with attention (Parikh et al., 2016)	words + rel_names	92.6	81.38
single-layer BiLSTM question encoder	words + rel_names	92.8	78.41
replacing BiLSTM with CNN (HR-CNN)	words + rel_names	92.9	79.08

Table 2: Accuracy on the SimpleQuestions and WebQSP relation detection tasks (test sets). The top shows performance of baselines. On the bottom we give the results of our proposed model together with the ablation tests.

ods. We re-implemented the BiCNN model from (Yih et al., 2015), where both questions and relations are represented with the word hash trick on character tri-grams. The baseline BiLSTM with relation word sequence appears to be the best baseline on WebQSP and is close to the previous best result of AMPCNN on SimpleQuestions. Our proposed HR-BiLSTM outperformed the best baselines on both tasks by margins of 2-3% ($p < 0.001$ and 0.01 compared to the best baseline *BiLSTM w/ words* on SQ and WQ respectively).

Note that using only relation names instead of words results in a weaker baseline BiLSTM model. The model yields a significant performance drop on SimpleQuestions (91.2% to 88.9%). However, the drop is much smaller on WebQSP, and it suggests that unseen relations have a much bigger impact on SimpleQuestions.

Ablation Test: The bottom of Table 2 shows ablation results of the proposed HR-BiLSTM. First, hierarchical matching between questions and both relation names and relation words yields improvement on both datasets, especially for SimpleQuestions (93.3% vs. 91.2/88.8%). Second, residual learning helps hierarchical matching compared to weighted-sum and attention-based baselines (see Section 4.3). For the attention-based baseline, we tried the model from (Parikh et al., 2016) and its one-way variations, where the one-way model gives better results[10]. Note that residual learning significantly helps on WebQSP (80.65% to 82.53%), while it does not help as much on SimpleQuestions. On SimpleQuestions, even removing the deep layers only causes a small drop in performance. WebQSP benefits more from residual and deeper architecture, possibly because in this dataset it is more important to handle larger scope of context matching.

Finally, on WebQSP, replacing BiLSTM with CNN in our hierarchical matching framework results in a large performance drop. Yet on SimpleQuestions the gap is much smaller. We believe this is because the LSTM relation encoder can better learn the composition of chains of relations in WebQSP, as it is better at dealing with longer dependencies.

Analysis Next, we present empirical evidences, which show why our HR-BiLSTM model achieves the best scores. We use WebQSP for the analysis purposes. First, we have the hypothesis that *training of the weighted-sum model usually falls to local optima, since deep BiLSTMs do not guarantee that the two-levels of question hidden representations are comparable*. This is evidenced by that during training one layer usually gets a weight close to 0 thus is ignored. For example, one run gives us weights of -75.39/0.14 for the two layers (we take exponential for the final weighted sum). It also gives much lower training accuracy (91.94%) compared to HR-BiLSTM (95.67%), suffering from training difficulty.

Second, compared to our deep BiLSTM with shortcut connections, we have the hypothesis that for KB relation detection, *training deep BiLSTMs is more difficult without shortcut connections*. Our experiments suggest that deeper BiLSTM does not always result in lower training accuracy. In the experiments a two-layer BiLSTM converges to 94.99%, even lower than the 95.25% achieved by a

[10]We also tried to apply the same attention method on deep BiLSTM with residual connections, but it does not lead to better results compared to HR-BiLSTM. We hypothesize that the idea of hierarchical matching with attention mechanism may work better for long sequences, and the new advanced attention mechanisms (Wang and Jiang, 2016; Wang et al., 2017) might help hierarchical matching. We leave the above directions to future work.

single-layer BiLSTM. Under our setting the two-layer model captures the single-layer model as a special case (so it could potentially better fit the training data), this result suggests that the deep BiLSTM without shortcut connections might suffers more from training difficulty.

Finally, we hypothesize that *HR-BiLSTM is more than combination of two BiLSTMs with residual connections, because it encourages the hierarchical architecture to learn different levels of abstraction*. To verify this, we replace the deep BiLSTM question encoder with two single-layer BiLSTMs (both on words) with shortcut connections between their hidden states. This decreases test accuracy to 76.11%. It gives similar training accuracy compared to HR-BiLSTM, indicating a more serious over-fitting problem. This proves that the residual and deep structures both contribute to the good performance of HR-BiLSTM.

6.3 KBQA End-Task Results

Table 3 compares our system with two published baselines (1) STAGG (Yih et al., 2015), the state-of-the-art on WebQSP[11] and (2) AMPCNN (Yin et al., 2016), the state-of-the-art on SimpleQuestions. Since these two baselines are specially designed/tuned for one particular dataset, they do not generalize well when applied to the other dataset. In order to highlight the effect of different relation detection models on the KBQA end-task, we also implemented another baseline that uses our KBQA system but replaces HR-BiLSTM with our implementation of AMPCNN (for SimpleQuestions) or the char-3-gram BiCNN (for WebQSP) relation detectors (second block in Table 3).

Compared to the *baseline relation detector* (3rd row of results), our method, which includes an improved relation detector (HR-BiLSTM), improves the KBQA end task by 2-3% (4th row). Note that in contrast to previous KBQA systems, our system does not use joint-inference or feature-based re-ranking step, nevertheless it still achieves better or comparable results to the state-of-the-art.

The third block of the table details two ablation tests for the proposed components in our KBQA systems: (1) Removing the entity re-ranking step significantly decreases the scores. Since the re-ranking step relies on the relation detection models, this shows that our HR-BiLSTM model contributes to the good performance in multiple ways.

[11]The STAGG score on SQ is from (Bao et al., 2016).

System	Accuracy	
	SQ	**WQ**
STAGG	72.8	**63.9**
AMPCNN (Yin et al., 2016)	*76.4*	-
Baseline: Our Method w/ baseline relation detector	75.1	60.0
Our Method	**77.0**	63.0
w/o entity re-ranking	74.9	60.6
w/o constraints	-	58.0
Our Method (multi-detectors)	**78.7**	**63.9**

Table 3: KBQA results on SimpleQuestions (SQ) and WebQSP (WQ) test sets. The numbers in *green* color are directly comparable to our results since we start with the same entity linking results.

Appendix C gives the detailed performance of the re-ranking step. (2) In contrast to the conclusion in (Yih et al., 2015), constraint detection is crucial for our system[12]. This is probably because our joint performance on topic entity and core-chain detection is more accurate (77.5% top-1 accuracy), leaving a huge potential (77.5% vs. 58.0%) for the constraint detection module to improve.

Finally, like STAGG, which uses multiple relation detectors (see Yih et al. (2015) for the three models used), we also try to use the top-3 relation detectors from Section 6.2. As shown on the last row of Table 3, this gives a significant performance boost, resulting in a new state-of-the-art result on SimpleQuestions and a result comparable to the state-of-the-art on WebQSP.

7 Conclusion

KB relation detection is a key step in KBQA and is significantly different from general relation extraction tasks. We propose a novel KB relation detection model, HR-BiLSTM, that performs hierarchical matching between questions and KB relations. Our model outperforms the previous methods on KB relation detection tasks and allows our KBQA system to achieve state-of-the-arts. For future work, we will investigate the integration of our HR-BiLSTM into end-to-end systems. For example, our model could be integrated into the decoder in (Liang et al., 2016), to provide better sequence prediction. We will also investigate new emerging datasets like GraphQuestions (Su et al., 2016) and ComplexQuestions (Bao et al., 2016) to handle more characteristics of general QA.

[12]Note that another reason is that we are evaluating on accuracy here. When evaluating on F1 the gap will be smaller.

References

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 2503–2514.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, pages 1431–1440.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1533–1544.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075* .

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. pages 2787–2795.

Zihang Dai, Lei Li, and Wei Xu. 2016. Cfo: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 800–810.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 626–634.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL (1)*. Citeseer, pages 1608–1618.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. *arXiv preprint arXiv:1604.00727* .

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1774–1784.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 770–778.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 68–74.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2249–2255.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Uppsala, Sweden, pages 256–259.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 562–572. https://aclweb.org/anthology/D16-1054.

Ang Sun, Ralph Grishman, and Satoshi Sekine. 2011. Semi-supervised relation extraction with large-scale word clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 521–529.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 534–539.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1298–1307.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1442–1451. http://www.aclweb.org/anthology/N16-1170.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814* .

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2326–2336.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 504–513.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. 2014. Freebase qa: Information extraction or semantic parsing? *ACL 2014* page 82.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *ACL (1)*. Citeseer, pages 956–966.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Association for Computational Linguistics (ACL)*.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 643–648.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 201–206.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1746–1756.

Mo Yu, Mark Dredze, Raman Arora, and Matthew R. Gormley. 2016. Embedding lexical features via low-rank tensors. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1019–1029. http://www.aclweb.org/anthology/N16-1117.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 2335–2344.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Association for Computational Linguistics*. pages 427–434.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 207–212.

Deep Keyphrase Generation

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He*, Peter Brusilovsky, Yu Chi
School of Computing and Information
University of Pittsburgh
Pittsburgh, PA, 15213
{rui.meng, saz31, shh69, daqing, peterb, yuc73}@pitt.edu

Abstract

Keyphrase provides highly-summative information that can be effectively used for understanding, organizing and retrieving text content. Though previous studies have provided many workable solutions for automated keyphrase extraction, they commonly divided the to-be-summarized content into multiple text chunks, then ranked and selected the most meaningful ones. These approaches could neither identify keyphrases that do not appear in the text, nor capture the real semantic meaning behind the text. We propose a generative model for keyphrase prediction with an encoder-decoder framework, which can effectively overcome the above drawbacks. We name it as *deep keyphrase generation* since it attempts to capture the deep semantic meaning of the content with a deep learning method. Empirical analysis on six datasets demonstrates that our proposed model not only achieves a significant performance boost on extracting keyphrases that appear in the source text, but also can generate absent keyphrases based on the semantic meaning of the text. Code and dataset are available at https://github.com/memray/seq2seq-keyphrase.

1 Introduction

A keyphrase or keyword is a piece of short, summative content that expresses the main semantic meaning of a longer text. The typical use of a keyphrase or keyword is in scientific publications to provide the core information of a paper. We use the term "keyphrase" interchangeably with "keyword" in the rest of this paper, as both terms have an implication that they may contain multiple words. High-quality keyphrases can facilitate the understanding, organizing, and accessing of document content. As a result, many studies have focused on ways of automatically extracting keyphrases from textual content (Liu et al., 2009; Medelyan et al., 2009a; Witten et al., 1999). Due to public accessibility, many scientific publication datasets are often used as test beds for keyphrase extraction algorithms. Therefore, this study also focuses on extracting keyphrases from scientific publications.

Automatically extracting keyphrases from a document is called *keypharase extraction*, and it has been widely used in many applications, such as information retrieval (Jones and Staveley, 1999), text summarization (Zhang et al., 2004), text categorization (Hulth and Megyesi, 2006), and opinion mining (Berend, 2011). Most of the existing keyphrase extraction algorithms have addressed this problem through two steps (Liu et al., 2009; Tomokiyo and Hurst, 2003). The first step is to acquire a list of keyphrase candidates. Researchers have tried to use n-grams or noun phrases with certain part-of-speech patterns for identifying potential candidates (Hulth, 2003; Le et al., 2016; Liu et al., 2010; Wang et al., 2016). The second step is to rank candidates on their importance to the document, either through supervised or unsupervised machine learning methods with a set of manually-defined features (Frank et al., 1999; Liu et al., 2009, 2010; Kelleher and Luz, 2005; Matsuo and Ishizuka, 2004; Mihalcea and Tarau, 2004; Song et al., 2003; Witten et al., 1999).

There are two major drawbacks in the above keyphrase extraction approaches. First, these methods can only extract the keyphrases that ap-

*Corresponding author

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 582–592
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1054

pear in the source text; they fail at predicting meaningful keyphrases with a slightly different sequential order or those that use synonyms. However, authors of scientific publications commonly assign keyphrases based on their semantic meaning, instead of following the written content in the publication. In this paper, we denote phrases that do not match any contiguous subsequence of source text as **absent keyphrases**, and the ones that fully match a part of the text as **present keyphrases**. Table 1 shows the proportion of present and absent keyphrases from the document abstract in four commonly-used datasets, from which we can observe large portions of absent keyphrases in all the datasets. The absent keyphrases cannot be extracted through previous approaches, which further prompts the development of a more powerful keyphrase prediction model.

Second, when ranking phrase candidates, previous approaches often adopted machine learning features such as TF-IDF and PageRank. However, these features only target to detect the importance of each word in the document based on the statistics of word occurrence and co-occurrence, and are unable to reveal the full semantics that underlie the document content.

Table 1: Proportion of the present keyphrases and absent keyphrases in four public datasets

Dataset	# Keyphrase	% Present	% Absent
Inspec	19,275	55.69	44.31
Krapivin	2,461	44.74	52.26
NUS	2,834	67.75	32.25
SemEval	12,296	42.01	57.99

To overcome the limitations of previous studies, we re-examine the process of *keyphrase prediction* with a focus on how real human annotators would assign keyphrases. Given a document, human annotators will first read the text to get a basic understanding of the content, then they try to digest its essential content and summarize it into keyphrases. Their generation of keyphrases relies on an understanding of the content, which may not necessarily use the exact words that occur in the source text. For example, when human annotators see "*Latent Dirichlet Allocation*" in the text, they might write down "*topic modeling*" and/or "*text mining*" as possible keyphrases. In addition to the semantic understanding, human annotators might also go back and pick up the most important parts, based on syntactic features. For example, the phrases following "*we propose/apply/use*" could be important in the text. As a result, a better keyphrase prediction model should understand the semantic meaning of the content, as well as capture the contextual features.

To effectively capture both the semantic and syntactic features, we use recurrent neural networks (RNN) (Cho et al., 2014; Gers and Schmidhuber, 2001) to compress the semantic information in the given text into a dense vector (i.e., semantic understanding). Furthermore, we incorporate a copying mechanism (Gu et al., 2016) to allow our model to find important parts based on positional information. Thus, our model can generate keyphrases based on an understanding of the text, regardless of the presence or absence of keyphrases in the text; at the same time, it does not lose important in-text information.

The contribution of this paper is three-fold. First, we propose to apply an RNN-based generative model to keyphrase prediction, as well as incorporate a copying mechanism in RNN, which enables the model to successfully predict phrases that rarely occur. Second, this is the first work that concerns the problem of absent keyphrase prediction for scientific publications, and our model recalls up to 20% of absent keyphrases. Third, we conducted a comprehensive comparison against six important baselines on a broad range of datasets, and the results show that our proposed model significantly outperforms existing supervised and unsupervised extraction methods.

In the remainder of this paper, we first review the related work in Section 2. Then, we elaborate upon the proposed model in Section 3. After that, we present the experiment setting in Section 4 and results in Section 5, followed by our discussion in Section 6. Section 7 concludes the paper.

2 Related Work

2.1 Automatic Keyphrase Extraction

A keyphrase provides a succinct and accurate way of describing a subject or a subtopic in a document. A number of extraction algorithms have been proposed, and the process of extracting keyphrases can typically be broken down into two steps.

The first step is to generate a list of phrase can-

didates with heuristic methods. As these candidates are prepared for further filtering, a considerable number of candidates are produced in this step to increase the possibility that most of the correct keyphrases are kept. The primary ways of extracting candidates include retaining word sequences that match certain part-of-speech tag patterns (e.g., nouns, adjectives) (Liu et al., 2011; Wang et al., 2016; Le et al., 2016), and extracting important n-grams or noun phrases (Hulth, 2003; Medelyan et al., 2008).

The second step is to score each candidate phrase for its likelihood of being a keyphrase in the given document. The top-ranked candidates are returned as keyphrases. Both supervised and unsupervised machine learning methods are widely employed here. For supervised methods, this task is solved as a binary classification problem, and various types of learning methods and features have been explored (Frank et al., 1999; Witten et al., 1999; Hulth, 2003; Medelyan et al., 2009b; Lopez and Romary, 2010; Gollapalli and Caragea, 2014). As for unsupervised approaches, primary ideas include finding the central nodes in text graph (Mihalcea and Tarau, 2004; Grineva et al., 2009), detecting representative phrases from topical clusters (Liu et al., 2009, 2010), and so on.

Aside from the commonly adopted two-step process, another two previous studies realized the keyphrase extraction in entirely different ways. Tomokiyo and Hurst (2003) applied two language models to measure the phraseness and informativeness of phrases. Liu et al. (2011) share the most similar ideas to our work. They used a word alignment model, which learns a translation from the documents to the keyphrases. This approach alleviates the problem of vocabulary gaps between source and target to a certain degree. However, this translation model is unable to handle semantic meaning. Additionally, this model was trained with the target of title/summary to enlarge the number of training samples, which may diverge from the real objective of generating keyphrases.

Zhang et al. (2016) proposed a joint-layer recurrent neural network model to extract keyphrases from tweets, which is another application of deep neural networks in the context of keyphrase extraction. However, their work focused on sequence labeling, and is therefore not able to predict absent keyphrases.

2.2 Encoder-Decoder Model

The RNN Encoder-Decoder model (which is also referred as sequence-to-sequence Learning) is an end-to-end approach. It was first introduced by Cho et al. (2014) and Sutskever et al. (2014) to solve translation problems. As it provides a powerful tool for modeling variable-length sequences in an end-to-end fashion, it fits many natural language processing tasks and can rapidly achieve great successes (Rush et al., 2015; Vinyals et al., 2015; Serban et al., 2016).

Different strategies have been explored to improve the performance of the Encoder-Decoder model. The attention mechanism (Bahdanau et al., 2014) is a soft alignment approach that allows the model to automatically locate the relevant input components. In order to make use of the important information in the source text, some studies sought ways to copy certain parts of content from the source text and paste them into the target text (Allamanis et al., 2016; Gu et al., 2016; Zeng et al., 2016). A discrepancy exists between the optimizing objective during training and the metrics during evaluation. A few studies attempted to eliminate this discrepancy by incorporating new training algorithms (Marc'Aurelio Ranzato et al., 2016) or by modifying the optimizing objectives (Shen et al., 2016).

3 Methodology

This section will introduce our proposed deep keyphrase generation method in detail. First, the task of keyphrase generation is defined, followed by an overview of how we apply the RNN Encoder-Decoder model. Details of the framework as well as the copying mechanism will be introduced in Sections 3.3 and 3.4.

3.1 Problem Definition

Given a keyphrase dataset that consists of $\mathbf{N}$ data samples, the i-th data sample $(\mathbf{x^{(i)}}, \mathbf{p^{(i)}})$ contains one source text $\mathbf{x^{(i)}}$, and M_i target keyphrases $\mathbf{p^{(i)}} = (\mathbf{p^{(i,1)}}, \mathbf{p^{(i,2)}}, \ldots, \mathbf{p^{(i,M_i)}})$. Both the source text $\mathbf{x^{(i)}}$ and keyphrase $\mathbf{p^{(i,j)}}$ are sequences of words:

$$\mathbf{x^{(i)}} = x_1^{(i)}, x_2^{(i)}, \ldots, x_{L_{\mathbf{x^i}}}^{(i)}$$

$$\mathbf{p^{(i,j)}} = y_1^{(i,j)}, y_2^{(i,j)}, \ldots, y_{L_{\mathbf{p^{(i,j)}}}}^{(i,j)}$$

$L_{\mathbf{x^{(i)}}}$ and $L_{\mathbf{p^{(i,j)}}}$ denotes the length of word sequence of $\mathbf{x^{(i)}}$ and $\mathbf{p^{(i,j)}}$ respectively.

Each data sample contains one source text sequence and multiple target phrase sequences. To apply the RNN Encoder-Decoder model, the data need to be converted into text-keyphrase pairs that contain only one source sequence and one target sequence. We adopt a simple way, which splits the data sample $(\mathbf{x^{(i)}}, \mathbf{p^{(i)}})$ into M_i pairs: $(\mathbf{x^{(i)}}, \mathbf{p^{(i,1)}}), (\mathbf{x^{(i)}}, \mathbf{p^{(i,2)}}), \ldots, (\mathbf{x^{(i)}}, \mathbf{p^{(i,M_i)}})$. Then the Encoder-Decoder model is ready to be applied to learn the mapping from the source sequence to target sequence. For the purpose of simplicity, $(\mathbf{x}, \mathbf{y})$ is used to denote each data pair in the rest of this section, where $\mathbf{x}$ is the word sequence of a source text and $\mathbf{y}$ is the word sequence of its keyphrase.

3.2 Encoder-Decoder Model

The basic idea of our keyphrase generation model is to compress the content of source text into a hidden representation with an encoder and to generate corresponding keyphrases with the decoder, based on the representation . Both the encoder and decoder are implemented with recurrent neural networks (RNN).

The encoder RNN converts the variable-length input sequence $\mathbf{x} = (x_1, x_2, ..., x_T)$ into a set of hidden representation $\mathbf{h} = (h_1, h_2, \ldots, h_T)$, by iterating the following equations along time t:

$$\mathbf{h}_t = f(x_t, \mathbf{h}_{t-1}) \tag{1}$$

where f is a non-linear function. We get the context vector $\mathbf{c}$ acting as the representation of the whole input $\mathbf{x}$ through a non-linear function q.

$$\mathbf{c} = q(h_1, h_2, ..., h_T) \tag{2}$$

The decoder is another RNN; it decompresses the context vector and generates a variable-length sequence $\mathbf{y} = (y_1, y_2, ..., y_{T'})$ word by word, through a conditional language model:

$$\begin{aligned} \mathbf{s}_t &= f(y_{t-1}, \mathbf{s}_{t-1}, \mathbf{c}) \\ p(y_t | y_{1,\ldots,t-1}, \mathbf{x}) &= g(y_{t-1}, \mathbf{s}_t, \mathbf{c}) \end{aligned} \tag{3}$$

where $\mathbf{s}_t$ is the hidden state of the decoder RNN at time t. The non-linear function g is a softmax classifier, which outputs the probabilities of all the words in the vocabulary. y_t is the predicted word at time t, by taking the word with largest probability after $g(\cdot)$.

The encoder and decoder networks are trained jointly to maximize the conditional probability of the target sequence, given a source sequence. After training, we use the beam search to generate phrases and a max heap is maintained to get the predicted word sequences with the highest probabilities.

3.3 Details of the Encoder and Decoder

A bidirectional gated recurrent unit (GRU) is applied as our encoder to replace the simple recurrent neural network. Previous studies (Bahdanau et al., 2014; Cho et al., 2014) indicate that it can generally provide better performance of language modeling than a simple RNN and a simpler structure than other Long Short-Term Memory networks (Hochreiter and Schmidhuber, 1997). As a result, the above non-linear function f is replaced by the **GRU** function (see in (Cho et al., 2014)).

Another forward GRU is used as the decoder. In addition, an attention mechanism is adopted to improve performance. The attention mechanism was firstly introduced by Bahdanau et al. (2014) to make the model dynamically focus on the important parts in input. The context vector $\mathbf{c}$ is computed as a weighted sum of hidden representation $\mathbf{h} = (h_1, \ldots, h_T)$:

$$\begin{aligned} \mathbf{c}_i &= \sum_{j=1}^{T} \alpha_{ij} h_j \\ \alpha_{ij} &= \frac{\exp(a(s_{i-1}, h_j))}{\sum_{k=1}^{T} \exp(a(s_{i-1}, h_k))} \end{aligned} \tag{4}$$

where $a(s_{i-1}, h_j)$ is a soft alignment function that measures the similarity between s_{i-1} and h_j; namely, to which degree the inputs around position j and the output at position i match.

3.4 Copying Mechanism

To ensure the quality of learned representation and reduce the size of the vocabulary, typically the RNN model considers a certain number of frequent words (e.g. 30,000 words in (Cho et al., 2014)), but a large amount of long-tail words are simply ignored. Therefore, the RNN is not able to recall any keyphrase that contains out-of-vocabulary words. Actually, important phrases can also be identified by positional and syntactic information in their contexts, even though their exact meanings are not known. The copying mechanism (Gu et al., 2016) is one feasible solution that enables RNN to predict out-of-vocabulary words by selecting appropriate words from the source text.

By incorporating the copying mechanism, the probability of predicting each new word y_t consists of two parts. The first term is the probability of generating the term (see Equation 3) and the second one is the probability of copying it from the source text:

$$\begin{aligned} &p(y_t|y_{1,\ldots,t-1}, \mathbf{x}) \\ &= p_g(y_t|y_{1,\ldots,t-1}, \mathbf{x}) + p_c(y_t|y_{1,\ldots,t-1}, \mathbf{x}) \end{aligned} \tag{5}$$

Similar to attention mechanism, the copying mechanism weights the importance of each word in source text with a measure of positional attention. But unlike the generative RNN which predicts the next word from all the words in vocabulary, the copying part $p_c(y_t|y_{1,\ldots,t-1}, \mathbf{x})$ only considers the words in source text. Consequently, on the one hand, the RNN with copying mechanism is able to predict the words that are out of vocabulary but in the source text; on the other hand, the model would potentially give preference to the appearing words, which caters to the fact that most keyphrases tend to appear in the source text.

$$\begin{aligned} p_c(y_t|y_{1,\ldots,t-1}, \mathbf{x}) &= \frac{1}{Z} \sum_{j:x_j=y_t} \exp(\psi_c(x_j)), y \in \chi \\ \psi_c(x_j) &= \sigma(\mathbf{h}_j^T \mathbf{W}_c)\mathbf{s}_t \end{aligned} \tag{6}$$

where χ is the set of all of the unique words in the source text $\mathbf{x}$, σ is a non-linear function and $\mathbf{W_c} \in \mathbb{R}$ is a learned parameter matrix. $\mathbf{Z}$ is the sum of all the scores and is used for normalization. Please see (Gu et al., 2016) for more details.

4 Experiment Settings

This section begins by discussing how we designed our evaluation experiments, followed by the description of training and testing datasets. Then, we introduce our evaluation metrics and baselines.

4.1 Training Dataset

There are several publicly-available datasets for evaluating keyphrase generation. The largest one came from Krapivin et al. (2008), which contains 2,304 scientific publications. However, this amount of data is unable to train a robust recurrent neural network model. In fact, there are millions of scientific papers available online, each of which contains the keyphrases that were assigned by their authors. Therefore, we collected a large amount of high-quality scientific metadata in the computer science domain from various online digital libraries, including ACM Digital Library, ScienceDirect, Wiley, and Web of Science etc. (Han et al., 2013; Rui et al., 2016). In total, we obtained a dataset of 567,830 articles, after removing duplicates and overlaps with testing datasets, which is 200 times larger than the one of Krapivin et al. (2008). Note that our model is only trained on 527,830 articles, since 40,000 publications are randomly held out, among which 20,000 articles were used for building a new test dataset **KP20k**. Another 20,000 articles served as the validation dataset to check the convergence of our model, as well as the training dataset for supervised baselines.

4.2 Testing Datasets

For evaluating the proposed model more comprehensively, four widely-adopted scientific publication datasets were used. In addition, since these datasets only contain a few hundred or a few thousand publications, we contribute a new testing dataset **KP20k** with a much larger number of scientific articles. We take the title and abstract as the source text. Each dataset is described in detail below.

- **Inspec** (Hulth, 2003): This dataset provides 2,000 paper abstracts. We adopt the 500 testing papers and their corresponding uncontrolled keyphrases for evaluation, and the remaining 1,500 papers are used for training the supervised baseline models.

- **Krapivin** (Krapivin et al., 2008): This dataset provides 2,304 papers with full-text and author-assigned keyphrases. However, the author did not mention how to split testing data, so we selected the first 400 papers in alphabetical order as the testing data, and the remaining papers are used to train the supervised baselines.

- **NUS** (Nguyen and Kan, 2007): We use the author-assigned keyphrases and treat all 211 papers as the testing data. Since the NUS dataset did not specifically mention the ways of splitting training and testing data, the results of the supervised baseline models are obtained through a five-fold cross-validation.

- **SemEval-2010** (Kim et al., 2010): 288 articles were collected from the ACM Digital

Library. 100 articles were used for testing and the rest were used for training supervised baselines.

– **KP20k**: We built a new testing dataset that contains the titles, abstracts, and keyphrases of 20,000 scientific articles in computer science. They were randomly selected from our obtained 567,830 articles. Due to the memory limits of implementation, we were not able to train the supervised baselines on the whole training set. Thus we take the 20,000 articles in the validation set to train the supervised baselines. It is worth noting that we also examined their performance by enlarging the training dataset to 50,000 articles, but no significant improvement was observed.

4.3 Implementation Details

In total, there are 2,780,316 ⟨text, keyphrase⟩ pairs for training, in which text refers to the concatenation of the title and abstract of a publication, and keyphrase indicates an author-assigned keyword. The text pre-processing steps including tokenization, lowercasing and replacing all digits with symbol ⟨digit⟩ are applied. Two encoder-decoder models are trained, one with only attention mechanism (RNN) and one with both attention and copying mechanism enabled (CopyRNN). For both models, we choose the top 50,000 frequently-occurred words as our vocabulary, the dimension of embedding is set to 150, the dimension of hidden layers is set to 300, and the word embeddings are randomly initialized with uniform distribution in [-0.1,0.1]. Models are optimized using Adam (Kingma and Ba, 2014) with initial learning rate = 10^{-4}, gradient clipping = 0.1 and dropout rate = 0.5. The max depth of beam search is set to 6, and the beam size is set to 200. The training is stopped once convergence is determined on the validation dataset (namely early-stopping, the cross-entropy loss stops dropping for several iterations).

In the generation of keyphrases, we find that the model tends to assign higher probabilities for shorter keyphrases, whereas most keyphrases contain more than two words. To resolve this problem, we apply a simple heuristic by preserving only the first single-word phrase (with the highest generating probability) and removing the rest.

4.4 Baseline Models

Four unsupervised algorithms (Tf-Idf, TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), and ExpandRank (Wan and Xiao, 2008)) and two supervised algorithms (KEA (Witten et al., 1999) and Maui (Medelyan et al., 2009a)) are adopted as baselines. We set up the four unsupervised methods following the optimal settings in (Hasan and Ng, 2010), and the two supervised methods following the default setting as specified in their papers.

4.5 Evaluation Metric

Three evaluation metrics, the macro-averaged *precision*, *recall* and *F-measure* (F_1) are employed for measuring the algorithm's performance. Following the standard definition, precision is defined as the number of correctly-predicted keyphrases over the number of all predicted keyphrases, and recall is computed by the number of correctly-predicted keyphrases over the total number of data records. Note that, when determining the match of two keyphrases, we use Porter Stemmer for preprocessing.

5 Results and Analysis

We conduct an empirical study on three different tasks to evaluate our model.

5.1 Predicting Present Keyphrases

This is the same as the keyphrase extraction task in prior studies, in which we analyze how well our proposed model performs on a commonly-defined task. To make a fair comparison, we only consider the present keyphrases for evaluation in this task. Table 2 provides the performances of the six baseline models, as well as our proposed models (i.e., RNN and CopyRNN). For each method, the table lists its F-measure at top 5 and top 10 predictions on the five datasets. The best scores are highlighted in bold and the underlines indicate the second best performances.

The results show that the four unsupervised models (Tf-idf, TextTank, SingleRank and ExpandRank) have a robust performance across different datasets. The ExpandRank fails to return any result on the KP20k dataset, due to its high time complexity. The measures on NUS and SemEval here are higher than the ones reported in (Hasan and Ng, 2010) and (Kim et al., 2010), probably because we utilized the paper abstract instead of the full text for training, which may

Method	Inspec		Krapivin		NUS		SemEval		KP20k	
	F_1@5	F_1@10	F_1@5	F_1@10	F_1@5	F_1@10	F_1@5	F_1@10	F_1@5	F_1@10
Tf-Idf	0.221	0.313	0.129	0.160	0.136	0.184	0.128	0.194	0.102	0.126
TextRank	0.223	0.281	0.189	0.162	0.195	0.196	0.176	0.187	0.175	0.147
SingleRank	0.214	0.306	0.189	0.162	0.140	0.173	0.135	0.176	0.096	0.119
ExpandRank	0.210	0.304	0.081	0.126	0.132	0.164	0.139	0.170	N/A	N/A
Maui	0.040	0.042	0.249	0.216	0.249	0.268	0.044	0.039	0.270	0.230
KEA	0.098	0.126	0.110	0.152	0.069	0.084	0.025	0.026	0.171	0.154
RNN	0.085	0.064	0.135	0.088	0.169	0.127	0.157	0.124	0.179	0.189
CopyRNN	**0.278**	**0.342**	**0.311**	**0.266**	**0.334**	**0.326**	**0.293**	**0.304**	**0.333**	**0.262**

Table 2: The performance of predicting present keyphrases of various models on five benchmark datasets

filter out some noisy information. The performance of the two supervised models (i.e., Maui and KEA) were unstable on some datasets, but Maui achieved the best performances on three datasets among all the baseline models.

As for our proposed keyphrase prediction approaches, the RNN model with the attention mechanism did not perform as well as we expected. It might be because the RNN model is only concerned with finding the hidden semantics behind the text, which may tend to generate keyphrases or words that are too general and may not necessarily refer to the source text. In addition, we observe that 2.5% (70,891/2,780,316) of keyphrases in our dataset contain out-of-vocabulary words, which the RNN model is not able to recall, since the RNN model can only generate results with the 50,000 words in vocabulary. This indicates that a pure generative model may not fit the extraction task, and we need to further link back to the language usage within the source text. The CopyRNN model, by considering more contextual information, significantly outperforms not only the RNN model but also all baselines, exceeding the best baselines by more than 20% on average. This result demonstrates the importance of source text to the extraction task. Besides, nearly 2% of all correct predictions contained out-of-vocabulary words.

The example in Figure 1(a) shows the result of predicted present keyphrases by RNN and CopyRNN for an article about video search. We see that both models can generate phrases that relate to the topic of information retrieval and video. However most of RNN predictions are high-level terminologies, which are too general to be selected as keyphrases. CopyRNN, on the other hand, predicts more detailed phrases like "video metadata" and "integrated ranking". An interesting bad case, "rich content" coordinates with a keyphrase "video metadata", and the CopyRNN mistakenly puts it into prediction.

5.2 Predicting Absent Keyphrases

As stated, one important motivation for this work is that we are interested in the proposed model's capability for predicting absent keyphrases based on the "understanding" of content. It is worth noting that such prediction is a very challenging task, and, to the best of our knowledge, no existing methods can handle this task. Therefore, we only provide the RNN and CopyRNN performances in the discussion of the results of this task. Here, we evaluate the performance within the recall of the top 10 and top 50 results, to see how many absent keyphrases can be correctly predicted. We use the absent keyphrases in the testing datasets for evaluation.

Dataset	RNN		CopyRNN	
	R@10	R@50	R@10	R@50
Inspec	0.031	0.061	**0.047**	**0.100**
Krapivin	0.095	0.156	**0.113**	**0.202**
NUS	0.050	0.089	**0.058**	**0.116**
SemEval	0.041	0.060	**0.043**	**0.067**
KP20k	0.083	0.144	**0.125**	**0.211**

Table 3: Absent keyphrases prediction performance of RNN and CopyRNN on five datasets

Table 3 presents the recall results of the top 10/50 predicted keyphrases for our RNN and CopyRNN models, in which we observe that the CopyRNN can, on average, recall around 8%

(15%) of keyphrases at top 10 (50) predictions. This indicates that, to some extent, both models can capture the hidden semantics behind the textual content and make reasonable predictions. In addition, with the advantage of features from the source text, the CopyRNN model also outperforms the RNN model in this condition, though it does not show as much improvement as the present keyphrase extraction task. An example is shown in Figure 1(b), in which we see that two absent keyphrases, "video retrieval" and "video indexing", are correctly recalled by both models. Note that the term "indexing" does not appear in the text, but the models may detect the information "index videos" in the first sentence and paraphrase it to the target phrase. And the CopyRNN successfully predicts another two keyphrases by capturing the detailed information from the text (highlighted text segments).

Model	F_1	Model	F_1
Tf-Idf	0.270	**ExpandRank**	0.269
TextRank	0.097	**KeyCluster**	0.140
SingleRank	0.256	**CopyRNN**	0.164

Table 4: Keyphrase prediction performance of CopyRNN on DUC-2001. The model is trained on scientific publication and evaluated on news.

5.3 Transferring the Model to the News Domain

RNN and CopyRNN are supervised models, and they are trained on data in a specific domain and writing style. However, with sufficient training on a large-scale dataset, we expect the models to be able to learn universal language features that are also effective in other corpora. Thus in this task, we will test our model on another type of text, to see whether the model would work when being transferred to a different environment.

We use the popular news article dataset **DUC-2001** (Wan and Xiao, 2008) for analysis. The dataset consists of 308 news articles and 2,488 manually annotated keyphrases. The result of this analysis is shown in Table 4, from which we could see that the CopyRNN can extract a portion of correct keyphrases from a unfamiliar text. Compared to the results reported in (Hasan and Ng, 2010), the performance of CopyRNN is better than TextRank (Mihalcea and Tarau, 2004) and KeyCluster (Liu et al., 2009), but lags behind the other three baselines.

As it is transferred to a corpus in a completely different type and domain, the model encounters more unknown words and has to rely more on the positional and syntactic features within the text. In this experiment, the CopyRNN recalls 766 keyphrases. 14.3% of them contain out-of-vocabulary words, and many names of persons and places are correctly predicted.

6 Discussion

Our experimental results demonstrate that the CopyRNN model not only performs well on predicting present keyphrases, but also has the ability to generate topically relevant keyphrases that are absent in the text. In a broader sense, this model attempts to map a long text (i.e., paper abstract) with representative short text chunks (i.e., keyphrases), which can potentially be applied to improve information retrieval performance by generating high-quality index terms, as well as assisting user browsing by summarizing long documents into short, readable phrases.

Thus far, we have tested our model with scientific publications and news articles, and have demonstrated that our model has the ability to capture universal language patterns and extract key information from unfamiliar texts. We believe that our model has a greater potential to be generalized to other domains and types, like books, online reviews, etc., if it is trained on a larger data corpus. Also, we directly applied our model, which was trained on a publication dataset, into generating keyphrases for news articles without any adaptive training. We believe that with proper training on news data, the model would make further improvement.

Additionally, this work mainly studies the problem of discovering core content from textual materials. Here, the encoder-decoder framework is applied to model language; however, such a framework can also be extended to locate the core information on other data resources, such as summarizing content from images and videos.

7 Conclusions and Future Work

In this paper, we proposed an RNN-based generative model for predicting keyphrases in scientific text. To the best of our knowledge, this is the first application of the encoder-decoder model to a keyphrase prediction task. Our model summarizes phrases based the deep semantic meaning

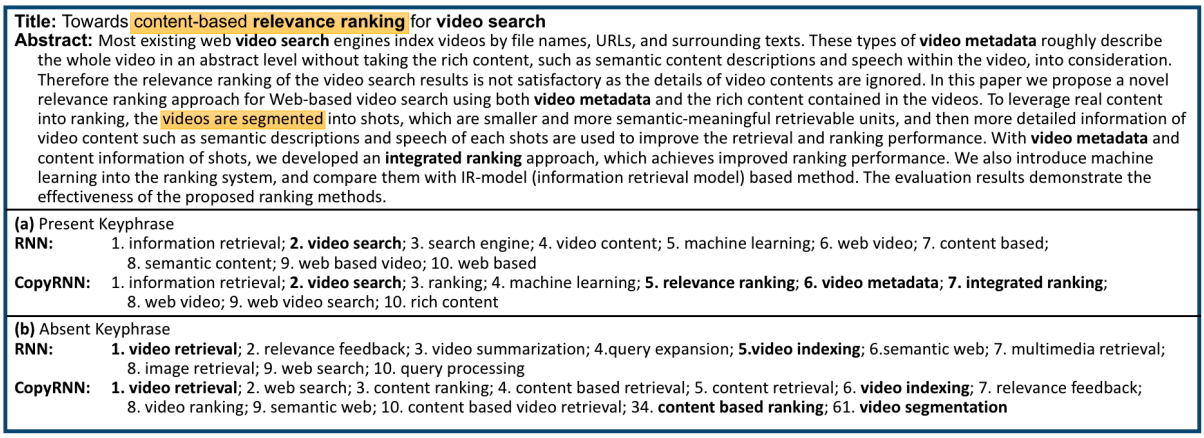


Figure 1: An example of predicted keyphrase by RNN and CopyRNN. Phrases shown in bold are correct predictions.

of the text, and is able to handle rarely-occurred phrases by incorporating a copying mechanism. Comprehensive empirical studies demonstrate the effectiveness of our proposed model for generating both present and absent keyphrases for different types of text. Our future work may include the following two directions.

– In this work, we only evaluated the performance of the proposed model by conducting off-line experiments. In the future, we are interested in comparing the model to human annotators and using human judges to evaluate the quality of predicted phrases.

– Our current model does not fully consider correlation among target keyphrases. It would also be interesting to explore the multiple-output optimization aspects of our model.

Acknowledgments

We would like to thank Jiatao Gu and Miltiadis Allamanis for sharing the source code and giving helpful advice. We also thank Wei Lu, Yong Huang, Qikai Cheng and other IRLAB members at Wuhan University for the assistance of dataset development. This work is partially supported by the National Science Foundation under Grant No.1525186.

References

M. Allamanis, H. Peng, and C. Sutton. 2016. A Convolutional Attention Network for Extreme Summarization of Source Code. *ArXiv e-prints* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Gábor Berend. 2011. Opinion expression mining by exploiting keyphrase extraction. In *IJCNLP*. Citeseer, pages 1162–1170.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-specific keyphrase extraction .

Felix A Gers and E Schmidhuber. 2001. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks* 12(6):1333–1340.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'14, pages 1629–1635. http://dl.acm.org/citation.cfm?id=2892753.2892779.

Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th International Conference on World Wide Web*. ACM, New York, NY, USA, WWW '09, pages 661–670. https://doi.org/10.1145/1526709.1526798.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* .

Shuguang Han, Daqing He, Jiepu Jiang, and Zhen Yue. 2013. Supporting exploratory people search: a study of factor transparency and user control. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, pages 449–458.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 365–373.

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

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*. Association for Computational Linguistics, pages 216–223.

Anette Hulth and Beáta B Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 537–544.

Steve Jones and Mark S Staveley. 1999. Phrasier: a system for interactive document retrieval using keyphrases. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 160–167.

Daniel Kelleher and Saturnino Luz. 2005. Automatic hypertext keyphrase detection. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'05, pages 1608–1609. http://dl.acm.org/citation.cfm?id=1642293.1642576.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 21–26.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Mikalai Krapivin, Aliaksandr Autayeu, and Maurizio Marchese. 2008. Large dataset for keyphrases extraction. Technical Report DISI-09-055, DISI, Trento, Italy.

Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. 2016. *Unsupervised Keyphrase Extraction: Introducing New Kinds of Words to Keyphrases*, Springer International Publishing, Cham, pages 665–671.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 135–144.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 366–376.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, pages 257–266.

Patrice Lopez and Laurent Romary. 2010. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Stroudsburg, PA, USA, SemEval '10, pages 248–251. http://dl.acm.org/citation.cfm?id=1859664.1859719.

Sumit Chopra Marc'Aurelio Ranzato, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR, San Juan, Puerto Rico* .

Yutaka Matsuo and Mitsuru Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools* 13(01):157–169.

Olena Medelyan, Eibe Frank, and Ian H Witten. 2009a. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*. Association for Computational Linguistics, pages 1318–1327.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009b. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '09, pages 1318–1327. http://dl.acm.org/citation.cfm?id=1699648.1699678.

Olena Medelyan, Ian H Witten, and David Milne. 2008. Topic indexing with wikipedia. In *Proceedings of the AAAI WikiAI workshop*. volume 1, pages 19–24.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. Association for Computational Linguistics.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *International Conference on Asian Digital Libraries*. Springer, pages 317–326.

Meng Rui, Han Shuguang, Huang Yun, He Daqing, and Brusilovsky Peter. 2016. Knowledge-based content linking for online textbooks. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence*. The Institute of Electrical and Electronics Engineers, pages 18–25.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 379–389. http://aclweb.org/anthology/D/D15/D15-1044.pdf.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1683–1692. http://www.aclweb.org/anthology/P16-1159.

Min Song, Il-Yeol Song, and Xiaohua Hu. 2003. Kpspotter: a flexible information gain-based keyphrase extraction system. In *Proceedings of the 5th ACM international workshop on Web information and data management*. ACM, pages 50–53.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Takashi Tomokiyo and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment - Volume 18*. Association for Computational Linguistics, Stroudsburg, PA, USA, MWE '03, pages 33–40. https://doi.org/10.3115/1119282.1119287.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge.

Minmei Wang, Bo Zhao, and Yihua Huang. 2016. *PTR: Phrase-Based Topical Ranking for Automatic Keyphrase Extraction in Scientific Publications*, Springer International Publishing, Cham, pages 120–128.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*. ACM, pages 254–255.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382* .

Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 836–845. https://aclweb.org/anthology/D16-1080.

Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. 2004. World wide web site summarization. *Web Intelligence and Agent Systems: An International Journal* 2(1):39–53.

Attention-over-Attention Neural Networks for Reading Comprehension

Yiming Cui[†], **Zhipeng Chen**[†], **Si Wei**[†], **Shijin Wang**[†], **Ting Liu**[‡] and **Guoping Hu**[†]
[†]Joint Laboratory of HIT and iFLYTEK, iFLYTEK Research, Beijing, China
[‡]Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology, Harbin, China
[†]{ymcui,zpchen,siwei,sjwang3,gphu}@iflytek.com
[‡]tliu@ir.hit.edu.cn

Abstract

Cloze-style reading comprehension is a representative problem in mining relationship between document and query. In this paper, we present a simple but novel model called *attention-over-attention* reader for better solving cloze-style reading comprehension task. The proposed model aims to place another attention mechanism over the document-level attention and induces "attended attention" for final answer predictions. One advantage of our model is that it is simpler than related works while giving excellent performance. In addition to the primary model, we also propose an N-best re-ranking strategy to double check the validity of the candidates and further improve the performance. Experimental results show that the proposed methods significantly outperform various state-of-the-art systems by a large margin in public datasets, such as CNN and Children's Book Test.

1 Introduction

To read and comprehend the human languages are challenging tasks for the machines, which requires that the understanding of natural languages and the ability to do reasoning over various clues. Reading comprehension is a general problem in the real world, which aims to read and comprehend a given article or context, and answer the questions based on it. Recently, the cloze-style reading comprehension problem has become a popular task in the community. The cloze-style query (Taylor, 1953) is a problem that to fill in an appropriate word in the given sentences while taking the context information into account.

To teach the machine to do cloze-style reading comprehensions, large-scale training data is necessary for learning relationships between the given document and query. To create large-scale training data for neural networks, Hermann et al. (2015) released the CNN/Daily Mail news dataset, where the document is formed by the news articles and the queries are extracted from the summary of the news. Hill et al. (2015) released the Children's Book Test dataset afterwards, where the training samples are generated from consecutive 20 sentences from books, and the query is formed by 21st sentence. Following these datasets, a vast variety of neural network approaches have been proposed (Kadlec et al., 2016; Cui et al., 2016; Chen et al., 2016; Dhingra et al., 2016; Sordoni et al., 2016; Trischler et al., 2016; Seo et al., 2016; Xiong et al., 2016), and most of them stem from the attention-based neural network (Bahdanau et al., 2014), which has become a stereotype in most of the NLP tasks and is well-known by its capability of learning the "importance" distribution over the inputs.

In this paper, we present a novel neural network architecture, called *attention-over-attention* model. As we can understand the meaning literally, our model aims to place another attention mechanism over the existing document-level attention. Unlike the previous works, that are using heuristic merging functions (Cui et al., 2016), or setting various pre-defined non-trainable terms (Trischler et al., 2016), our model could automatically generate an "attended attention" over various document-level attentions, and make a mutual look not only from *query-to-document* but also *document-to-query*, which will benefit from the interactive information.

To sum up, the main contributions of our work are listed as follows.

- To our knowledge, this is the first time that

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 593–602
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1055

the mechanism of nesting another attention over the existing attentions is proposed, i.e. *attention-over-attention* mechanism.

- Unlike the previous works on introducing complex architectures or many non-trainable hyper-parameters to the model, our model is much more simple but outperforms various state-of-the-art systems by a large margin.

- We also propose an N-best re-ranking strategy to re-score the candidates in various aspects and further improve the performance.

The following of the paper will be organized as follows. In Section 2, we will give a brief introduction to the cloze-style reading comprehension task as well as related public datasets. Then the proposed attention-over-attention reader will be presented in detail in Section 3 and N-best re-ranking strategy in Section 4. The experimental results and analysis will be given in Section 5 and Section 6. Related work will be discussed in Section 7. Finally, we will give a conclusion of this paper and envisions on future work.

2 Cloze-style Reading Comprehension

In this section, we will give a brief introduction to the cloze-style reading comprehension task at the beginning. And then, several existing public datasets will be described in detail.

2.1 Task Description

Formally, a general Cloze-style reading comprehension problem can be illustrated as a triple:

$$\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$$

The triple consists of a document $\mathcal{D}$, a query $\mathcal{Q}$ and the answer to the query $\mathcal{A}$. Note that the answer is usually a *single* word in the document, which requires the human to exploit context information in both document and query. The type of the answer word varies from predicting a preposition given a fixed collocation to identifying a named entity from a factual illustration.

2.2 Existing Public Datasets

Large-scale training data is essential for training neural networks. Several public datasets for the cloze-style reading comprehension has been released. Here, we introduce two representative and widely-used datasets.

- **CNN / Daily Mail**

Hermann et al. (2015) have firstly published two datasets: CNN and Daily Mail news data [1]. They construct these datasets with web-crawled CNN and Daily Mail news data. One of the characteristics of these datasets is that the news article is often associated with a summary. So they first regard the main body of the news article as the *Document*, and the *Query* is formed by the summary of the article, where one entity word is replaced by a special placeholder to indicate the missing word. The replaced entity word will be the *Answer* of the *Query*. Apart from releasing the dataset, they also proposed a methodology that anonymizes the named entity tokens in the data, and these tokens are also re-shuffle in each sample. The motivation is that the news articles are containing limited named entities, which are usually celebrities, and the world knowledge can be learned from the dataset. So this methodology aims to exploit general relationships between anonymized named entities within a single document rather than the common knowledge. The following research on these datasets showed that the entity word anonymization is not as effective as expected (Chen et al., 2016).

- **Children's Book Test**

There was also a dataset called the Children's Book Test (CBTest) released by Hill et al. (2015), which is built on the children's book story through Project Gutenberg [2]. Different from the CNN/Daily Mail datasets, there is no summary available in the children's book. So they proposed another way to extract query from the original data. The document is composed of 20 consecutive sentences in the story, and the 21st sentence is regarded as the query, where one word is blanked with a special placeholder. In the CBTest datasets, there are four types of sub-datasets available which are classified by the part-of-speech and named entity tag of the answer word, containing Named Entities (NE), Common Nouns (CN), Verbs and Prepositions. In their studies, they have found that the answering of verbs and prepositions are relatively less dependent on the content of document, and the humans can even do preposi-

[1] The pre-processed CNN and Daily Mail datasets are available at http://cs.nyu.edu/~kcho/DMQA/

[2] The CBTest datasets are available at http://www.thespermwhale.com/jaseweston/babi/CBTest.tgz

tion blank-filling without the presence of the document. The studies shown by Hill et al. (2015), answering verbs and prepositions are less dependent with the presence of document. Thus, most of the related works are focusing on solving NE and CN types.

3 Attention-over-Attention Reader

In this section, we will give a detailed introduction to the proposed Attention-over-Attention Reader (AoA Reader). Our model is primarily motivated by Kadlec et al., (2016), which aims to directly estimate the answer from the document-level attention instead of calculating blended representations of the document. As previous studies by Cui et al. (2016) showed that the further investigation of query representation is necessary, and it should be paid more attention to utilizing the information of query. In this paper, we propose a novel work that placing another attention over the primary attentions, to indicate the "importance" of each attentions.

Now, we will give a formal description of our proposed model. When a cloze-style training triple $\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$ is given, the proposed model will be constructed in the following steps.

- **Contextual Embedding**

We first transform every word in the document $\mathcal{D}$ and query $\mathcal{Q}$ into one-hot representations and then convert them into continuous representations with a shared embedding matrix W_e. By sharing word embedding, both the document and query can participate in the learning of embedding and both of them will benefit from this mechanism. After that, we use two bi-directional RNNs to get contextual representations of the document and query individually, where the representation of each word is formed by concatenating the forward and backward hidden states. After making a trade-off between model performance and training complexity, we choose the Gated Recurrent Unit (GRU) (Cho et al., 2014) as recurrent unit implementation.

$$e(x) = W_e \cdot x, \ where \ \ x \in \mathcal{D}, \mathcal{Q} \quad (1)$$

$$\overrightarrow{h_s(x)} = \overrightarrow{GRU}(e(x)) \quad (2)$$

$$\overleftarrow{h_s(x)} = \overleftarrow{GRU}(e(x)) \quad (3)$$

$$h_s(x) = [\overrightarrow{h_s(x)}; \overleftarrow{h_s(x)}] \quad (4)$$

We take $h_{doc} \in \mathbb{R}^{|\mathcal{D}|*2d}$ and $h_{query} \in \mathbb{R}^{|\mathcal{Q}|*2d}$ to denote the contextual representations of document and query, where d is the dimension of GRU (one-way).

- **Pair-wise Matching Score**

After obtaining the contextual embeddings of the document h_{doc} and query h_{query}, we calculate a pair-wise matching matrix, which indicates the pair-wise matching degree of one document word and one query word. Formally, when given ith word of the document and jth word of query, we can compute a matching score by their dot product.

$$M(i, j) = h_{doc}(i)^T \cdot h_{query}(j) \quad (5)$$

In this way, we can calculate every pair-wise matching score between each document and query word, forming a matrix $M \in \mathbb{R}^{|\mathcal{D}|*|\mathcal{Q}|}$, where the value of ith row and jth column is filled by $M(i, j)$.

- **Individual Attentions**

After getting the pair-wise matching matrix M, we apply a column-wise softmax function to get probability distributions in each column, where each column is an individual document-level attention when considering a single query word. We denote $\alpha(t) \in \mathbb{R}^{|\mathcal{D}|}$ as the document-level attention regarding query word at time t, which can be seen as a *query-to-document* attention.

$$\alpha(t) = softmax(M(1, t), ..., M(|\mathcal{D}|, t)) \quad (6)$$

$$\alpha = [\alpha(1), \alpha(2), ..., \alpha(|\mathcal{Q}|)] \quad (7)$$

- **Attention-over-Attention**

Different from Cui et al. (2016), instead of using naive heuristics (such as *summing* or *averaging*) to combine these individual attentions into a final attention, we introduce another attention mechanism to automatically decide the importance of each individual attention.

First, we calculate a reversed attention, that is, for every document word at time t, we calculate the "importance" distribution on the query, to indicate which query words are more important given a single document word. We apply a row-wise softmax function to the pair-wise matching matrix M to get query-level attentions. We denote $\beta(t) \in \mathbb{R}^{|\mathcal{Q}|}$ as the query-level attention regarding document word at time t, which can be seen as a

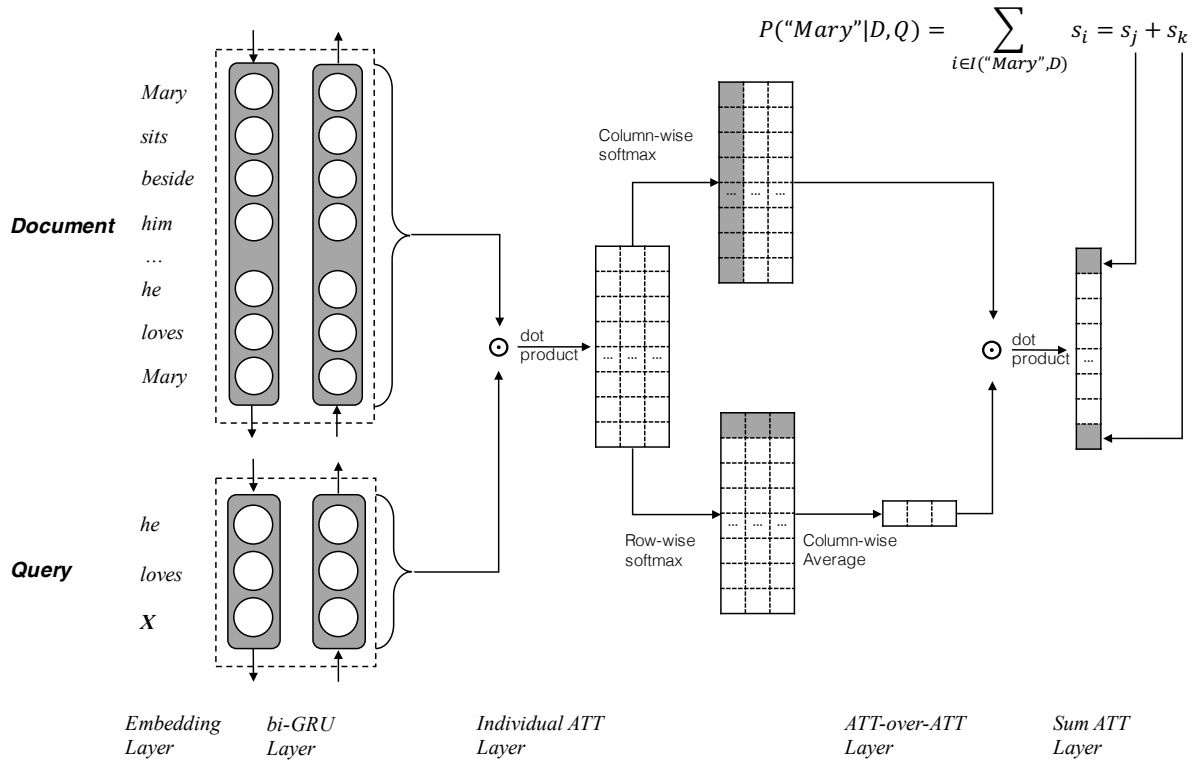


Figure 1: Neural network architecture of the proposed Attention-over-Attention Reader (AoA Reader).

document-to-query attention.

$$\beta(t) = softmax(M(t,1), ..., M(t,|\mathcal{Q}|)) \quad (8)$$

So far, we have obtained both *query-to-document* attention α and *document-to-query* attention β. Our motivation is to exploit mutual information between the document and query. However, most of the previous works are only relying on *query-to-document* attention, that is, only calculate one document-level attention when considering the whole query.

Then we average all the $\beta(t)$ to get an averaged query-level attention β. Note that, we do not apply another softmax to the β, because averaging individual attentions do not break the normalizing condition.

$$\beta = \frac{1}{n} \sum_{t=1}^{|\mathcal{D}|} \beta(t) \quad (9)$$

Finally, we calculate dot product of α and β to get the "attended document-level attention" $s \in \mathbb{R}^{|\mathcal{D}|}$, i.e. the *attention-over-attention* mechanism. Intuitively, this operation is calculating a weighted sum of each individual document-level attention $\alpha(t)$ when looking at query word at time t. In this way, the contributions by each query word can be learned explicitly, and the final decision (document-level attention) is made through the voted result by the importance of each query word.

$$s = \alpha^T \beta \quad (10)$$

- **Final Predictions**

Following Kadlec et al. (2016), we use *sum attention* mechanism to get aggregated results. Note that the final output should be reflected in the vocabulary space V, rather than document-level attention $|\mathcal{D}|$, which will make a significant difference in the performance, though Kadlec et al. (2016) did not illustrate this clearly.

$$P(w|\mathcal{D},\mathcal{Q}) = \sum_{i \in I(w,\mathcal{D})} s_i, \quad w \in V \quad (11)$$

where $I(w,\mathcal{D})$ indicate the positions that word w appears in the document $\mathcal{D}$. As the training objectives, we seek to maximize the log-likelihood of the correct answer.

$$\mathcal{L} = \sum_i \log(p(x)) \quad , x \in \mathcal{A} \quad (12)$$

	CNN News			CBT NE			CBT CN		
	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
# Query	380,298	3,924	3,198	108,719	2,000	2,500	120,769	2,000	2,500
Max # candidates	527	187	396	10	10	10	10	10	10
Avg # candidates	26	26	25	10	10	10	10	10	10
Avg # tokens	762	763	716	433	412	424	470	448	461
Vocabulary	118,497			53,063			53,185		

Table 1: Statistics of cloze-style reading comprehension datasets: CNN news and CBTest NE / CN.

The proposed neural network architecture is depicted in Figure 1. Note that, as our model mainly adds limited steps of calculations to the AS Reader (Kadlec et al., 2016) and does not employ any additional weights, the computational complexity is similar to the AS Reader.

4 N-best Re-ranking Strategy

Intuitively, when we do cloze-style reading comprehensions, we often refill the candidate into the blank of the query to double-check its appropriateness, fluency and grammar to see if the candidate we choose is the most suitable one. If we do find some problems in the candidate we choose, we will choose the second possible candidate and do some checking again.

To mimic the process of double-checking, we propose to use N-best re-ranking strategy after generating answers from our neural networks. The procedure can be illustrated as follows.

- **N-best Decoding**

Instead of only picking the candidate that has the highest possibility as answer, we can also extract follow-up candidates in the decoding process, which forms an N-best list.

- **Refill Candidate into Query**

As a characteristic of the cloze-style problem, each candidate can be refilled into the blank of the query to form a complete sentence. This allows us to check the candidate according to its context.

- **Feature Scoring**

The candidate sentences can be scored in many aspects. In this paper, we exploit three features to score the N-best list.

  - Global N-gram LM: This is a fundamental metric in scoring sentence, which aims to evaluate its fluency. This model is trained on the document part of training data.
  - Local N-gram LM: Different from global LM, the local LM aims to explore the information with the given document, so the statistics are obtained from the test-time document. It should be noted that the local LM is trained sample-by-sample, it is not trained on the entire test set, which is not legal in the real test case. This model is useful when there are many unknown words in the test sample.
  - Word-class LM: Similar to global LM, the word-class LM is also trained on the document part of training data, but the words are converted to its word class ID. The word class can be obtained by using clustering methods. In this paper, we simply utilized the *mkcls* tool for generating 1000 word classes (Josef Och, 1999).

- **Weight Tuning**

To tune the weights among these features, we adopt the K-best MIRA algorithm (Cherry and Foster, 2012) to automatically optimize the weights on the validation set, which is widely used in statistical machine translation tuning procedure.

- **Re-scoring and Re-ranking**

After getting the weights of each feature, we calculate the weighted sum of each feature in the N-best sentences and then choose the candidate that has the lowest cost as the final answer.

5 Experiments

5.1 Experimental Setups

The general settings of our neural network model are listed below in detail.

- Embedding Layer: The embedding weights are randomly initialized with the uniformed distribution in the interval $[-0.05, 0.05]$.

	CNN News		CBTest NE		CBTest CN	
	Valid	Test	Valid	Test	Valid	Test
Deep LSTM Reader (Hermann et al., 2015)	55.0	57.0	-	-	-	-
Attentive Reader (Hermann et al., 2015)	61.6	63.0	-	-	-	-
Human (context+query) (Hill et al., 2015)	-	-	-	*81.6*	-	*81.6*
MemNN (window + self-sup.) (Hill et al., 2015)	63.4	66.8	70.4	66.6	64.2	63.0
AS Reader (Kadlec et al., 2016)	68.6	69.5	73.8	68.6	68.8	63.4
CAS Reader (Cui et al., 2016)	68.2	70.0	74.2	69.2	68.2	65.7
Stanford AR (Chen et al., 2016)	72.4	72.4	-	-	-	-
GA Reader (Dhingra et al., 2016)	73.0	73.8	74.9	69.0	69.0	63.9
Iterative Attention (Sordoni et al., 2016)	72.6	73.3	75.2	68.6	72.1	69.2
EpiReader (Trischler et al., 2016)	**73.4**	74.0	75.3	69.7	71.5	67.4
AoA Reader	73.1	**74.4**	**77.8**	**72.0**	**72.2**	**69.4**
AoA Reader + Reranking	-	-	**79.6**	**74.0**	**75.7**	**73.1**
MemNN (Ensemble)	66.2	69.4	-	-	-	-
AS Reader (Ensemble)	73.9	75.4	74.5	70.6	71.1	68.9
GA Reader (Ensemble)	76.4	77.4	75.5	71.9	72.1	69.4
EpiReader (Ensemble)	-	-	76.6	71.8	73.6	70.6
Iterative Attention (Ensemble)	74.5	75.7	76.9	72.0	74.1	**71.0**
AoA Reader (Ensemble)	-	-	**78.9**	**74.5**	**74.7**	70.8
AoA Reader (Ensemble + Reranking)	-	-	**80.3**	**75.6**	**77.0**	**74.1**

Table 2: Results on the CNN news, CBTest NE and CN datasets. The best baseline results are depicted in italics, and the overall best results are in bold face.

For regularization purpose, we adopted l_2-regularization to 0.0001 and dropout rate of 0.1 (Srivastava et al., 2014). Also, it should be noted that we do not exploit any pre-trained embedding models.

- Hidden Layer: Internal weights of GRUs are initialized with random orthogonal matrices (Saxe et al., 2013).

- Optimization: We adopted ADAM optimizer for weight updating (Kingma and Ba, 2014), with an initial learning rate of 0.001. As the GRU units still suffer from the gradient exploding issues, we set the gradient clipping threshold to 5 (Pascanu et al., 2013). We used batched training strategy of 32 samples.

Dimensions of embedding and hidden layer for each task are listed in Table 3. In re-ranking step, we generate 5-best list from the baseline neural network model, as we did not observe a significant variance when changing the N-best list size. All language model features are trained on the training proportion of each dataset, with 8-gram word-based setting and Kneser-Ney smoothing (Kneser and Ney, 1995) trained by SRILM toolkit (Stolcke, 2002). The results are reported with the best model, which is selected by the performance of validation set. The ensemble model is made up of four best models, which are trained using different random seed. Implementation is done with Theano (Theano Development Team, 2016) and Keras (Chollet, 2015), and all models are trained on Tesla K40 GPU.

	Embed. # units	Hidden # units
CNN News	384	256
CBTest NE	384	384
CBTest CN	384	256

Table 3: Embedding and hidden layer dimensions for each task.

5.2 Overall Results

Our experiments are carried out on public datasets: CNN news datasets (Hermann et al., 2015) and CBTest NE/CN datasets (Hill et al., 2015). The statistics of these datasets are listed in Table 1, and the experimental results are given in Table 2.

As we can see that, our AoA Reader outperforms state-of-the-art systems by a large margin, where 2.3% and 2.0% absolute improvements over EpiReader in CBTest NE and CN test sets, which demonstrate the effectiveness of our model. Also by adding additional features in the re-ranking step, there is another significant boost 2.0% to 3.7% over AoA Reader in CBTest NE/CN test sets. We have also found that our single model could stay on par with the previous best ensemble system, and even we have an absolute improvement of 0.9% beyond the best ensemble model (Iterative Attention) in the CBTest NE validation set. When it comes to ensemble model, our AoA Reader also shows significant improvements over previous best ensemble models by a large margin and set up a new state-of-the-art system.

To investigate the effectiveness of employing *attention-over-attention* mechanism, we also compared our model to CAS Reader, which used pre-defined merging heuristics, such as *sum* or *avg* etc. Instead of using pre-defined merging heuristics, and letting the model explicitly learn the weights between individual attentions results in a significant boost in the performance, where 4.1% and 3.7% improvements can be made in CNN validation and test set against CAS Reader.

5.3 Effectiveness of Re-ranking Strategy

As we have seen that the re-ranking approach is effective in cloze-style reading comprehension task, we will give a detailed ablations in this section to show the contributions by each feature. To have a thorough investigation in the re-ranking step, we listed the detailed improvements while adding each feature mentioned in Section 4.

From the results in Table 4, we found that the NE and CN category both benefit a lot from the re-ranking features, but the proportions are quite different. Generally speaking, in NE category, the performance is mainly boosted by the LM_{local} feature. However, on the contrary, the CN category benefits from LM_{global} and LM_{wc} rather than the LM_{local}.

Also, we listed the weights of each feature in Table 5. The LM_{global} and LM_{wc} are all trained by training set, which can be seen as *Global Feature*. However, the LM_{local} is only trained within the respective document part of test sample, which can be seen as *Local Feature*.

$$\eta = \frac{LM_{global} + LM_{wc}}{LM_{local}} \tag{13}$$

	CBTest NE		CBTest CN	
	Valid	Test	Valid	Test
AoA Reader	77.8	72.0	72.2	69.4
+Global LM	78.3	72.6	73.9	71.2
+Local LM	79.4	73.8	74.7	71.7
+Word-class LM	79.6	74.0	75.7	73.1

Table 4: Detailed results of 5-best re-ranking on CBTest NE/CN datasets. Each row includes all of the features from previous rows. LM_{global} denotes the global LM, LM_{local} denotes the local LM, LM_{wc} denotes the word-class LM.

	CBTest NE	CBTest CN
NN	0.64	0.20
Global LM	0.16	0.10
Word-class LM	0.04	0.39
Local LM	0.16	0.31
RATIO η	1.25	1.58

Table 5: Weight of each feature in N-best re-ranking step. *NN* denotes the feature (probability) produced by baseline neural network model.

We calculated the ratio between the global and local features and found that the NE category is much more dependent on local features than CN category. Because it is much more likely to meet a new named entity than a common noun in the test phase, so adding the local LM provides much more information than that of common noun. However, on the contrary, answering common noun requires less local information, which can be learned in the training data relatively.

6 Quantitative Analysis

In this section, we will give a quantitative analysis to our AoA Reader. The following analyses are carried out on CBTest NE dataset. First, we investigate the relations between the length of the document and corresponding accuracy. The result is depicted in Figure 2.

As we can see that the AoA Reader shows consistent improvements over AS Reader on the different length of the document. Especially, when the length of document exceeds 700, the improvements become larger, indicating that the AoA Reader is more capable of handling long documents.

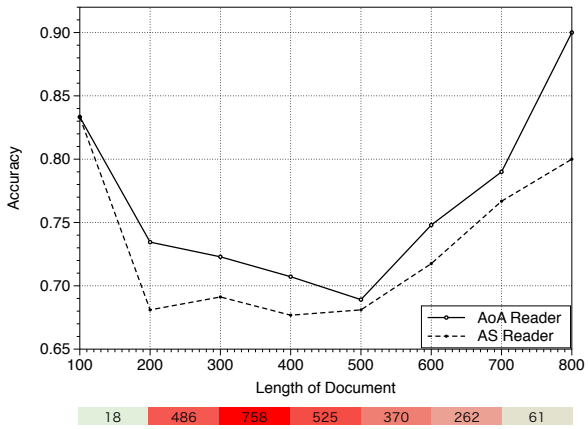


Figure 2: Test accuracy against the length of the document. The bar below the figure indicates the number of samples in each interval.

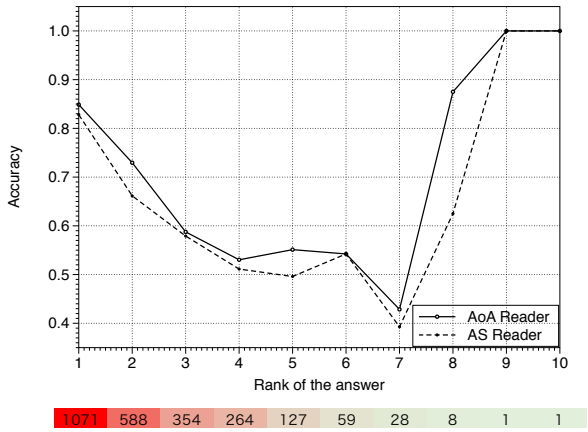


Figure 3: Test accuracy against the frequency rank of the answer. The bar below the figure indicates the number of samples in each rank.

Furthermore, we also investigate if the model tends to choose a high-frequency candidate than a lower one, which is shown in Figure 3. Not surprisingly, we found that both models do a good job when the correct answer appears more frequent in the document than the other candidates. This is because that the correct answer that has the highest frequency among the candidates takes up over 40% of the test set (1071 out of 2500). But interestingly we have also found that, when the frequency rank of correct answer exceeds 7 (less frequent among candidates), these models also give a relatively high performance. Empirically, we think that these models tend to choose extreme cases in terms of candidate frequency (either too high or too low). One possible reason is that it is hard for the model to choose a candidate that has a neutral frequency as the correct answer, because of its ambiguity (neutral choices are hard to made).

7 Related Work

Cloze-style reading comprehension tasks have been widely investigated in recent studies. We will take a brief revisit to the related works.

Hermann et al. (2015) have proposed a method for obtaining large quantities of $\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$ triples through news articles and its summary. Along with the release of cloze-style reading comprehension dataset, they also proposed an attention-based neural network to handle this task. Experimental results showed that the proposed neural network is effective than traditional baselines.

Hill et al. (2015) released another dataset, which stems from the children's books. Different from Hermann et al. (2015)'s work, the document and query are all generated from the raw story without any summary, which is much more general than previous work. To handle the reading comprehension task, they proposed a window-based memory network, and self-supervision heuristics is also applied to learn hard-attention.

Unlike previous works, that using blended representations of document and query to estimate the answer, Kadlec et al. (2016) proposed a simple model that directly pick the answer from the document, which is motivated by the Pointer Network (Vinyals et al., 2015). A restriction of this model is that the answer should be a single word and appear in the document. Results on various public datasets showed that the proposed model is effective than previous works.

Liu et al. (2016) proposed to exploit reading comprehension models to other tasks. They first applied the reading comprehension model into Chinese zero pronoun resolution task with automatically generated large-scale pseudo training data. The experimental results on OntoNotes 5.0 data showed that their method significantly outperforms various state-of-the-art systems.

Our work is primarily inspired by Cui et al. (2016) and Kadlec et al. (2016) , where the latter model is widely applied to many follow-up works (Sordoni et al., 2016; Trischler et al., 2016; Cui et al., 2016). Unlike the CAS Reader (Cui et al., 2016), we do not assume any heuristics to our model, such as using merge functions: sum, avg etc. We used a mechanism called "attention-

over-attention" to explicitly calculate the weights between different individual document-level attentions, and get the final attention by computing the weighted sum of them. Also, we find that our model is typically general and simple than the recently proposed model, and brings significant improvements over these cutting edge systems.

8 Conclusion

We present a novel neural architecture, called attention-over-attention reader, to tackle the cloze-style reading comprehension task. The proposed AoA Reader aims to compute the attentions not only for the document but also the query side, which will benefit from the mutual information. Then a weighted sum of attention is carried out to get an attended attention over the document for the final predictions. Among several public datasets, our model could give consistent and significant improvements over various state-of-the-art systems by a large margin.

The future work will be carried out in the following aspects. We believe that our model is general and may apply to other tasks as well, so firstly we are going to fully investigate the usage of this architecture in other tasks. Also, we are interested to see that if the machine really "comprehend" our language by utilizing neural networks approaches, but not only serve as a "document-level" language model. In this context, we are planning to investigate the problems that need comprehensive reasoning over several sentences.

Acknowledgments

We would like to thank all three anonymous reviewers for their thorough reviewing and providing thoughtful comments to improve our paper. This work was supported by the National 863 Leading Technology Research Project via grant 2015AA015409.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Danqi Chen, Jason Bolton, and D. Christopher Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2358–2367. https://doi.org/10.18653/v1/P16-1223.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Montréal, Canada, pages 427–436. http://www.aclweb.org/anthology/N12-1047.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1724–1734. http://aclweb.org/anthology/D14-1179.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Yiming Cui, Ting Liu, Zhipeng Chen, Shijin Wang, and Guoping Hu. 2016. Consensus attention-based neural networks for chinese reading comprehension. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 1777–1786. http://aclweb.org/anthology/C16-1167.

Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* .

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

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*. http://aclweb.org/anthology/E99-1010.

Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 908–918. https://doi.org/10.18653/v1/P16-1086.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing*. pages 181–184 vol.1.

Ting Liu, Yiming Cui, Qingyu Yin, Shijin Wang, Weinan Zhang, and Guoping Hu. 2016. Generating and exploiting large-scale pseudo training data for zero pronoun resolution. *arXiv preprint arXiv:1606.01603* .

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28:1310–1318.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120* .

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2016. Bi-directional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245* .

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Andreas Stolcke. 2002. Srilm — an extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002)*. pages 901–904.

Wilson L Taylor. 1953. Cloze procedure: a new tool for measuring readability. *Journalism and Mass Communication Quarterly* 30(4):415.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.

Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni, and Kaheer Suleman. 2016. Natural language comprehension with the epireader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 128–137. http://aclweb.org/anthology/D16-1013.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Alignment at Work: Using Language to Distinguish the Internalization and Self-Regulation Components of Cultural Fit in Organizations

Gabriel Doyle
Department of Psychology
Stanford University
gdoyle@stanford.edu

Amir Goldberg
Graduate School of Business
Stanford University
amirgo@stanford.edu

Sameer B. Srivastava
Haas School of Business
UC Berkeley
srivastava@haas.berkeley.edu

Michael C. Frank
Department of Psychology
Stanford University
mcfrank@stanford.edu

Abstract

Cultural fit is widely believed to affect the success of individuals and the groups to which they belong. Yet it remains an elusive, poorly measured construct. Recent research draws on computational linguistics to measure cultural fit but overlooks asymmetries in cultural adaptation. By contrast, we develop a directed, dynamic measure of cultural fit based on linguistic alignment, which estimates the influence of one person's word use on another's and distinguishes between two enculturation mechanisms: internalization and self-regulation. We use this measure to trace employees' enculturation trajectories over a large, multi-year corpus of corporate emails and find that patterns of alignment in the first six months of employment are predictive of individuals downstream outcomes, especially involuntary exit. Further predictive analyses suggest referential alignment plays an overlooked role in linguistic alignment.

1 Introduction

Entering a new group is rarely easy. Adjusting to unfamiliar behavioral norms and donning a new identity can be cognitively and emotionally taxing, and failure to do so can lead to exclusion. But successful enculturation to the group often yields significant rewards, especially in organizational contexts. Fitting in has been tied to positive career outcomes such as faster time-to-promotion, higher performance ratings, and reduced risk of being fired (O'Reilly et al., 1991; Goldberg et al., 2016).

A major challenge for enculturation research is distinguishing between *internalization* and *self-regulation*. Internalization, a more inwardly focused process, involves identifying as a group member and accepting group norms, while self-regulation, a more outwardly oriented process, entails deciphering the group's normative code and adjusting one's behavior to comply with it. Existing approaches, which generally rely on self-reports, are subject to various forms of reporting bias and typically yield only static snapshots of this process. Recent computational approaches that use language as a behavioral signature of group integration uncover dynamic traces of enculturation but cannot distinguish between internalization and self-regulation.

To overcome these limitations, we introduce a dynamic measure of *directed* linguistic accommodation between a newcomer and existing group members. Our approach differentiates between an individual's (1) *base rate* of word use and (2) *linguistic alignment* to interlocutors. The former corresponds to internalization of the group's linguistic norms, whereas the latter reflects the capacity to regulate one's language in response to peers' language use. We apply this language model to a corpus of internal email communications and personnel records, spanning a seven-year period, from a mid-sized technology firm. We show that changes in base rates and alignment, especially with respect to pronoun use, are consistent with successful assimilation into a group and can predict eventual employment outcomes—continued employment, involuntary exit, or voluntary exit—at levels above chance. We use this predictive problem to investigate the nature of linguistic alignment. Our results suggest that the common formulation of alignment as a lexical-level phenomenon is incomplete.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 603–612
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1056

2 Linguistic Alignment and Group Fit

Linguistic alignment Linguistic alignment is the tendency to use the same or similar words as one's conversational partner. Alignment is an instance of a widespread and socially important human behavior: *communication accommodation*, the tendency of two interacting people to nonconsciously adopt similar behaviors. Evidence of accommodation appears in many behavioral dimensions, including gestures, postures, speech rate, self-disclosure, and language or dialect choice (see Giles et al. (1991) for a review). More accommodating people are rated by their interlocutors as more intelligible, attractive, and cooperative (Feldman, 1968; Ireland et al., 2011; Triandis, 1960). These perceptions have material consequences—for example, high accommodation requests are more likely to be fulfilled, and pairs who accommodate more in how they express uncertainty perform better in lab-based tasks (Buller and Aune, 1988; Fusaroli et al., 2012).

Although accommodation is ubiquitous, individuals vary in their levels of accommodation in ways that are socially informative. Notably, more powerful people are accommodated more strongly in many settings, including trials (Gnisci, 2005), online forums (Danescu-Niculescu-Mizil et al., 2012), and Twitter (Doyle et al., 2016). Most relevant for this work, speakers may increase their accommodation to signal camaraderie or decrease it to differentiate from the group. For example, Bourhis and Giles (1977) found that Welsh English speakers increased their use of the Welsh accent and language in response to an English speaker who dismissed it.

Person-group fit and linguistic alignment These findings suggest that linguistic alignment is a useful avenue for studying how people assimilate into a group. Whereas traditional approaches to studying person-group fit rely on self-reports that are subject to various forms of reporting bias and cannot feasibly be collected with high granularity across many points in time, recent studies have proposed language-based measures as a means to tracing the dynamics of person-group fit without having to rely on self-reports. Building on Danescu-Niculescu-Mizil et al. (2013)'s research into language use similarities as a proxy for social distance between individuals, Srivastava et al. (forthcoming) and Goldberg et al. (2016) developed a measure of cultural fit based on the similarity in linguistic style between individuals and their colleagues in an organization. Their time-varying measure highlights linguistic compatibility as an important facet of cultural fit and reveals distinct trajectories of enculturation for employees with different career outcomes.

While this approach can help uncover the dynamics and consequences of an individual's fit with her colleagues in an organization, it cannot disentangle the underlying reasons for this alignment. For two primary reasons, it cannot distinguish between fit that arises from internalization and fit produced by self-regulation. First, Goldberg et al. (2016) and Srivastava et al. (forthcoming) define fit using a symmetric measure, the Jensen-Shannon divergence, which does not take into account the direction of alignment. Yet the distinction between an individual adapting to peers versus peers adapting to the individual would appear to be consequential. Second, this prior work considers fit across a wide range of linguistic categories but does not interrogate the role of particular categories, such as pronouns, that can be especially informative about enculturation. For example, a person's base rate use of the first-person singular (*I*) or plural (*we*) might indicate the degree of group identity internalization, whereas adjustment to *we* usage in response to others' use of the pronoun might reveal the degree of self-regulation to the group's normative expectations.

Modeling fit with WHAM To address these limitations, we build upon and extend the WHAM alignment framework (Doyle and Frank, 2016) to analyze the dynamics of internalization and self-regulation using the complete corpus of email communications and personnel records from a mid-sized technology company over a seven-year period. WHAM uses a conditional measure of alignment, separating overall homophily (unconditional similarity in people's language use, driven by *internalized* similarity) from in-the-moment adaptation (adjusting to another's usage, corresponding to *self-regulation*). WHAM also provides a directed measure of alignment, in that it estimates a *replier's* adaptation to the other conversational participant separately from the participant's adaptation to the replier.

Level(s) of alignment The convention within linguistic alignment research, dating back to early

work on Linguistic Style Matching (Niederhoffer and Pennebaker, 2002), is to look at lexical alignment: the repetition of the same or similar words across conversation participants. From a communication accommodation standpoint, this is justified by assuming that one's choice of words represents a stylistic signal that is partially independent of the meaning one intends to express—similar to the accommodation on paralinguistic signals discussed above. The success of previous linguistic alignment research shows that this is valid.

However, words are difficult to divorce from their meanings, and sometimes repeating a word conflicts with repeating its referent. In particular, pronouns often refer to different people depending on who uses the pronoun. While there is evidence that one person using a first-person singular pronoun increases the likelihood that her conversation partner will as well (Chung and Pennebaker, 2007), we may also expect that one person using first-person singular pronouns may cause the other to use more second-person pronouns, so that both people are referring to the same person. This is especially important under the Interactive Alignment Model view (Pickering and Garrod, 2004), where conversants align their entire mental representations, which predicts both lexical and referential alignment behaviors will be observed. Discourse-strategic explanations for alignment also predict alignment at multiple levels (Doyle and Frank, 2016).

Since we have access to a high-quality corpus with meaningful outcome measures, we can investigate the relative importance of these two types of alignment. We will show that referential alignment is more predictive of employment outcomes than is lexical alignment, suggesting a need for alignment research to consider both levels rather than just the latter.

3 Data: Corporate Email Corpus

We use the complete corpus of internal emails exchanged among full-time employees at a mid-sized US-based technology company between 2009 to 2014 (Srivastava et al., forthcoming). Each email was summarized as a count of word categories in its text. These categories are a subset of the Linguistic Information and Word Count system (Pennebaker et al., 2007). The categories were chosen because they are likely to be indicative of one's standing/role within a group.[1]

We divided email chains into message-reply pairs to investigate conditional alignment between a message and its reply. To limit these pairs to cases where the reply was likely related to the preceding message, we removed all emails with more than one sender or recipient (including CC/BCC), identical sender and recipient, or where the sender or recipient was an automatic notification system or any other mailbox that was not specific to a single employee. We also excluded emails with no body text or more than 500 words in the body text, and pairs with more than a week's latency between message and reply.

Finally, because our analyses involve enculturation dynamics over the first six months of employment, we excluded replies sent by an employee whose overall tenure was less than six months. This resulted in a collection of 407,779 message-reply pairs, with 485 distinct replying employees. We combined this with monthly updates of employees joining and leaving the company and whether they left voluntarily or involuntarily. Of the 485, 66 left voluntarily, 90 left involuntarily, and 329 remained employed at the end of the observation period.

Privacy protections and ethical considerations Research based on employees' archived electronic communications in organizational settings poses potential threats to employee privacy and company confidentiality. To address these concerns, and following established ethical guidelines for the conduct of such research (Borgatti and Molina, 2003), we implemented the following procedures: (a) raw data were stored on secure research servers behind the company's firewall; (b) messages exchanged with individuals outside the firm were eliminated; (c) all identifying information such as email addresses was transformed into hashed identifiers, with the company retaining access to the key code linking identifying information to hashed identifiers; and (d) raw message content was transformed into linguistic categories so that identities could not be inferred from message content. Per terms of the non-disclosure agreement we signed with the firm, we are not able to share the data underlying the analyses reported below.

[1] Six pronoun categories (first singular (*I*), first plural (*we*), second (*you*), third singular personal (*he, she*), third singular impersonal (*it, this*), and third plural (*they*)) and five time/certainty categories (past tense, present tense, future tense, certainty, and tentativity).

We can, however, share the code and dummy test data, both of which can be accessed at http://github.com/gabedoyle/acl2017.

4 Model: An Extended WHAM Framework

To assess alignment, we use the Word-Based Hierarchical Alignment Model (WHAM) framework (Doyle and Frank, 2016). The core principle of WHAM is that alignment is a change, usually an increase, in the frequency of using a word category in a reply when the word category was used in the preceding message. For instance, a reply to the message *What **will** we discuss at the meeting?*, is likely to have more instances of future tense than a reply to the message *What **did** we discuss at the meeting?* Under this definition, alignment is the log-odds shift from the *baseline* reply frequency, the frequency of the word in a reply when the preceding message did **not** contain the word.

WHAM is a hierarchical generative modeling framework, so it uses information from related observations (e.g., multiple repliers with similar demographics) to improve its robustness on sparse data (Doyle et al., 2016). There are two key parameters, shown in Figure 2: η^{base}, the log-odds of a given word category c when the preceding message did not contain c, and η^{align}, the increase in the log-odds of c when the preceding message did contain c.

A dynamic extension To understand enculturation, we need to track changes in both the alignment and baseline over time. We add a month-by-month change term to WHAM, yielding a piecewise linear model of these factors over the course of an employee's tenure. Each employee's tenure is broken into two or three segments: their first six months after being hired, their last six months before leaving (if they leave), and the rest of their tenure.[2] The linear segments for their alignment are fit as an intercept term η^{align}, based at their first month (for the initial period) or their last month (for the final period), and per-month slopes α. Baseline segments are fit similarly, with parameters η^{base} and β.[3] To visualize the alignment behaviors and the parameter values, we create "sawhorse" plots, with an example in Figure 1.

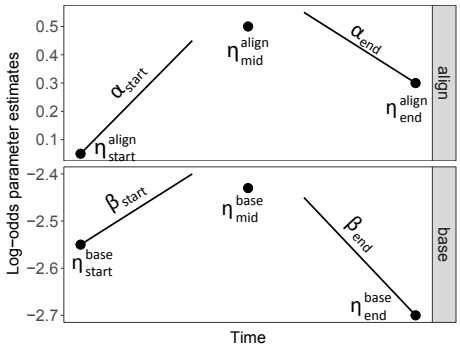


Figure 1: Sample sawhorse plot with key variables labelled. The η point parameters (first month, last month, and middle average) and α (or β) by-month slope (start, end) parameters are estimated by WHAM for each word category and employee group.

In our present work, we are focused on changes in cultural fit during the transitions into or out of the group, so we collapse observations outside the first/last six months into a stable point estimate, constraining their slopes to be zero. This simplification also circumvents the issue of different employees having different middle-period lengths.[4]

Model structure The graphical model for our instantiation of WHAM is shown in Figure 2. For each word category c, WHAM's generative model represents each reply as a series of token-by-token independent draws from a binomial distribution. The binomial probability μ is dependent on whether the preceding message did (μ^{align}) or did not (μ^{base}) contain a word from category c, and the inferred alignment value is the difference between these probabilities in log-odds space (η^{align}).

The specific values of these variables depend on three hierarchical features: the word category c, the group g that a given employee falls into, and the time period t (a piece of the piece-wise

[2]Within each segment, the employee's alignment model is similar to that of Yurovsky et al. (2016), who introduced a constant by-month slope parameter to model changes in parent-child alignment during early linguistic development.

[3]The six month timeframe was chosen as previous research has found it to be a critical period for early enculturation (Bauer et al., 1998). Pilot investigations into the change in baseline usage over time showed roughly linear changes over the first/last six months, but our linearity assumption may mask interesting variation in the enculturation trajectories.

[4]As shown in Figure 1, the pieces do not need to define a continuous function. Alignment behaviors continue to change in the middle of an employee's tenure (Srivastava et al., forthcoming), so alignment six months in to the job is unlikely to be equal to alignment six months from leaving, or the average alignment over the middle tenure.

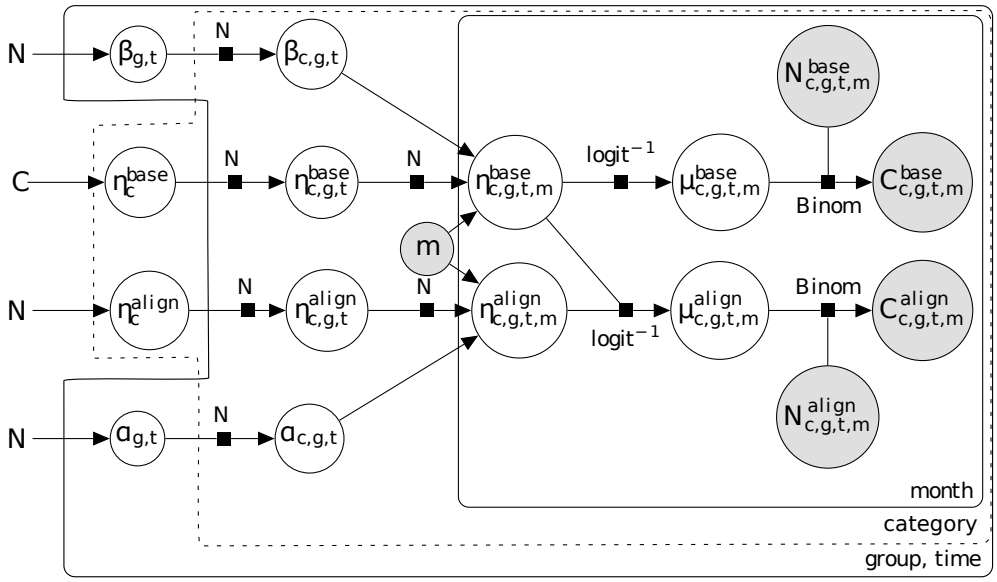


Figure 2: The Word-Based Hierarchical Alignment Model (WHAM). Hierarchical chains of normal distributions capture relationships between word categories, individuals, outcome groups, and time, and generate linear predictors η, which are converted into probabilities μ for binomial draws of the words in replies.

linear function: beginning, middle, or end). Note that the hierarchical ordering is different for the η chains and the α/β chains; c is above g and t for the η chains, but below them for the α/β chains. This is because we expect the static (η) values for a given word category to be relatively consistent across different groups and at different times, but we expect the values to be independent across the different word categories. Conversely, we expect that the enculturation trajectories across word categories (α/β) will be similar, while the trajectories may vary substantially across different groups and different times. Lastly, the month m in which a reply is written (measured from the start of the time period t) has a linear effect on the η value, as described below.

To estimate alignment, we first divide the replies up by group, time period, and calendar month. We separate the replies into two sets based on whether the preceding message contained the category c (the "alignment" set) or not (the "baseline" set). All replies within a set are then aggregated in a single bag-of-words representation, with category token counts $C^{align}_{c,g,t,m}$ and $C^{base}_{c,g,t,m}$, and total token counts $N^{base}_{c,g,t,m}$ and $N^{base}_{c,g,t,m}$ comprising the observed variables on the far right of the model. Moving from right to left, these counts are assumed to come from binomial draws with probability $\mu^{align}_{c,g,t,m}$ or $\mu^{base}_{c,g,t,m}$. The μ values are then in turn generated from η values in log-odds space by an inverse-logit transform, similar to linear predictors in logistic regression.

The η^{base} variables are representations of the baseline frequency of a marker in log-odds space, and μ^{base} is simply a conversion of η^{base} to probability space, the equivalent of an intercept term in a logistic regression. η^{align} is an additive value, with $\mu^{align} = \text{logit}^{-1}(\eta^{base} + \eta^{align})$, the equivalent of a binary feature coefficient in a logistic regression. The specific month's η variables are calculated as a linear function: $\eta^{align}_{c,g,t,m} = \eta^{align}_{c,g,t} + m\alpha_{c,g,t}$, and similarly with β for the baseline.

The remainder of the model is a hierarchy of normal distributions that integrate social structure into the analysis. In the present work, we have three levels in the hierarchy: category, group, and time period. In Analysis 1, employees are grouped by their employment outcome (stay, leave voluntarily, leave involuntarily); in Analyses 2 & 3, where we predict the employment outcomes, each group is a single employee. The normal distributions that connect these levels have identical standard deviations $\sigma^2 = .25$.[5] The hierarchies

[5]The deviation is not a theoretically motivated choice, and was chosen as a good empirical balance between reasonable parameter convergence (improved by smaller σ^2) and good model log-probability (improved by larger σ^2).

are headed by a normal distribution centered at 0, except for the η_{base} hierarchy, which has a $Cauchy(0, 2.5)$ distribution.[6]

Message and reply length can affect alignment estimates; the WHAM model was developed in part to reduce this effect. As different employees had different email length distributions, we further accounted for length by dividing all replies into five quintile length bins, and treated each bin as separate observations for each employee. This design choice adds an additional control factor, but results were qualitatively similar without it. All of our analyses are based on parameter estimates from RStan fits of WHAM with 500 iterations over four chains.

While previous research on cultural fit has emphasized either its internalization (O'Reilly et al., 1991) or self-regulation (Goldberg et al., 2016) components, our extension to the WHAM framework helps disentangle them by estimating them as separate baseline and alignment trajectories. For example, we can distinguish between an archetypal individual who initially aligns to her colleagues and then internalizes this style of communication such that her baseline use also shifts and another archetypal person who aligns to her colleagues but does not change her baseline usage. The former exhibits high correspondence between internalization and self-regulation, whereas the latter demonstrates an ability to decouple them.

5 Analyses

We perform three analyses on this data. First, we examine the qualitative behaviors of pronoun alignment and how they map onto employee outcomes in the data. Second, we show that these qualitative differences in early enculturation are meaningful, with alignment behaviors predicting employment outcome above chance. Lastly, we consider lexical versus referential levels of alignment and show that predictions are improved under the referential formulation, suggesting that alignment is not limited to low-level word-repetition effects.

[6] As η^{base} is the log-odds of each word in a reply being a part of the category c, it is expected to be substantially negative. For example, second person pronouns (*you*), are around 2% of the words in replies, approximately -4 in log-odds space. We follow Gelman et al. (2008)'s recommendation of the Cauchy prior as appropriate for parameter estimation in logistic regression.

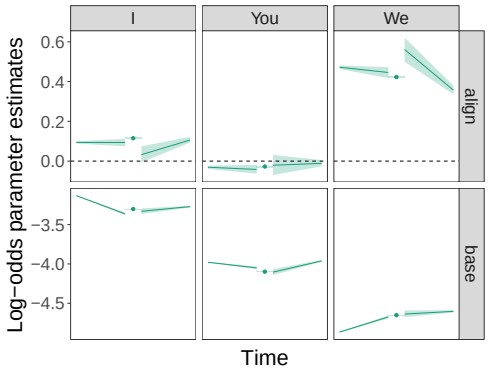


Figure 3: Sawhorse plots showing the dynamics of pronoun alignment behavior across employees. Vertical axis shows log-odds for baseline and alignment. Top row shows estimated alignment, highest for *we* and smallest for *you*. Bottom row shows baseline dynamics, with employees shifting toward the average usage as they enculturate. The shaded region is one standard deviation over parameter samples.

5.1 Analysis 1: Dynamic Qualitative Changes

We begin with descriptive analyses of the behavior of pronouns, which are likely to reflect incorporation into the company. In particular, we look at first-person singular (*I*), first-person plural (*we*), and second-person pronouns (*you*). We expect that increases in *we* usage will occur as the employee is integrated into the group, while *I* and *you* usage will decrease, and want to understand whether these changes manifest on baseline usage (i.e., internalization), alignment (i.e., self-regulation), or both.

Design We divided each employee's emails by calendar month, and separated them into the employee's first six months, their last six months (if an employee left the company within the observation period), and the middle of their tenure. Employees with fewer than twelve months at the company were excluded from this analysis, so that their first and last months did not overlap.

We fit two WHAM models in this analysis. The first aggregated all employees, regardless of employment outcome, to minimize noise; the second separated them by outcome to analyze cultural fit differences.

Outcome-aggregated model We start with the aggregated behavior of all employees, shown in Figure 3. For baselines, we see decreased use of *I*

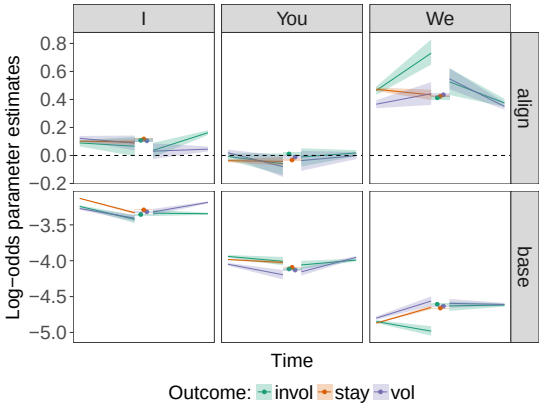


Figure 4: Sawhorse plots split by employment outcome. Mid-tenure points are jittered for improved readability.

and *you* over the first six months, with *we* usage increasing over the same period, confirming the expected result that incorporating into the group is accompanied by more inclusive pronoun usage. Despite the baseline changes, alignment is fairly stable through the first six months. Alignment on first-person singular and second-person pronouns is lower than first-person plural pronouns, likely due to the fact that *I* or *you* have different referents when used by the two conversants, while both conversants could use *we* to refer to the same group. We will consider this referential alignment in more detail in Analysis 3. Since employees with different outcomes have much different experiences over their last six months, we will not discuss them in aggregate, aside from noting the sharp decline in *we* alignment near the end of the employees' tenures.

Outcome-separated model Figure 4 shows outcome-specific trajectories, with green lines showing involuntary leavers (i.e., those who are fired or downsized), blue showing voluntary leavers, and orange showing employees who remained at the company through the final month of the data. The use of *I* and *you* is similar to the aggregates in Figure 3, regardless of group. The last six months of *I* usage show an interesting difference, where involuntary leavers align more on *I* but retain a stable baseline while voluntary leavers retain a stable alignment but increase *I* overall, which is consistent with group separation.

The most compelling result we see here, though, is the changes in *we* usage by different groups of employees. Employees who eventually leave the company involuntarily show signs of more self-regulation than internalization over the first six months, increasing their alignment while decreasing their baseline use (though they return to more similar levels as other employees later in their tenure). Employees who stay at the company, as well as those who later leave voluntarily, show signs of internalization, increasing their baseline usage to the company average, as well as adapting their alignment levels to the mean. This finding suggests that how quickly the employees internalize culturally-standard language use predicts their eventual employment outcome, even if they eventually end up near the average.

5.2 Analysis 2: Predicting Outcomes

This analysis tests the hypothesis that there are meaningful differences in employees' initial enculturation, captured by alignment behaviors. We examine the first six months of communications and attempt to predict whether the employee will leave the company. We find that, even with a simple classifier, alignment behaviors are predictive of employment outcome.

Design We fit the WHAM model to only the first six months of email correspondence for all employees who had at least six months of email. The model estimated the initial level of baseline use (η_{base}) and alignment (η_{align}) for each employee, as well as the slope (α, β) for baseline and alignment over those first six months, over all 11 word categories mentioned in Section 3.

We then created logistic regression classifiers, using the parameter estimates to predict whether an employee would leave the company. We fit separate classifiers for leaving voluntarily or involuntarily. Our results show that early alignment behaviors are better at identifying employees who will leave involuntarily than voluntarily, consistent with Srivastava et al.'s (forthcoming) findings that voluntary leavers are similar to stayers until late in their tenure. We fit separate classifiers using the alignment parameters and the baseline parameters to investigate their relative informativity.

For each model, we report the area under the curve (AUC). This value is estimated from the receiver operating characteristic (ROC) curve, which plots the true positive rate against the false positive rate over different classification thresholds. An AUC of 0.5 represents chance performance. We use balanced, stratified cross-

validation to reduce AUC misestimation due to unbalanced outcome frequencies and high noise (Parker et al., 2007).

Results The left column of Figure 5 shows the results over 10 runs of 10-fold balanced logistic classifiers with stratified cross-validation in R. The alignment-based classifiers are both above chance at predicting that an employee will leave the company, whether involuntarily or voluntarily. The baseline-based classifiers perform worse, especially on voluntary leavers. This finding is consistent with the idea that voluntary leavers resemble stayers (who form the bulk of the employees) until late in their tenure when their cultural fit declines.

We fit a model using both alignment and baseline parameters, but this model yielded an AUC value below the alignment-only classifier. This suggests that where alignment and baseline behaviors are both predictive, they do not provide substantially different predictive power and lead to overfitting. A more sophisticated classifier may overcome these challenges; our goal here was not to achieve maximal classification performance but to test whether alignment provided any useful information about employment outcomes.

5.3 Analysis 3: Types of Alignment

Our final analysis investigates the nature of linguistic alignment: specifically, whether there is an effect of referential alignment beyond that of the more commonly used lexical alignment.

Testing this hypothesis requires a small change to the alignment calculations. Lexical alignment is based on the conditional probability of the replier using a word category c given that the preceding message used that same category c. For referential alignment, we examine the conditional probability of the replier using a word category c_j given that the preceding message used the category c_i, where c_i and c_j are likely to be referentially linked. We also consider cases where c_i is likely to transition to c_j throughout the course of the conversation, such as present tense verbs turning into past tense as the event being described recedes into the past. The pairs of categories that are likely to be referentially or transitionally linked are: (*you, I*); (*we, I*); (*you, we*); (past, present); (present, future); and (certainty, tentativity). We include both directions of these pairs, so this provides approximately the same number of predictor variables for both situations to maximize comparability (12 for the referential alignments, 11 for the lexical). This modification does not change the structure of the WHAM model, but rather changes its C and N counts by reclassifying replies between the baseline or alignment pathways.

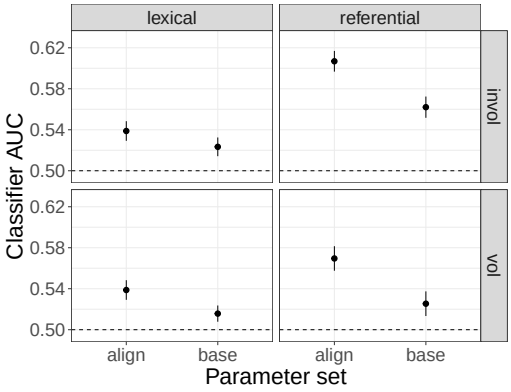


Figure 5: AUC values for 10 runs of 10-fold cross-validated logistic classifiers, with 95% confidence intervals on the mean AUC. Both lexical (left column) and referential (right column) alignment parameters lead to above chance classifier performance, but referential alignment outperforms lexical alignment at predicting both voluntary and involuntary departures.

Results Figure 5 plots the differences in predictive model performance using lexical versus referential alignment parameters. We find that the semantic parameters provide more accurate classification than the lexical both for voluntarily and involuntarily-leaving employees. This suggests that while previous work looking at lexical alignment successfully captures social structure, referential alignment may reflect a deeper and more accurate representation of the social structure. It is unclear if this behavior holds in less formal situations or with weaker organizational structure and shared goals, but these results suggest that the traditional alignment approach of only measuring lexical alignment should be augmented with referential alignment measures for a more complete analysis.

6 Discussion

A key finding from this work is that pronoun usage behaviors in employees' email communication are consistent with social integration into the group; employees use "I" pronouns less and

"we" pronouns more as they integrate. Furthermore, we see the importance of using an alignment measure such as WHAM for distinguishing the base rate and alignment usage of words. Employees who leave the company involuntarily show increased "we" usage through greater alignment, using "we" more when prompted by a colleague, but introducing it less of their own accord. This suggests that these employees do not feel fully integrated into the group, although they are willing to identify as a part of it when a more fully-integrated group member includes them, corresponding to self-regularization over internalization. The fact that these alignment measures alone, without any job productivity or performance metrics, have some predictive capability for employees' leaving the company suggests the potential for support or intervention programs to help high-performing but poorly-integrated employees integrate into the company better.

More generally, the prominence of pronominally-driven communication changes suggest that alignment analyses can provide insight into a range of social integration settings. This may be especially helpful in cases where there is great pressure to integrate smoothly, and people would be likely to adopt a self-regulating approach even if they do not internalize their group membership. Such settings not only include the high-stakes situation of keeping one's job, but of transitioning from high school to college or moving to a new country or region. Maximizing the chances for new members to become comfortable within a group is critical both for spreading useful aspects of the group's existing culture to new members and for integrating new ideas from the new members' knowledge and practices. Alignment-based approaches can be a useful tool in separating effective interventions that cause internalization of the group dynamics from those that lead to more superficial self-regularization changes.

7 Conclusions

This paper described an effort to use directed linguistic alignment as a measure of cultural fit within an organization. We adapted a hierarchical alignment model from previous work to estimate fit within corporate email communications, focusing on changes in language during employees' entry to and exit from the company. Our results showed substantial changes in the use of pronouns, with pronoun patterns varying by employees' outcomes within the company.The use of the first-person plural "we" during an employee's first six months is particularly instructive. Whereas stayers exhibited increased baseline use, indicating internalization, those eventually departing involuntarily were on the one hand decreasingly likely to introduce "we" into conversation, but increasingly responsive to interlocutors' use of the pronoun. While not internalizing a shared identity with their peers, involuntarily departed employees were overly self-regulating in response to its invocation by others.

Quantitatively, rates of usage and alignment in the first six months of employment carried information about whether employees left involuntarily, pointing towards fit within the company culture early on as an indicator of eventual employment outcomes. Finally, we saw ways in which the application of alignment to cultural fit might help to refine ideas about alignment itself: preliminary analysis suggested that referential, rather than lexical, alignment was more predictive of employment outcomes. More broadly, these results suggest ways that quantitative methods can be used to make precise application of concepts like "cultural fit" at scale.

8 Acknowledgments

This work was supported by NSF Grant #1456077; The Garwood Center for Corporate Innovation at the Haas School of Business, University of California, Berkeley; the Stanford Data Science Initiative; and the Stanford Graduate School of Business.

References

Talya N. Bauer, Elizabeth Wolfe Morrison, and Ronda Roberts Callister. 1998. Socialization research: A review and directions for future research. In *Research in Personnel and Human Resources Management*, Emerald Group, Bingley, UK, volume 16, pages 149–214.

Stephen P. Borgatti and José Luis Molina. 2003. Ethical and strategic issues in organizational social network analysis. *The Journal of Applied Behavioral Science* 39(3):337–349.

Richard Y. Bourhis and Howard Giles. 1977. The language of intergroup distinctiveness. In H Giles, editor, *Language, Ethnicity, and Intergroup Relations*, Academic Press, London, pages 119–135.

David B. Buller and R. Kelly Aune. 1988. The effects of vocalics and nonverbal sensitivity on compliance: A speech accommodation theory explanation. *Human Communication Research* 14:301–32.

Cindy Chung and James W. Pennebaker. 2007. The psychological functions of function words. In K Fiedler, editor, *Social communication*, Psychology Press, New York, chapter 12, pages 343–359.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of the 21st international conference on World Wide Web*. page 699. https://doi.org/10.1145/2187836.2187931.

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. No country for old members: user lifecycle and linguistic change in online communities. In *Proceedings of the 22nd International Conference on World Wide Web*. pages 307–318.

Gabriel Doyle and Michael C. Frank. 2016. Investigating the sources of linguistic alignment in conversation. In *Proceedings of ACL*.

Gabriel Doyle, Dan Yurovsky, and Michael C. Frank. 2016. A robust framework for estimating linguistic alignment in Twitter conversations. In *Proceeedings of WWW*.

R. E. Feldman. 1968. Response to compatriots and foriegners who seek assistance. *Journal of Personality and Social Psychology* 10:202–14.

Riccardo Fusaroli, Bahador Bahrami, Karsten Olsen, Andreas Roepstorff, Geraint Rees, Chris Frith, and Kristian Tylén. 2012. Coming to Terms: Quantifying the Benefits of Linguistic Coordination. *Psychological Science* 23(8):931–939. https://doi.org/10.1177/0956797612436816.

Andrew Gelman, Aleks Jakulin, Maria Grazia Pittau, and Yu-Sung Su. 2008. A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics* .

Howard Giles, Nikolas Coupland, and Justine Coupland. 1991. Accommodation theory: Communication, context, and consequences. In Howard Giles, Justine Coupland, and Nikolas Coupland, editors, *Contexts of accommodation: Developments in applied sociolinguistics*, Cambridge University Press, Cambridge.

Augusto Gnisci. 2005. Sequential strategies of accommodation: A new method in courtroom. *British Journal of Social Psychology* 44(4):621–643.

Amir Goldberg, Sameer B. Srivastava, V. Govind Manian, and Christopher Potts. 2016. Fitting in or standing out? The tradeoffs of structural and cultural embeddedness. *American Sociological Review* .

Molly E. Ireland, Richard B. Slatcher, Paul W. Eastwick, Lauren E. Scissors, Eli J. Finkel, and James W. Pennebaker. 2011. Language style matching predicts relationship initiation and stability. *Psychological Science* 22:39–44. https://doi.org/10.1177/0956797610392928.

Kate G. Niederhoffer and James W. Pennebaker. 2002. Linguistic style matching in social interaction. *Journal of Language and Social Psychology* 21(4):337–360. http://jls.sagepub.com/content/21/4/337.short.

Charles A. O'Reilly, Jennifer Chatman, and David F. Caldwell. 1991. People and organizational culture: a profile comparison approach to assessing person-organization fit. *Academy of Management Journal* 34(3):487–516.

Brian J Parker, Simon Günter, and Justin Bedo. 2007. Stratification bias in low signal microarray studies. *BMC bioinformatics* 8(1):1.

James W. Pennebaker, Cindy K. Chung, Molly Ireland, Amy Gonzalez, and Roger J. Booth. 2007. The development and psychometric properties of liwc2007. Technical report, LIWC.net.

Martin J. Pickering and Simon Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and brain sciences* 27(2):169–190. https://doi.org/10.1017/S0140525X04000056.

Sameer B. Srivastava, Amir Goldberg, V. Govind Manian, and Christopher Potts. forthcoming. Enculturation trajectories: Language, cultural adaptation, and individual outcomes in organizations. *Management Science* .

Harry C. Triandis. 1960. Cognitive similarity and communication in a dyad. *Human Relations* 13:175–183.

Dan Yurovsky, Gabriel Doyle, and Michael C. Frank. 2016. Linguistic input is tuned to children's developmental level. In *Proceedings of the 38th Annual Meeting of the Cognitive Science Society*.

Representations of language in a model of visually grounded speech signal

Grzegorz Chrupała
Tilburg University
g.chrupala@uvt.nl

Lieke Gelderloos
Tilburg University
l.j.gelderloos@uvt.nl

Afra Alishahi
Tilburg University
a.alishahi@uvt.nl

Abstract

We present a visually grounded model of speech perception which projects spoken utterances and images to a joint semantic space. We use a multi-layer recurrent highway network to model the temporal nature of spoken speech, and show that it learns to extract both form and meaning-based linguistic knowledge from the input signal. We carry out an in-depth analysis of the representations used by different components of the trained model and show that encoding of semantic aspects tends to become richer as we go up the hierarchy of layers, whereas encoding of form-related aspects of the language input tends to initially increase and then plateau or decrease.

1 Introduction

Speech recognition is one of the success stories of language technology. It works remarkably well in a range of practical settings. However, this success relies on the use of very heavy supervision where the machine is fed thousands of hours of painstakingly transcribed audio speech signal. Humans are able to learn to recognize and understand speech from notably weaker and noisier supervision: they manage to learn to extract structure and meaning from speech by simply being exposed to utterances situated and grounded in their daily sensory experience. Modeling and emulating this remarkable skill has been the goal of numerous studies; however in the overwhelming majority of cases researchers used severely simplified settings where either the language input or the extralinguistic sensory input, or both, are small scale and symbolically represented. Section 2 provides a brief overview of this research.

More recently several lines of work have moved towards more realistic inputs while modeling or emulating language acquisition in a grounded setting. Gelderloos and Chrupała (2016) use the image captioning dataset MS COCO (Lin et al., 2014) to mimic the setting of grounded language learning: the sensory input consists of images of natural scenes, while the language input are phonetically transcribed descriptions of these scenes. The use of such moderately large and low-level data allows the authors to train a multi-layer recurrent neural network model, and to explore the nature and localization of the emerging hierarchy of linguistic representations learned in the process. Furthermore, in a series of recent studies Harwath and Glass (2015); Harwath et al. (2016); Harwath and Glass (2017) use image captioning datasets to model learning to understand spoken language from visual context with convolutional neural network models. Finally, there is a small but growing body of work dedicated to elucidating the nature of representations learned by neural networks from language data (see Section 2.2 for a brief overview). In the current work we build on these three strands of research and contribute the following advances:

- We use a multi-layer gated recurrent neural network to properly model the temporal nature of speech signal and substantially improve performance compared to the convolutional architecture from Harwath and Glass (2015);
- We carry out an in-depth analysis of the representations used by different components of the trained model and correlate them to representations learned by a text-based model and to human patterns of judgment on linguistic stimuli. This analysis is especially novel for a model with speech signal as input.

The general pattern of findings in our analysis is

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 613–622
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1057

as follows: The model learns to extract from the acoustic input both form-related and semantics-related information, and encodes it in the activations of the hidden layers. Encoding of semantic aspects tends to become richer as we go up the hierarchy of layers. Meanwhile, encoding of form-related aspects of the language input, such as utterance length or the presence of specific words, tends to initially increase and then decay.

We release the code for our models and analyses as open source, available at https://github.com/gchrupala/visually-grounded-speech. We also release a dataset of synthetically spoken image captions based on MS COCO, available at https://doi.org/10.5281/zenodo.400926.

2 Related work

Children learn to recognize and assign meaning to words from continuous perceptual data in extremely noisy context. While there have been many computational studies of human word meaning acquisition, they typically make strong simplifying assumptions about the nature of the input. Often language input is given in the form of word symbols, and the context consists of a set of symbols representing possible referents (e.g. Siskind, 1996; Frank et al., 2007; Fazly et al., 2010). In contrast, several studies presented models that learn from sensory rather than symbolic input, which is rich with regards to the signal itself, but very limited in scale and variation (e.g. Roy and Pentland, 2002; Yu and Ballard, 2004; Lazaridou et al., 2016).

2.1 Multimodal language acquisition

Chrupała et al. (2015) introduce a model that learns to predict the visual context from image captions. The model is trained on image-caption pairs from MSCOCO (Lin et al., 2014), capturing both rich visual input as well as larger scale input, but the language input still consists of word symbols. Gelderloos and Chrupała (2016) propose a similar architecture that instead takes phoneme-level transcriptions as language input, thereby incorporating the word segmentation problem into the learning task. In this work, we introduce an architecture that learns from continuous speech and images directly.

This work is related to research on visual grounding of language. The field is large and growing, with most work dedicated to the grounding of written text, particularly in image captioning tasks (see Bernardi et al. (2016) for an overview). However, learning to ground language to visual information is also interesting from an automatic speech recognition point of view. Potentially, ASR systems could be trained from naturally co-occurring visual context information, without the need for extensive manual annotation – a particularly promising prospect for speech recognition in low-resource languages. There have been several attempts along these lines. Synnaeve et al. (2014) present a method of learning to recognize spoken words in isolation from co-occurrence with image fragments. Harwath and Glass (2015) present a model that learns to map pre-segmented spoken words in sequence to aspects of the visual context, while in Harwath and Glass (2017) the model also learns to recognize words in the unsegmented signal.

Most closely related to our work is that of Harwath et al. (2016), as it presents an architecture that learns to project images and unsegmented spoken captions to the same embedding space. The sentence representation is obtained by feeding the spectrogram to a convolutional network. The architecture is trained on crowd-sourced spoken captions for images from the Places dataset (Zhou et al., 2014), and evaluated on image search and caption retrieval. Unfortunately this dataset is not currently available and we were thus unable to directly compare the performance of our model to Harwath et al. (2016). We do compare to Harwath and Glass (2015) which was tested on a public dataset. We make different architectural choices, as our models are based on recurrent highway networks (Zilly et al., 2016). As in human cognition, speech is processed incrementally. This also allows our architecture to integrate information sequentially from speech of arbitrary duration.

2.2 Analysis of neural representations

While analysis of neural methods in NLP is often limited to evaluation of the performance on the training task, recently methods have been introduced to *peek inside the black box* and explore what it is that enables the model to perform the task. One approach is to look at the contribution of specific parts of the input, or specific units in the model, to final representations or decisions. Kádár et al. (2016) propose *omission scores*, a method to estimate the contribution of input tokens to the fi-

nal representation by removing them from the input and comparing the resulting representations to the ones generated by the original input. In a similar approach, Li et al. (2016) study the contribution of individual input tokens as well as hidden units and word embedding dimensions by erasing them from the representation and analyzing how this affects the model.

Miao et al. (2016) and Tang et al. (2016) use visualization techniques for fine-grained analysis of GRU and LSTM models for ASR. Visualization of input and forget gate states allows Miao et al. (2016) to make informed adaptations to gated recurrent architectures, resulting in more efficiently trainable models. Tang et al. (2016) visualize qualitative differences between LSTM- and GRU-based architectures, regarding the encoding of information, as well as how it is processed through time.

We specifically study linguistic properties of the information encoded in the trained model. Adi et al. (2016) introduce prediction tasks to analyze information encoded in sentence embeddings about word order, sentence length, and the presence of individual words. We use related techniques to explore encoding of aspects of form and meaning within components of our stacked architecture.

3 Models

We use a multi-layer, gated recurrent neural network (RHN) to model the temporal nature of speech signal. Recurrent neural networks are designed for modeling sequential data, and gated variants (GRUs, LSTMs) are widely used with speech and text in both cognitive modeling and engineering contexts. RHNs are a simple generalization of GRU networks such that the transform between time points can consist of several steps.

Our multimodal model projects spoken utterances and images to a joint semantic space. The idea of projecting different modalities to a shared semantic space via a pair of encoders has been used in work on language and vision (among them Vendrov et al. (2015)). The core idea is to encourage inputs representing the same meaning in different modalities to end up nearby, while maintaining a distance from unrelated inputs.

The model consists of two parts: an utterance encoder, and an image encoder. The utterance encoder starts from MFCC speech features, while the image encoder starts from features extracted with a VGG-16 pre-trained on ImageNet. Our loss function attempts to make the cosine distance between encodings of matching utterances and images greater than the distance between encodings of mismatching utterance/image pairs, by a margin:

$$\sum_{u,i}\left(\sum_{u'}\max[0,\alpha+d(u,i)-d(u',i)] + \sum_{i'}\max[0,\alpha+d(u,i)-d(u,i')]\right) \quad (1)$$

where $d(u,i)$ is the cosine distance between the encoded utterance u and encoded image i. Here (u,i) is the matching utterance-image pair, u' ranges over utterances not describing i and i' ranges over images not described by u.

The image encoder enc_i is a simple linear projection, followed by normalization to unit L2 norm:

$$\mathrm{enc}_i(\mathbf{i}) = \mathrm{unit}(\mathbf{A}\mathbf{i}+b) \quad (2)$$

where $\mathrm{unit}(x) = \frac{x}{(x^T x)^{0.5}}$ and with $(\mathbf{A}, b)$ as learned parameters. The utterance encoder enc_u consists of a 1-dimensional convolutional layer of length s, size d and stride z, whose output feeds into a Recurrent Highway Network with k layers and L microsteps, whose output in turn goes through an attention-like lookback operator, and finally L2 normalization:

$$\mathrm{enc}_u(\mathbf{u}) = \mathrm{unit}(\mathrm{Attn}(\mathrm{RHN}_{\mathrm{k,L}}(\mathrm{Conv}_{\mathrm{s,d,z}}(\mathbf{u})))) \quad (3)$$

The main function of the convolutional layer $\mathrm{Conv}_{s,d,z}$ is to subsample the input along the temporal dimension. We use a 1-dimensional convolution with full border mode padding. The attention operator simply computes a weighted sum of the RHN activation at all timesteps:

$$\mathrm{Attn}(\mathbf{x}) = \sum_t \alpha_t \mathbf{x}_t \quad (4)$$

where the weights α_t are determined by learned parameters $\mathbf{U}$ and $\mathbf{W}$, and passed through the timewise softmax function:

$$\alpha_t = \frac{\exp(\mathbf{U}\tanh(\mathbf{W}\mathbf{x}_t))}{\sum_{t'}\exp(\mathbf{U}\tanh(\mathbf{W}\mathbf{x}_{t'}))} \quad (5)$$

The main component of the utterance encoder is a recurrent network, specifically a Recurrent Highway Network (Zilly et al., 2016). The idea behind

RHN is to increase the depth of the transform between timesteps, or the recurrence depth. Otherwise they are a type of gated recurrent networks. The transition from timestep $t-1$ to t is then defined as:

$$\mathrm{rhn}(\mathbf{x}_t, \mathbf{s}_{t-1}^{(L)}) = \mathbf{s}_t^{(L)} \tag{6}$$

where $\mathbf{x}_t$ stands for input at time t, and $\mathbf{s}_t^{(l)}$ denotes the state at time t at recurrence layer l, with L being the top layer of recurrence. Furthermore,

$$\mathbf{s}_t^{(l)} = \mathbf{h}_t^{(l)} \odot \mathbf{t}_t^{(l)} + \mathbf{s}_t^{(l-1)} \odot \left(\mathbf{1} - \mathbf{t}_t^{(l)}\right) \tag{7}$$

where $\odot$ is elementwise multiplication, and

$$\mathbf{h}_t^{(l)} = \tanh\left(\mathbb{I}[l=1]\mathbf{W}_H\mathbf{x}_t + \mathbf{U}_{H_l}\mathbf{s}_t^{(l-1)}\right) \tag{8}$$

$$\mathbf{t}_t^{(l)} = \sigma\left(\mathbb{I}[l=1]\mathbf{W}_T\mathbf{x}_t + \mathbf{U}_{T_l}\mathbf{s}^{(l-1)}\right) \tag{9}$$

Here $\mathbb{I}$ is the indicator function: input is only included in the computation for the first layer of recurrence $l=1$. By applying the rhn function repeatedly, an RHN layer maps a sequence of inputs to a sequence of states:

$$\begin{aligned}&\mathrm{RHN}(\mathbf{X}, \mathbf{s}_0)\\ &\quad = \mathrm{rhn}(\mathbf{x}_n, \ldots, \mathrm{rhn}(\mathbf{x}_2, \mathrm{rhn}(\mathbf{x}_1, \mathbf{s}_0^{(L)})))\end{aligned} \tag{10}$$

Two or more RHN layers can be composed into a stack:

$$\mathrm{RHN}_2(\mathrm{RHN}_1(\mathbf{X}, \mathbf{s}_{\mathbf{1}_0}^{(L)}), \mathbf{s}_{\mathbf{2}_0}^{(L)}), \tag{11}$$

where $\mathbf{s}_{\mathbf{n}_t}^{(l)}$ stands for the state vector of layer n of the stack, at layer l of recurrence, at time t. In our version of the Stacked RHN architecture we use *residualized* layers:

$$\mathrm{RHN}_{\mathrm{res}}(\mathbf{X}, \mathbf{s}_0) = \mathrm{RHN}(\mathbf{X}, \mathbf{s}_0) + \mathbf{X} \tag{12}$$

This formulation tends to ease optimization in multi-layer models (cf. He et al., 2015; Oord et al., 2016).

In addition to the speech model described above, we also define a comparable text model. As it takes a sequence of words as input, we replace the convolutional layer with a word embedding lookup table. We found the text model did not benefit from the use of the attention mechanism, and thus the sentence embedding is simply the L2-normalized activation vector of the topmost layer, at the last timestep.

4 Experiments

Our main goal is to analyze the emerging representations from different components of the model and to examine the linguistic knowledge they encode. For this purpose, we employ a number of tasks that cover the spectrum from fully form-based to fully semantic.

In Section 4.2 we assess the effectiveness of our architecture by evaluating it on the task of ranking images given an utterance. Sections 4.3 to 4.6 present our analyses. In Sections 4.3 and 4.4 we define auxiliary tasks to investigate to what extent the network encodes information about the surface form of an utterance from the speech input. In Section 4.5 and 4.6 we focus on where semantic information is encoded in the model. In the analyses, we use the following features:

Utterance embeddings: the weighted sum of the unit activations on the last layer, as calculated by Equation (3).

Average unit activations: hidden layer activations averaged over time and L2-normalized for each hidden layer.

Average input vectors: the MFCC vectors averaged over time. We use this feature to examine how much information can be extracted from the input signal only.

4.1 Data

For the experiments reported in the remainder of the paper we use two datasets of images with spoken captions.

4.1.1 Flickr8K

The Flickr8k Audio Caption Corpus was constructed by having crowdsource workers read aloud the captions in the original Flickr8K corpus (Hodosh et al., 2013). For details of the data collection procedure refer to Harwath and Glass (2015). The datasets consist of 8,000 images, each image with five descriptions. One thousand images are held out for validation, and another one thousand for the final test set. We use the splits provided by (Karpathy and Fei-Fei, 2015). The image features come from the final fully connect layer of VGG-16 (Simonyan and Zisserman, 2014) pre-trained on Imagenet (Russakovsky et al., 2014).

We generate the input signal as follows: we extract 12-dimensional mel-frequency cepstral coefficients (MFCC) plus log of the total energy. We

then compute and add first order and second order differences (deltas) for a total of 37 dimensions. We use 25 milisecond windows, sampled every 10 miliseconds.[1]

4.1.2 Synthetically spoken COCO

We generated synthetic speech for the captions in the MS COCO dataset (Lin et al., 2014) via the Google Text-to-Speech API.[2] The audio and the corresponding MFCC features are released as Chrupała et al. (2017)[3]. This TTS system we used produces high-quality realistic-sounding speech. It is nevertheless much simpler than real human speech as it uses a single voice, and lacks tempo variation or ambient noise. The data consists of over 300,000 images, each with five spoken captions. Five thousand images each are held out for validation and test. We use the splits and image features provided by Vendrov et al. (2015).[4] The image features also come from the VGG-16 network, but are averages of feature vectors for ten crops of each image. For the MS COCO captions we extracted only plain MFCC and total energy features, and did not add deltas in order to keep the amount of computation manageable given the size of the dataset.

4.2 Image retrieval

We evaluate our model on the task of ranking images given a spoken utterance, such that highly ranked images contain scenes described by the utterance. The performance on this task on validation data is also used to choose the best variant of the model architecture and to tune the hyperparameters. We compare the speech models to models trained on written sentences split into words. The best settings found for the four models were the following:

Flickr8K Text RHN 300-dimensional word embeddings, 1 hidden layer with 1024 dimensions, 1 microstep, initial learning rate 0.001.

Flick8K Speech RHN convolutional layer with length 6, size 64, stride 2, 4 hidden layers with 1024 dimensions, 2 microsteps, attention MLP with 128 hidden units, initial learning rate 0.0002

COCO Text RHN 300-dimensional word embeddings, 1 hidden layer with 1024 dimensions, 1 microstep, initial learning rate 0.001

COCO Speech RHN convolutional layer with length 6, size 64, stride 3, 5 hidden layers with 512 dimensions, 2 microsteps, attention MLP with 512 hidden units, initial learning rate 0.0002

All models were optimized with Adam (Kingma and Ba, 2014) with early stopping: we kept the parameters for the epoch which showed the best recall@10 on validation data.

Model	R@1	R@5	R@10	$\tilde{r}$
Speech $\text{RHN}_{4,2}$	0.055	0.163	0.253	48
Spectr. CNN	-	-	0.179	-
Text $\text{RHN}_{1,1}$	0.127	0.364	0.494	11

Table 1: Image retrieval performance on Flickr8K. R@N stands for recall at N; $\tilde{r}$ stands for median rank of the correct image.

Model	R@1	R@5	R@10	$\tilde{r}$
Speech $\text{RHN}_{5,2}$	0.111	0.310	0.444	13
Text $\text{RHN}_{1,1}$	0.169	0.421	0.565	8

Table 2: Image retrieval performance on MS COCO. R@N stands for recall at N; $\tilde{r}$ stands for median rank of the correct image.

Table 1 shows the results for the human speech from the Flickr8K dataset. The Speech RHN model scores substantially higher than model of Harwath and Glass (2015) on the same data. However the large gap between its perfomance and the scores of the text model suggests that Flickr8K is rather small for the speech task. In Table 2 we present the results on the dataset of synthetic speech from MS COCO. Here the text model is still better, but the gap is much smaller than for Flickr8K. We attribute this to the much larger size of dataset, and to the less noisy and less variable synthetic speech.

While the MS COCO text model is overall better than the speech model, there are cases where it outperforms the text model. We listed the top hundred cases where the ratio of the ranks of the correct image according to the two models was the smallest, as well as another hundred cases where it was the largest. Manual inspection did not turn

[1] We noticed that for a number of utterances the audio signal was very long: on inspection it turned out that most of these involved failure to switch off the microphone on the part of the workers, and the audio contained ambient noise or unrelated speech. We thus trucated all audio for this dataset at 10,000 miliseconds.

[2] Available at https://github.com/pndurette/gTTS.

[3] Available at https://doi.org/10.5281/zenodo.400926.

[4] See https://github.com/ivendrov/order-embedding.

up any obvious patterns for the cases of text being better than speech. For the cases where speech outperformed text, two patterns stood out: (i) sentences with spelling mistakes, (ii) unusually long sentences. For example for the sentence *a yellow*

Figure 1: Images returned for utterance *a yellow and white birtd is in flight* by the text (left) and speech (right) models.

and white birtd is in flight the text model misses the misspelled word *birtd* and returns an irrelevant image, while the speech model seems robust to some degree of variation in pronunciation and returns the target image at rank 1 (see Figure 1). In an attempt to quantify this effect we counted the number of unique words with training set frequencies below 5 in the top 100 utterances with lowest and highest rank ratio: for the utterances where text was better there were 16 such words; for utterances where speech was better there were 28, among them misspellings such as *streeet*, *scears* (for *skiers*), *contryside, scull, birtd, devise*.

The distribution of utterance lengths in Figure 2 confirms pattern (ii): the set of 100 sentences where speech beats text by a large margin are longer on average and there are extremely long outliers among them. One of them is the 36-word-

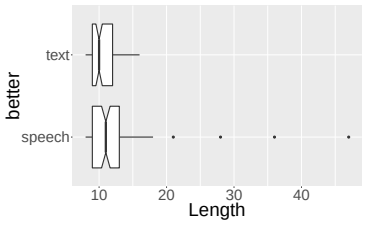


Figure 2: Length distribution for sentences where one model performs much better than the other.

long utterance depicted in Figure 3, with ranks 470 and 2 for text and speech respectively. We suspect that the speech model's attention mechanism enables it to cherry pick key fragments of such monster utterances, while the text model lacking this mechanism may struggle. Figure 3 shows the plot of the attention weights for this utterance from the speech model.

4.3 Predicting utterance length

Our first auxiliary task is to predict the length of the utterance, using the features explained at the beginning of Section 4. Since the length of an utterance directly corresponds to how long it takes to articulate, we also use the number of time steps[5] as a feature and expect it to provide the upper bound for our task, especially for synthetic speech. We use a Ridge Regression model for predicting utterance length using each set of features. The model is trained on 80% of the sentences in the validation set, and tested on the remaining 20%. For all features regularization penalty $\alpha = 1.0$ gave the best results.

Figure 4 shows the results for this task on human speech from Flickr8K and synthetic speech from COCO. With the exception of the average input vectors for Flickr8K, all features can explain a high proportion of variance in the predicted utterance length. The pattern observed for the two datasets is slightly different: due to the systematic conversion of words to synthetic speech in COCO, using the number of time steps for this dataset yields the highest R^2. However, this feature is not as informative for predicting the utterance length in Flickr8K due to noise and variation in human speech, and is in fact outperformed by some of the features extracted from the model. Also, the input vectors from COCO are much more informative than Flickr8K due to larger quantity and simpler structure of the speech signal. However, in both datasets the best (non-ceiling) performance is obtained by using average unit activations from the hidden layers (layer 2 for COCO, and layers 3 and 4 for Flickr8K). These features outperform utterance embeddings, which are optimized according to the visual grounding objective of the model and most probably learn to ignore the superficial characteristics of the utterance that do not contribute to matching the corresponding image.

Note that the performance on COCO plateaus after the second layer, which might suggest that form-based knowledge is learned by lower layers. Since Flickr8K is much smaller in size, the stabilising happens later in layer 3.

[5]This is approximately $\frac{\text{duration in milliseconds}}{10 \times \text{stride}}$.

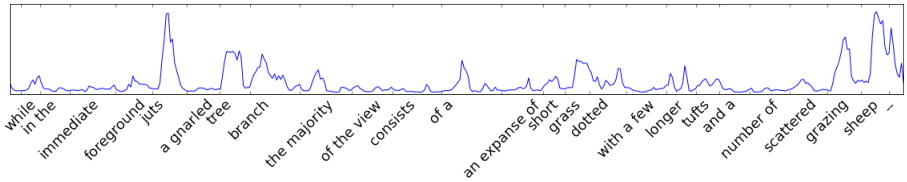


Figure 3: Attention weight distribution for a long utterance.

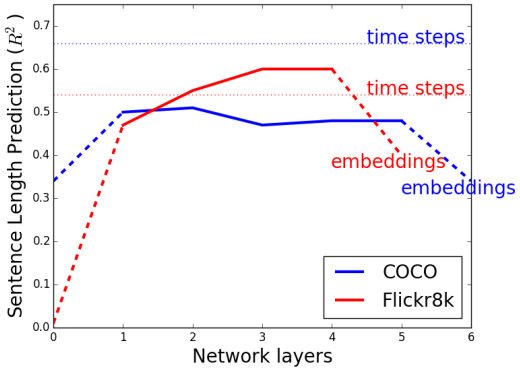


Figure 4: R^2 values for predicting utterance length for Flickr8K and COCO. Layers 1–5 represent (normalized) average unit activation, whereas the first (#0) and last point represent average input vectors and utterance embeddings, respectively.

4.4 Predicting word presence

Results from the previous experiment suggest that our model acquires information about higher level building blocks (words) in the continuous speech signal. Here we explore whether it can detect the presence or absence of individual words in an utterance. We formulate detecting a word in an utterance as a binary classification task, for which we use a multi-layer perceptron with a single hidden layer of size 1024, optimized by Adam. The input to the model is a concatenation of the feature vector representing an utterance and the one representing a target word. We again use utterance embeddings, average unit activations on each layer, and average input vectors as features, and represent each target word as a vector of MFCC features extracted from the audio signal synthetically produced for that word.

For each utterance in the validation set, we randomly pick one positive and one negative target (i.e., one word that does and one that does not appear in the utterance) that is not a stop word. To balance the probability of a word being positive or negative, we use each positive target as a negative target for another utterance in the validation set. The MLP model is trained on the positive and negative examples corresponding to 80% of the utterances in the validation set of each dataset, and evaluated on the remaining 20%.

Figure 5 shows the mean accuracy of the MLP on Flickr8K and COCO. All results using features extracted from the model are above chance (0.5), with the average unit activations of the hidden layers yielding the best results (0.65 for Flickr8K on layer 3, and 0.79 for COCO on layer 4). These numbers show that the speech model infers reliable information about word-level blocks from the low-level audio features it receives as input. The observed trend is similar to the previous task: average unit activations on the higher-level hidden layers are more informative for this task than the utterance embeddings, but the performance plateaus before the topmost layer.

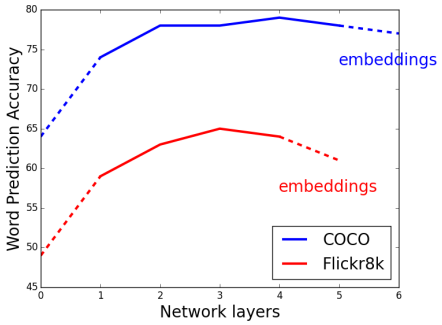


Figure 5: Mean accuracy values for predicting the presence of a word in an utterance for Flickr8K and COCO. Layers 1–5 represent the (normalized) average unit activations, whereas the first (#0) and last point represent average input vectors and utterance embeddings, respectively.

4.5 Sentence similarity

Next we explore to what extent the model's representations correspond to those of humans. We employ the Sentences Involving Compositional Knowledge (SICK) dataset (Marelli et al., 2014). SICK consists of image descriptions taken from

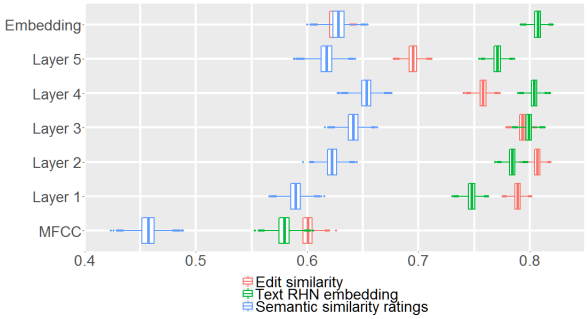


Figure 6: Pearson's r of cosine similarities of averaged input MFCCs and COCO Speech RHN hidden layer activation vectors and embeddings of sentence pairs with relatedness scores from SICK, cosine similarity of COCO Text RHN embeddings, and edit similarity.

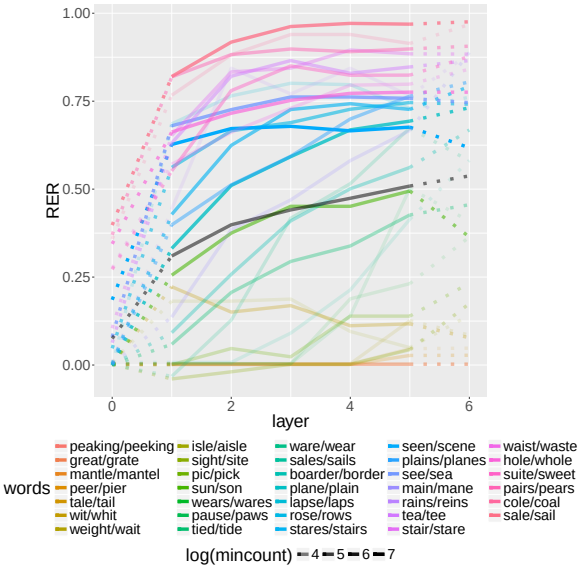


Figure 7: Disambiguation performance per layer. Points #0 and #6 (connected via dotted lines) represent the input vectors and utterance embeddings, respectively. The black line shows the overall mean RER.

Flickr8K and video captions from the SemEval 2012 STS MSRVideo Description data set (STS) (Agirre et al., 2012). Captions were paired at random, as well as modified to obtain semantically similar and contrasting counterparts, and the resulting pairs were rated for semantic similarity.

For all sentence pairs in SICK, we generate synthetic spoken sentences and feed them to the COCO Speech RHN, and calculate the cosine similarity between the averaged MFCC input vectors, the averaged hidden layer activation vectors, and the sentence embeddings. Z-score transformation was applied before calculating the cosine similarities. We then correlate these cosine similarities with

- semantic relatedness according to human ratings
- cosine similarities according to z-score transformed embeddings from COCO Text RHN
- *edit similarities*, a measure of how similar the sentences are in form, specifically, 1−normalized Levenshtein distance over character sequences

Figure 6 shows a boxplot over 10,000 bootstrap samples for all correlations. We observe that (i) correlation with edit similarity initially increases, then decreases; (ii) correlation with human relatedness scores and text model embeddings increases until layer 4, but decreases for hidden layer 5. The initially increasing and then decreasing correlation with edit similarity is consistent with the findings that information about form is encoded by lower layers. The overall growing correlation with both human semantic similarity ratings and the COCO Text RHN indicate that higher layers learn to represent semantic knowledge. We were somewhat surprised by the pattern for the correlation with human ratings and the Text model similarities which drops for layer 5. We suspect it may be caused by the model at this point in the layer hierarchy being strongly tuned to the specifics of the COCO dataset. To test this, we checked the correlations with COCO Text embeddings on validation sentences from the COCO dataset instead of SICK. These increased monotonically, in support of our conjecture.

4.6 Homonym disambiguation

Next we simulate the task of distinguishing between pairs of homonyms, i.e. words with the same acoustic form but different meaning. We group the words in the union of the training and validation data of the COCO dataset by their phonetic transcription. We then pick pairs of words which have the same pronunciation but different spelling, for example *suite/sweet*. We impose the following conditions: (a) both forms appear more than 20 times, (b) the two forms have different meaning (i.e. they are not simply variant spellings like *theater/theatre*), (c) neither form is a function word, and (d) the more frequent form constitutes less than 95% of the occurrences. This

gives us 34 word pairs. For each pair we generate a binary classification task by taking all the utterances where either form appears, using average input vectors, utterance embeddings, and average unit activations as features. Instances for all feature sets are normalized to unit L2 norm.

For each task and feature set we run stratified 10-fold cross validation using Logistic Regression to predict which of the two words the utterance contains. Figure 7 shows, for each pair, the relative error reduction of each feature set with respect to the majority baseline. There is substantial variation across word pairs, but overall the task becomes easier as the features come from higher layers in the network. Some forms can be disambiguated with very high accuracy (e.g. *sale/sail, cole/coal, pairs/pears*), while some others cannot be distinguished at all (*peaking/peeking, great/grate, mantle/mantel*). We examined the sentences containing the failing forms, and found out that almost all occurrences of *peaking* and *mantle* were misspellings of *peeking* and *mantel*, which explains the impossibility of disambiguating these cases.

5 Conclusion

We present a multi-layer recurrent highway network model of language acquisition from visually grounded speech signal. Through detailed analysis we uncover how information in the input signal is transformed as it flows through the network: formal aspects of language such as word identities that not directly present in the input are discovered and encoded low in the layer hierarchy, while semantic information is most strongly expressed in the topmost layers.

Going forward we would like to compare the representations learned by our model to the brain activity of people listening to speech in order to determine to what extent the patterns we found correspond to localized processing in the human cortex. This will hopefully lead to a better understanding of language learning and processing by both artificial and neural networks.

Acknowledgements

We would like to thank David Harwath for making the Flickr8k Audio Caption Corpus publicly available.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207* .

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, volume 2, pages 385–393.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *arXiv preprint arXiv:1601.03896* .

Grzegorz Chrupała, Akos Kádár, and Afra Alishahi. 2015. Learning language through pictures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Grzegorz Chrupała, Lieke Gelderloos, and Afra Alishahi. 2017. Synthetically spoken COCO. https://doi.org/10.5281/zenodo.400926.

Afsaneh Fazly, Afra Alishahi, and Suzanne Stevenson. 2010. A probabilistic computational model of cross-situational word learning. *Cognitive Science: A Multidisciplinary Journal* 34(6):1017–1063.

Michael C. Frank, Noah D. Goodman, and Joshua B. Tenenbaum. 2007. A Bayesian framework for cross-situational word-learning. In *Advances in Neural Information Processing Systems*. volume 20.

Lieke Gelderloos and Grzegorz Chrupała. 2016. From phonemes to images: levels of representation in a recurrent neural model of visually-grounded language learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

David Harwath and James Glass. 2015. Deep multimodal semantic embeddings for speech and images. In *IEEE Automatic Speech Recognition and Understanding Workshop*.

David Harwath and James R Glass. 2017. Learning word-like units from joint audio-visual analysis. *arXiv preprint arXiv:1701.07481* .

David Harwath, Antonio Torralba, and James Glass. 2016. Unsupervised learning of spoken language with visual context. In *Advances in Neural Information Processing Systems*. pages 1858–1866.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv:1512.03385* .

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research* 47:853–899.

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2016. Representation of linguistic form and function in recurrent neural networks. *CoRR* abs/1602.08952.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3128–3137.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Angeliki Lazaridou, Grzegorz Chrupała, Raquel Fernández, and Marco Baroni. 2016. Multimodal semantic learning from child-directed input. In *The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220* .

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*, Springer, pages 740–755.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*. pages 216–223.

Yajie Miao, Jinyu Li, Yongqiang Wang, Shi-Xiong Zhang, and Yifan Gong. 2016. Simplifying long short-term memory acoustic models for fast training and decoding. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pages 2284–2288.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759* .

Deb K Roy and Alex P Pentland. 2002. Learning words from sights and sounds: a computational model. *Cognitive Science* 26(1):113 – 146.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2014. ImageNet Large Scale Visual Recognition Challenge.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

Jeffrey M. Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition* 61(1-2):39–91.

Gabriel Synnaeve, Maarten Versteegh, and Emmanuel Dupoux. 2014. Learning words from images and speech. In *NIPS Workshop on Learning Semantics, Montreal, Canada*.

Zhiyuan Tang, Ying Shi, Dong Wang, Yang Feng, and Shiyue Zhang. 2016. Memory visualization for gated recurrent neural networks in speech recognition. *arXiv preprint arXiv:1609.08789* .

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361* .

Chen Yu and Dana H Ballard. 2004. A multimodal learning interface for grounding spoken language in sensory perceptions. *ACM Transactions on Applied Perception (TAP)* 1(1):57–80.

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. 2014. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 487–495.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent highway networks. *arXiv preprint arXiv:1607.03474* .

Spectral Analysis of Information Density in Dialogue Predicts Collaborative Task Performance

Yang Xu and **David Reitter**
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802, USA
yang.xu@psu.edu, reitter@psu.edu

Abstract

We propose a perspective on dialogue that focuses on relative information contributions of conversation partners as a key to successful communication. We predict the success of collaborative task in English and Danish corpora of task-oriented dialogue. Two features are extracted from the frequency domain representations of the lexical entropy series of each interlocutor, *power spectrum overlap* (PSO) and *relative phase* (RP). We find that PSO is a negative predictor of task success, while RP is a positive one. An SVM with these features significantly improved on previous task success prediction models. Our findings suggest that the strategic distribution of information density between interlocutors is relevant to task success.

1 Introduction

What factors affect whether information is conveyed effectively and reliably in conversations? Several theoretical frameworks have emerged that model dialogical behavior at different granularity levels. Can we use them to measure communicative effectiveness?

Grounding theory (Clark and Brennan, 1991) models a successful communication as a process during which "common ground" (i.e., mutual knowledge, beliefs etc.) is jointly built among interlocutors. The *interactive alignment model* (IAM) (Pickering and Garrod, 2004) proposes that the ultimate goal of dialogue is the alignment of interlocutors' situational model, which is helped by alignment at all other lower representation levels (e.g., lexical, syntactic etc.), driven by the psychologically well-documented priming effects.

Recently, empirical studies have verified the explanatory powers of the above-mentioned theories, especially the IAM, utilizing dialogues recorded and transcribed from various collaborative tasks conducted in laboratory settings (Reitter and Moore, 2007; Reitter and Moore, 2014; Fusaroli et al., 2012; Fusaroli and Tylén, 2016). In those studies, the quality of communication is directly reflected in the collaborative performance of interlocutors, i.e., how successful they are in accomplishing the task. Although they do not come to fully agree on which theoretical accounts of dialogue (e.g., interactive alignment vs. interpersonal synergy) provides better explanations (see Section 2.1 for details), the majority of these studies have confirmed that the alignment of certain linguistic markers, lexical items, or syntactic rules between interlocutors correlates with task success.

What is missing from the picture, however, is the computational understanding of how strategies of interaction and the mix of information contributions to the conversation facilitate successful communication. This is understandable because those higher level concepts do not directly map onto the atomic linguistic elements and thus are much more difficult to define and operationalize. In the present study, we intend to explore this missing part of work by characterizing how the interaction between interlocutors in terms of their information contributions affects the quality of communication.

1.1 An information-based approach

Recent work has already used information theory to study the dynamics of dialogue. Xu and Reitter (2016b) observed that the amount of lexical information (measured by entropy) from interlocutors of different roles, *converges* within the span of topic episodes in natural spoken dialogue. Anon (2017) interpret this converging pattern as a re-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 623–633
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1058

flection of the dynamic process in which the information contributed by two interlocutors fluctuates in a complementary way at the early stage, and gradually reaches an equilibrium status. Xu and Reitter (2016b) also correlated this entropy converging pattern with the *topic shift* phenomenon that frequently occurs in natural conversation (Ng and Bradac, 1993), and proposed that it reflects the process of interlocutors building the *common ground* that is necessary for the ongoing topics of conversation.

Based on Xu and Reitter's (2016) finding that entropy converging pattern repeatedly occurs within dialogue (though not necessarily at strictly regular intervals), it is reasonable to expect that after applying some spectral analysis techniques (time space to frequency space conversion) to the entropy series of dialogue, the frequency space representations should demonstrate some patterns that are distinct from white noise, because the periodicity properties in time space are captured.

Furthermore, we expect that how the frequency representations of two interlocutors correlate provides some information about the higher level properties of dialogue, e.g., the task performance etc. The thought is intuitive: If we imagine the entropy series from two interlocutors as two ideal sinusoidal signals s_1 and s_2 (supposedly of different frequencies, f_1 and f_2) (Figure 1), then the observed converging pattern can be thought of as a segment from the full spans of the signals. Then the frequency space properties, such as how close f_1 and f_2 are, and the phase difference ϕ between them, will definitely affect the shape of the converging pattern (solid lines in Figure 1). As Xu and Reitter (2016b) argues that the converging segment reflects the *grounding* process between interlocutors, it is reasonable to expect that the shape and length of this segment are reflective of how well interlocutors understand each other, and the overall collaborative performance as well.

Based on the above considerations, the goal of the present study is to explore how the frequency space representations of the entropy series of dialogue are correlated with the collaborative performance of task. We first demonstrate that entropy series satisfy the prerequisites of spectral analysis techniques in Section 4. Then we use two frequency space statistics, *power spectrum overlap* (PSO) and *relative phase* (RP), to predict task success. The reasons of using these two specific indices are discussed in Section 2.3, and their definitions are given in Section 3.3. The results are shown in Sections 5 to 7, and the implications are discussed.

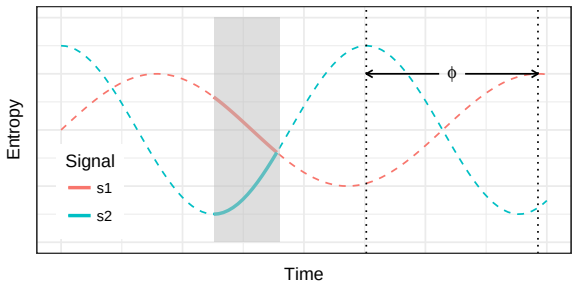


Figure 1: Analogizing the entropy converging patterns reported by Xu and Reitter (2016b) to a segment from two periodic signals. The shadowed area and the solid lines indicate the observed entropy convergence between interlocutors. The dashed lines are the imaginary parts of the ideal signals.

2 Related Work

2.1 The success of dialogue

The interactive-alignment model (IAM) (Pickering and Garrod, 2004) stipulates that communication is successful to the extent that communicators "understand relevant aspects of the world in the same way as each other" (Garrod and Pickering, 2009). Qualitative and quantitative studies (Garrod and A. Anderson, 1987; Pickering and Garrod, 2006; Reitter and Moore, 2014) have revealed that the alignment of linguistic elements at different representation levels between interlocutors facilitates the success of task-oriented dialogues.

More recently, different theoretical accounts other than IAM, such as *interpersonal synergy* (Fusaroli et al., 2014) and *complexity matching* (Abney et al., 2014) have been proposed to explain the mechanism of successful dialogue from the perspective of dynamic systems. Fusaroli and Tylén (2016) compare the approaches of interactive alignment and interpersonal synergy in terms of how well they predict the collective performance in a joint task. They find that the synergy approach is a better predictor than the alignment approach. Abney et al. (2014) differentiate the concepts of *behavior matching* and *complexity matching* in dyadic interaction. They demonstrate the acoustic onset events in speech signals exhibit power law clustering across timescales, and the

complexity matching in these power law functions is reflective of whether the conversation is affiliative or argumentative.

The perspective taken by the present study has some common places with Fusaroli and Tylén (2016) and Abney et al.'s (2014) work: we view dialogue as an interaction of two dynamic systems. The joint decision-making task used by Fusaroli and Tylén (2016) resulted in a small corpus of dialogue in Danish, which we will use for the present study.

2.2 Information density in natural language

Information Theory (Shannon, 1948) predicts that the optimal way to communicate is to send information at a constant rate, a.k.a. the principle of entropy rate constancy (ERC). The way humans use natural language to communicate also follows this principle: by computing the local per-word entropy of the sentence (which, under the prediction of ERC, will increase with sentence position), ERC is confirmed in both written text (Genzel and Charniak, 2002; Genzel and Charniak, 2003; Keller, 2004; Qian and Jaeger, 2011) and spoken dialogue (Xu and Reitter, 2016b; Xu and Reitter, 2016a). The theory of uniform information density (UID) extends ERC to syntactic representations (Jaeger, 2010) and beyond.

The information density in language, i.e., the distribution of entropy (predictability), reveal the discourse structure to some extent. For example, entropy drops at the boundaries between topics (Genzel and Charniak, 2003; Qian and Jaeger, 2011), and increases within a topic episode in dialogue (Xu and Reitter, 2016b) (see Section 1.1). The entropy of microblog text reflects changes in contextual information (e.g., an unexpected event in a sports game) (Doyle and Frank, 2015).

In sum, per-word entropy quantifies the amount of lexical information in natural language, and therefore fulfills the needs of modeling the information contribution from interlocutors.

2.3 Spectral analysis methodology

Spectral analysis, also referred to as frequency domain analysis, is a pervasively used technique in physics, engineering, economics and social sciences. The key idea of it is to decompose a complex signal in time space into simpler components in frequency space, using mathematical operations such as Fourier transform (Bracewell, 1986).

The application of spectral analysis in human language technology mainly focuses on processing the acoustic signals of human voice, and capturing the para-linguistics features relevant to certain tasks (Schuller et al., 2013). For example, Bitouk et al. (2010) find that utterance-level spectral features are useful for emotion recognition. Gregory Jr and Gallagher (2002) demonstrate that spectral information beneath 0.5 kHz can predict US president election outcomes. However, we are not aware of the usage of spectral analysis in studying linguistic phenomena at higher representation levels than the acoustic level.

For our study, we are looking for some techniques that can capture the coupling between two signals at frequency space. The nature of the signal (whether it is language-related or not) should not be the first concern from the perspective of methodology. Therefore, studies outside the field of speech communication and linguistics could also be enlightening to our work.

After searching the literature, we find that the spectral analysis techniques that Oullier et al. (2002) and Oullier et al. (2008) use to study the physical and social functions of human body movement are useful to our research goal. In Oullier et al.'s (2002) work, subjects stood in a moving room and were to track a target attached to the wall. A frequency space statistics, *power spectrum overlap* (PSO), was used to demonstrate the coupling between motion of the room and motion of the subject's head. Stronger coupling effect (higher PSO) was found in the tracking task than a no-tracking baseline. PSO in nature quantifies how much the frequency space representations of two signals (power spectrum density) overlap. It allows us to explore the frequency space coupling of two interlocutors' entropy series in dialogue.

Similarly, Oullier et al. (2008) used the metrics of *peak-to-peak relative phase* (RP) and PSO to study the spontaneous synchrony in behavior that emerges between interactants as a result of information exchange. The signals to be analyzed were the flexion-extension movement of index fingers of two subjects sitting in front of each other. Both metrics showed different patterns when the participants see each other or not. RP, in their work, measures the magnitude of delay between two signals, and it corresponds to the notion of ϕ in Section 1.1.

3 Methods

3.1 Corpus data

Two corpora are examined in this study: the HCRC Map Task Corpus (A. H. Anderson et al., 1991) and a smaller corpus in Danish from a joint decision-making study (Fusaroli et al., 2012), henceforth *DJD*.

Map Task contains a set of 128 dialogues between two subjects, who accomplished a cooperative task together. They were given two slightly different maps of imaginary landmarks. One of them plays as the instruction *giver*, who has routes marked on her map, and the other plays as the instruction *follower*, who does not have routes. The task for them is to reproduce the giver's route on the follower's map. The participants are free to speak, but they cannot see each other's map. The whole conversations were recorded, transcribed and properly annotated. The collaborative performance in the task is measured by the `PATHDEV` variable, which quantifies the deviation between the paths drawn by interlocutors. Larger values indicate poorer task performance.

DJD contains a set of 16 dialogues from native speakers of Danish (11,100 utterances and 56,600 words). In Fusaroli et al.'s (2012) original study the participants were to accomplish a series of visual perception task trials, by discussing the stimuli they saw and reaching a joint decision for each trial. The collaborative performance is measured by the `CollectivePerformance` variable, which is based on a psychometric function that measures the sensitivity of the dyad's joint decision to the actual contrast difference of the trial (Fusaroli et al., 2012). Higher value of this variable indicates better task performance.

The Switchboard Corpus (Godfrey et al., 1992) is used to train the language model for estimating the sentence entropy in Map Task. The Copenhagen Dependency Treebanks Corpus[1] is used for the same purpose for DJD.

3.2 Estimating information density in dialogue

The information density of language is estimated at the sentence level, by computing the per-word entropy of each sentence using a trigram language model trained from a different corpus. We consider a sentence to be a sequence of words, $S = \{w_1, w_2, \ldots, w_n\}$, and the per-word entropy is estimated by:

$$H(w_1 \ldots w_n) = -\frac{1}{n} \sum_{w_i \in W} \log P(w_i | w_1 \ldots w_{i-1}) \quad (1)$$

where $P(w_i|w_1 \ldots w_{i-1})$ is estimated by a trigram model that is trained from an outside corpus. The `SRILM` software (Stolcke, 2002) is used to train the language model and to compute sentence entropy.

Dialogue is a sequence of utterances contributed by two interlocutors. For the k th dialogue whose total utterance number is N_k, we mark it as $D_k = \{u_i^k \mid i = 1, 2, \ldots, N_k\}$, in which u_i^k is the i th utterance. Map Task contains annotations of sentence structure in utterances, and one utterance could consist of several sentences that are syntactically independent. Thus we further split D_k into a sequence of sentence, $D_k = \{s_i^k \mid i = 1, 2, \ldots, N_k'\}$, in which N_k' is number of sentences in D_k. Since DJD lacks the sentence annotations, we do not further split the utterance sequence, and simply treat an utterance as a complete sentence.

Given a sequence $\{s_i^k\}$ (Map Task), or $\{u_i^k\}$ (DJD), we calculate the per-word entropy for each item in the sequence:

$$H_k = \{H(s_i^k) \text{ or } H(u_i^k) \mid i = 1, 2, \ldots, N_k'(or N_k)\} \quad (2)$$

where $H(s_i^k)$ or $H(u_i^k)$ is computed according to Equation 1.

Then we split the entropy series H_k into two sub-series by the source of utterances (i.e., who speaks them), resulting in H_k^A for interlocutor *A*, and H_k^B for interlocutor *B*. For Map Task, the two interlocutors have distinct roles, instruction *giver* and *follower*. Thus the resulting two entropy series are H_k^g and H_k^f. These per-interlocutor entropy series will be the input of our next-step spectral analysis.

3.3 Computing power spectrum overlap and relative phase

The time intervals between utterances (or sentences) vary, but since we care about the average information contribution within a complete semantic unit, we treat entropy series as regular time series. The time scale is not measured in seconds but in turns (or sentences).

For a given dialogue D_k, we apply the fast Fourier transform (FFT) on its two entropy se-

[1] http://mbkromann.github.io/copenhagen-dependency-treebank/

ries H_k^A and H_k^B, and obtain the *power spectra* (or, power spectral density plots) of them, P_k^A and P_k^B. The power spectra are estimated with the *periodogram* method provided by the open source `R` software. The Y axis of a power spectrum is the squared amplitude of signal (or power), and X axis ranges from 0 to $\pi/2$ (we do not have sampling frequency, thus the X axis is in angular frequency but not in Hz).

The *power spectrum overlap*, PSO_k, is calculated by computing the common area under the curves of P_k^A and P_k^B is calculated, and normalizing by the total area of the two curves (see Figure 2). PSO_k ranges from 0 to 1, and a larger value indicates higher similarity between P_k^A and P_k^B.

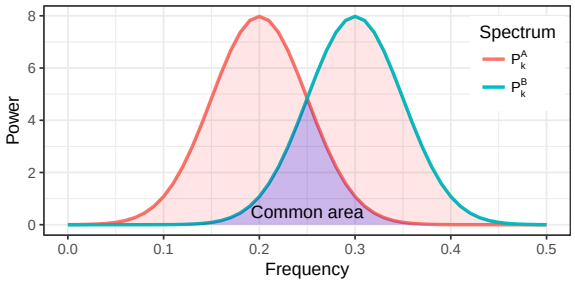


Figure 2: How PSO is computed. The blue shadow is the common area under two spectrums.

The *relative phase* (RP) between H_k^A and H_k^B is directly returned by the `spectrum` function in `R`. It is a vector of real numbers that range from 0 to π, and each element represent the phase difference between two signals at a particular frequency position of the spectrum.

4 Prerequisites of Spectral Analysis

Before proceeding to the actual analysis, we first examine whether the data we use satisfy some of the prerequisites of spectral analysis techniques. One common assumption of Fourier transforms is that the signals (time series) are *stationary* (Dwivedi and Subba Rao, 2011). Stationarity means that the mean, variance and other distributional properties do not change over time (Natrella, 2010). Another presumption we hold is that the entropy series contain some periodic patterns (see Section 1.1), which means their power spectrum should differ from that of white noise.

4.1 Examine stationarity

We use three pervasively used statistical tests to test the stationarity of our entropy series data: the augmented Dickey-Fuller (ADF) test (Dickey and Fuller, 1979), the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test (Kwiatkowski et al., 1992), and the Phillips-Perron (PP) test (Phillips and Perron, 1988). The percentage of entropy series that pass the stationarity tests are shown in Table 1. We can see that the majority of our data satisfy the assumption of stationarity, and thus it is valid to conduct Fourier transform on the entropy series.

Table 1: Percentage stationary data

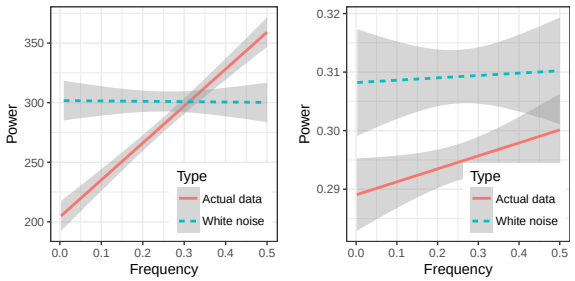
Corpus	ADF	KPSS	PP
Map Task	82.4%	95.5%	100%
DJD	100%	81.3%	100%

The stationarity property seems contradictory to the previous findings about entropy increase in written text and spoken dialogue (Genzel and Charniak, 2002; Genzel and Charniak, 2003; Xu and Reitter, 2016b), because stationarity predicts that the mean entropy stays constant over time. We examine this in our data by fitting a simple linear model with entropy as the dependent, and sentence position as the independent variable, which yields significant (marginal) effects of the latter: For Map Task, $\beta = 2.3 \times 10^{-3}$, $p < .05$, *Adj-*$R^2 = 1.7 \times 10^{-4}$; For DJD, $\beta = 7.2 \times 10^{-5}$, $p = .06$, *Adj-*$R^2 = 2.2 \times 10^{-4}$. It indicates that the stationarity of entropy series does not conflict with the entropy increasing trend predicted by the principle of ERC (Shannon, 1948). We conjecture that stationarity satisfies because the effect size (*Adj-*R^2) of entropy increase is very small.

4.2 Comparison with white noise

Power spectra for all entropy series are obtained with an FFT. We compare them with those of white noise. The white noise data are simulated with *i.i.d.* random data points that are generated from normal distributions (same means and standard deviations as the actual data). Figure 3 shows the smoothed average spectrums of the actual entropy data and the simulated white noise data.

White noise signals should demonstrate a constant power spectral density (Narasimhan and Veena, 2005), and if the entropy series is not completely random, then their average spectrum should be flat. Linear models show that the average spectrums of the entropy data have slopes that are significantly larger than zero (For Map Task,



(a) Map Task (b) DJD

Figure 3: Comparing the average power spectra of the actual entropy data and white noise. There are significant linear correlations between *power* (Y axis) and *frequency* (X axis) for the actual entropy data, which means the data are not completely random. Shadowed areas are 95% C.I.

$\beta = 2.3 \times 10^{-2}$, $SE = 9.4 \times 10^{-3}$, $p < .05$; for DJD, $\beta = 314.1$, $SE = 19.8$, $p < .001$), while the slopes of the white noise data are not significantly different from zero. This confirms our presumption that the entropy series of dialogue contains some periodic patterns that are identifiable in frequency space.

We also conduct Ljung-Box test (Ljung and Box, 1978) to examine how the entropy series is different from white noise. The null hypothesis is that the time series being tested is independent of the lagged sequence of itself. The test on a white noise series will give big p-values, for any lags greater than 0, because of its randomness nature. We try several lags on each entropy series, and pick the smallest p-value. Consequently, we obtain a mean p-value of .23 on MapTask, and a mean p-value of .27 on DJD. Therefore, we cannot reject the null hypothesis for all the entropy series data, but the Type-I error of considering them as different form white noise is pretty low.

5 PSO Predicts Task Success

5.1 Results of linear models

We compute PSO for all the dialogues in Map Task and DJD and fit two linear models using PSO as predictor, with `PATHDEV` and `CollectivePerformance` as dependent variables respectively.

PSO is a reliable predictor in both models ($p < .05$). The coefficients are shown in Table 2. Since `PATHDEV` is a measure of failure, but collaborative task performance is a measure of success, the negative correlation between PSO and collaborative task performance is consistent. Regression lines with residuals are plotted in Figure 4.

Table 2: Coefficients of PSO in predicting `PATHDEV` (Map Task) and `Collective-Performance` (DJD). * indicates $p < .05$.

Dependent	β	SE	F	$Adj\text{-}R^2$
`PATHDEV`	124.8	49.4	6.39*	.045
`Collective-Performance`	-40.9	15.9	6.60*	.271

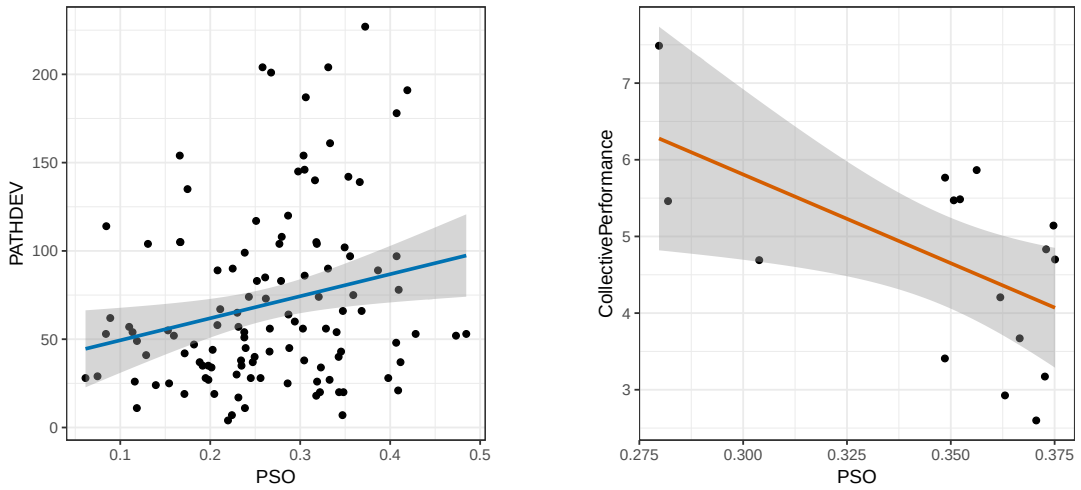
Figure 4 (a) suggests a heteroscedasticity problem, because the right half of data points seem to stretch up along the y axis. This was confirmed by a Breush-Pagan test (Breusch and Pagan, 1979) ($BP = 5.62$, $p < .05$). To rectify this issue, we adopt a Box-Cox transformation (Box and Cox, 1964) on the dependent variable, `PATHDEV`, which is a typical way of handling heteroscedasticity. The new model that uses PSO to predict the Box-Cox transformed `PATHDEV` also yields significant coefficients: $\beta = 3.85$, $SE = 1.67$, $F(1, 113) = 5.32$, $p < .05$. Therefore, the correlation between PSO and `PATHDEV` is reliable.

As for DJD, due to the lack of data (we only have 16 dialogues), we do not run further diagnostics analysis on the regression model.

5.2 Discussion

The coupling of entropy series in frequency space is negatively correlated with task success. In other words, synchrony between interlocutors in terms of their information distribution hinders the success of collaboration. By "synchrony", we mean an overlap in the frequencies at which they choose to inject novel information into the conversation.

This conclusion seems contradictory to the perspective of interactive alignment at the first glance. However, here we are starting with a very high-level model of dialogue that has does not refer to linguistic devices. Instead, we utilize the concept of "information density" and the entropy metric of natural language, to paint the picture of a system in which communicators inject novelty into the dialogue, and that each communicator does so regularly and with a set of overlapping frequencies. We assume that the rapid change of sentence entropy, i.e., the high frequency components in the spectrum, correspond to the moments in conversa-



(a) Map Task

(b) DJD

Figure 4: Regression lines of linear models using PSO to predict `PATHDEV` in Map Task (a) and `CollectivePerformance` in DJD (b). Shadowed areas are 95% C.I.

tion where one interlocutor brings relatively novel content to the table, such as a detailed instruction, a strange question, an unexpected response etc. This assumption is reasonable because previous work has shown that sudden change in entropy predicts topic change in dialogue (Genzel and Charniak, 2003; Qian and Jaeger, 2011; Xu and Reitter, 2016b).

We argue that higher synchrony (larger overlap in frequency space) in terms of how much novelty each interlocutor contributes, does not necessarily leads to better outcomes of communication. Rather, we would expect the correlation to be opposite (and our empirical results confirm this), because dialogue is a joint activity, in which a taking on different roles as interlocutors (e.g., the one who gives orders versus the one who follows) is often required to push the activity along (Clark, 1996). A dialogue with maximal synchrony or frequency overlap would be one where partners take turns at regular intervals. Perhaps because such regularity in turn-taking assigns no special roles to interlocutors, and because they engage in turn-taking with no regard for content, it is not strange that such synchrony is disadvantageous.

Let's look at several scenarios of different synchrony levels between interlocutors: First, high synchrony due to both interlocutors contributing large amount of new information, which means there is more overlap near the high frequency band of spectrums. In this case, they are more likely to have difficulty in comprehending each other due to the potential information overload. Situations such as arguing, or both speakers asking a lot of questions are good examples. Second, high synchrony due to both interlocutors providing ineffective information, which indicates overlap in spectrums near the low frequency band. Obviously this type of ineffective communication is not helpful to the collaborative task. Third, low synchrony due to one interlocutor providing more information and the other one providing less, which means the overlap in spectrums is minimum. An example of this case is that one interlocutor is saying something important, while the other one is producing short utterances such "uh-huh", "yes", or short questions to make sure that they are on the same page, which is known as the back-channel mechanism in conversation (Oreström, 1983). This complementary style of communication allows them to build mutual understand of each other's intention, and thus reaches better collaborative performance.

6 RP Predicts Task Success

6.1 Results of linear models

We obtain the relative phase (RP) vector (absolute values) of all frequency components, and fit linear models using the mean of RP as predictor, and task performance as the dependent variable. We get non-significant coefficients for both models: For Map Task, $F(1, 113) = .004,\ p > .05$; for DJD, $F(1, 14) = .772,\ p > .05$. This suggests that the phase information of all frequency components in spectrum is not very indicative of task success.

The power spectra describe the distribution of energy across the span of frequency components that compose the signal. The frequency components with higher energy (peaks in spectrum) are more dominant than those with lower energy (troughs) in determining the nature of the signal. Therefore it makes sense to only include the peak frequencies into the model, because they are more "representative" of the signal, and so the "noise" from the low energy frequencies are filtered out. Thus we obtain RP from the local peak frequency components, and use the mean, median, and maximum values of them as predictors. It turns out that for Map Task, the maximum of RP is a significant predictor (the mean and median are left out via stepwise analysis). For DJD, the mean of RP is a significant predictor of task success (when median and maximum are included in the model). (see Table 3).

Table 3: Coefficients of the linear models using the mean, median, and maximum values of RP from peak frequency components to predict task performance. $* \, p < .05$, $\dagger \, p < .1$.

Corpus	Predictor	β	SE	t score
Map Task	max	-64.9	30.3	**-2.14***
DJD	mean	15.6	5.7	**2.76***
	median	-7.4	3.6	-2.06†
	max	-11.5	7.2	-1.60

From the significant effect of maximum RP in Map Task and mean RP in DJD, it is safe to state that RP is positively correlated with task performance. However, this relationship is not as straight-forward as PSO, because of the marginal effect at the opposite direction. A more fine-grained analysis is required, but it is outside the scope of this study.

6.2 Discussion

The relative phase in frequency space can be understood as the "lag" between signals in time space. Imagine that we align the two entropy series from one dialogue onto the same time scale (just like Figure 1), the distance between the entropy "peaks" is proportionate to the relative phase in frequency space. Then, the positive correlation between relative phase and task performance suggests that relatively large delays between entropy "surges" seen in each interlocutor are beneficial to collaborative performance.

Table 4: R^2 performance on the HCRC MapTask task success prediction task (percentage of variance explained). 10-fold cross-validated by dialogue; same folds for each model. Reitter and Moore (2007) (R&M) contained length and lexical and syntactic repetition features.

Model	R^2
R&M	.17
R&M LENGTH only	.09
R&M LENGTH only (C=.5)	.1260
R&M (C=.5)	.1771
R&M + PSO + RP	**.2826**
R&M + PSO*RP	.2435
R&M LENGTH only + PSO*RP	.2494

The delay of entropy surges can be understood as a strategy for an interlocutor to distribute information in his or her own utterance accordingly with the information received. For example, after interlocutor *A* contributes a big piece of information, the other one, *B*, does not *rush* to make new substantial contributions, but instead keeps her utterances at low entropy until it is the proper time to take a turn to contribute. This does not have to coincide with dialogic turn-taking.

This delay gives *B* more time to "digest" the information provided by *A*, which could be an instruction that needs to be comprehended, or a question that needs to be thought about and so on. A relatively long delay guarantees enough time for interlocutors to reach mutual understanding. On the contrary, if *B* rushes to speak a lot shortly after the *A*'s input, then it will probably cause information overload and be harmful to communication.

Therefore, we believe that the RP statistic captures the extent to which interlocutors manage the proper "timing" of information contribution to maintain effective communication.

7 Prediction Task

Here we explore whether the frequency domain features, PSO and RP, can help with an existing framework that utilizes alignment features, such as the repetition of lexical and syntactic elements, to predict the success of dialogue in MapTask (Reitter and Moore, 2007).

R&M described an SVM model that takes into the repetition count of lexicons (LEXREP) and syntax structures (SYNREP), and the length of dialogues (LENGTH) as features. The full model achieves an R^2 score of .17, which means that it can account for 17% of the variance of task success.

We add the new PSO and RP (mean, median and maximum RP features per dialogue are included) covariates to the original SVM model. An RBF kernel ($\gamma = 5$) was used. The cost parameter C was (coarsely) tuned on different cross-validation folds to reduce overfitting on this relatively small dataset, and the R&M's original full model was recalculated (shown in Table 4 as R&M). Two models with PSO and RP interactions (once without the alignment/repetition features) are shown for comparison. (See Table 4).

Significant improvement in the model's explanatory power, i.e., R^2, is gained after the PSO and RP features are added. The best model we have is by adding PSO and RP as predictors without the interaction term (bold number in Table 4), which gives about 60% increase of R^2 compared to R&M's full model. Note that even if we exclude the alignment features, and include only (LENGTH) and the frequency features (last row in Table 4), the performance also exceeds R&M's full model.

The results indicate that the frequency domain features (PSO and RP) of the sentence information density can capture some hidden factors of task success that are unexplained by the alignment approach. It is not surprising that how people coordinate their information contribution matters a lot to the success of the collaboration. What we show here is that regular, repeated patterns of information-dense and information-sparse turns seem to make speakers more or less compatible with each other. Whether individuals have typical patterns (frequency distributions) of information density, or whether this is a result of dynamic interaction in each particular dialogue, remains to be seen.

8 Conclusions

The empirical results of the present study suggest that examining how the information contribution from interlocutors co-develops can provide a way to understand dialogue from a higher-level perspective, which has been missing in existing work.

Our work adds a brick to the series of endeavors on studying the linguistic and behavioral factors of successful dialogue, and for the first time (as far as we know) demonstrates quantitatively that the dynamics of not just "what" and "how" we say, but also "how much" we say and the "timing" of distributing what we say in dialogue, are relevant to the quality of communication. Although the way we model information in language is simply the entropy at lexical level, we believe the findings still reveal the nature of information production and processing in dialogue. We hope that by comparing and combining our methodology with other approaches of studying dialogue, we can reach a more comprehensive and holistic understanding of this common yet mysterious human practice.

Acknowledgments

We thank Riccardo Fusaroli for providing the DJD dataset. We have received very helpful input from Gesang Zeren in developing the initial ideas of this project. The work leading to this paper was funded by the National Science Foundation (IIS-1459300 and BCS-1457992).

References

Abney, D. H., Paxton, A., Dale, R., & Kello, C. T. (2014). Complexity matching in dyadic conversation. *Journal of Experimental Psychology: General*, *143*(6), 2304.

Anderson, A. H., Bader, M., Bard, E. G., Boyle, E., Doherty, G., Garrod, S., ... Miller, J. et al. (1991). The HCRC map task corpus. *Language and Speech*, *34*(4), 351–366.

Bitouk, D., Verma, R., & Nenkova, A. (2010). Class-level spectral features for emotion recognition. *Speech Communication*, *52*(7), 613–625.

Box, G. E. & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 211–252.

Bracewell, R. N. (1986). *The Fourier transform and its applications*. New York: McGraw-Hill.

Breusch, T. S. & Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica: Journal of the Econometric Society*, 1287–1294.

Clark, H. H. (1996). *Using language*. Cambridge University Press.

Clark, H. H. & Brennan, S. E. (1991). Grounding in communication. *Perspectives on Socially Shared Cognition*, *13*(1991), 127–149.

Dickey, D. A. & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, *74*(366a), 427–431.

Doyle, G. & Frank, M. C. (2015). Shared common ground influences information density in microblog texts. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (naacl-hlt)*. Denver, DO.

Dwivedi, Y. & Subba Rao, S. (2011). A test for second-order stationarity of a time series based on the discrete Fourier transform. *Journal of Time Series Analysis*, *32*(1), 68–91.

Fusaroli, R., Bahrami, B., Olsen, K., Roepstorff, A., Rees, G., Frith, C., & Tylén, K. (2012). Coming to terms quantifying the benefits of linguistic coordination. *Psychological Science*, *23*(8), 931–939.

Fusaroli, R., Raczaszek-Leonardi, J., & Tylén, K. (2014). Dialog as interpersonal synergy. *New Ideas in Psychology*, *32*, 147–157.

Fusaroli, R. & Tylén, K. (2016). Investigating conversational dynamics: interactive alignment, interpersonal synergy, and collective task performance. *Cognitive Science*, *40*(1), 145–171.

Garrod, S. & Anderson, A. (1987). Saying what you mean in dialogue: a study in conceptual and semantic co-ordination. *Cognition*, *27*(2), 181–218.

Garrod, S. & Pickering, M. J. (2009). Joint action, interactive alignment, and dialog. *Topics in Cognitive Science*, *1*(2), 292–304.

Genzel, D. & Charniak, E. (2002). Entropy rate constancy in text. In *Proc. 40th Annual Meeting on Association for Computational Linguistics* (pp. 199–206). Philadelphia, PA.

Genzel, D. & Charniak, E. (2003). Variation of entropy and parse trees of sentences as a function of the sentence number. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing* (pp. 65–72). Association for Computational Linguistics.

Godfrey, J. J., Holliman, E. C., & McDaniel, J. (1992). Switchboard: telephone speech corpus for research and development. In *International Conference on Acoustics, Speech, and Signal Processing* (Vol. 1, pp. 517–520). IEEE. San Francisco, CA.

Gregory Jr, S. W. & Gallagher, T. J. (2002). Spectral analysis of candidates' nonverbal vocal communication: predicting us presidential election outcomes. *Social Psychology Quarterly*, 298–308.

Jaeger, T. F. (2010). Redundancy and reduction: speakers manage syntactic information density. *Cognitive Psychology*, *61*(1), 23–62.

Keller, F. (2004). The entropy rate principle as a predictor of processing effort: an evaluation against eye-tracking data. In *Proc. conference on Empirical Methods in Natural Language Processing* (pp. 317–324). Barcelona, Spain.

Kwiatkowski, D., Phillips, P. C., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: how sure are we that economic time series have a unit root? *Journal of Econometrics*, *54*(1-3), 159–178.

Ljung, G. M. & Box, G. E. (1978). On a measure of lack of fit in time series models. *Biometrika*, 297–303.

Narasimhan, S. & Veena, S. (2005). *Signal processing: principles and implementation*. Alpha Science Int'l Ltd.

Natrella, M. (2010). *Nist/sematech e-handbook of statistical methods*. NIST/SEMATECH.

Ng, S. H. & Bradac, J. J. (1993). *Power in language: Verbal communication and social influence.* Sage.

Oreström, B. (1983). *Turn-taking in english conversation*. Lund: CWK Gleerup.

Oullier, O., Bardy, B. G., Stoffregen, T. A., & Bootsma, R. J. (2002). Postural coordination in looking and tracking tasks. *Human Movement Science*, *21*(2), 147–167.

Oullier, O., De Guzman, G. C., Jantzen, K. J., Lagarde, J., & Kelso, S. J. (2008). Social coordination dynamics: measuring human bonding. *Social Neuroscience*, *3*(2), 178–192.

Phillips, P. C. & Perron, P. (1988). Testing for a unit root in time series regression. *Biometrika*, 335–346.

Pickering, M. J. & Garrod, S. (2004). Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, *27*(02), 169–190.

Pickering, M. J. & Garrod, S. (2006). Alignment as the basis for successful communication. *Research on Language and Computation*, *4*(2-3), 203–228.

Qian, T. & Jaeger, T. F. (2011). Topic shift in efficient discourse production. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society* (pp. 3313–3318).

Reitter, D. & Moore, J. D. (2007). Predicting success in dialogue. In *Proc. 45th Annual Meeting of the Association of Computational Linguistics* (pp. 808–815). Prague, Czech Republic.

Reitter, D. & Moore, J. D. (2014). Alignment and task success in spoken dialogue. *Journal of Memory and Language*, *76*, 29–46.

Schuller, B., Steidl, S., Batliner, A., Burkhardt, F., Devillers, L., Müller, C., & Narayanan, S. (2013). Paralinguistics in speech and language-state-of-the-art and the challenge. *Computer Speech and Language*, *27*(1), 4–39.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*, 379–423.

Stolcke, A. (2002). SRILM - an extensible language modeling toolkit. In *The 7th International Conference on Spoken Language Processing*. Denver, Colorado.

Xu, Y. & Reitter, D. (2016a). Convergence of syntactic complexity in conversation. In *Proc. 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 443–448). Berlin, Germany.

Xu, Y. & Reitter, D. (2016b, August). Entropy Converges Between Dialogue Participants: Explanations from an Information-Theoretic Perspective. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 537–546). Berlin, Germany: Association for Computational Linguistics.

Affect-LM: A Neural Language Model for Customizable Affective Text Generation

Sayan Ghosh[1], Mathieu Chollet[1], Eugene Laksana[1], Louis-Philippe Morency[2] and Stefan Scherer[1]

[1]Institute for Creative Technologies, University of Southern California, CA, USA
[2]Language Technologies Institute, Carnegie Mellon University, PA, USA
[1]{*sghosh,chollet,elaksana,scherer*}*@ict.usc.edu*
[2]*morency@cs.cmu.edu*

Abstract

Human verbal communication includes affective messages which are conveyed through use of emotionally colored words. There has been a lot of research in this direction but the problem of integrating state-of-the-art neural language models with affective information remains an area ripe for exploration. In this paper, we propose an extension to an LSTM (Long Short-Term Memory) language model for generating conversational text, conditioned on affect categories. Our proposed model, *Affect-LM* enables us to customize the degree of emotional content in generated sentences through an additional design parameter. Perception studies conducted using Amazon Mechanical Turk show that *Affect-LM* generates naturally looking emotional sentences without sacrificing grammatical correctness. *Affect-LM* also learns affect-discriminative word representations, and perplexity experiments show that additional affective information in conversational text can improve language model prediction.

1 Introduction

Affect is a term that subsumes emotion and longer term constructs such as mood and personality and refers to the experience of feeling or emotion (Scherer et al., 2010). Picard (1997) provides a detailed discussion of the importance of affect analysis in human communication and interaction. Within this context the analysis of human affect from text is an important topic in natural language understanding, examples of which include sentiment analysis from Twitter (Nakov et al., 2016), affect analysis from poetry (Kao and Jurafsky, 2012) and studies of correlation between function words and social/psychological processes (Pennebaker, 2011). People exchange verbal messages which not only contain syntactic information, but also information conveying their mental and emotional states. Examples include the use of emotionally colored words (such as *furious* and *joy*) and swear words. The automated processing of affect in human verbal communication is of great importance to understanding spoken language systems, particularly for emerging applications such as dialogue systems and conversational agents.

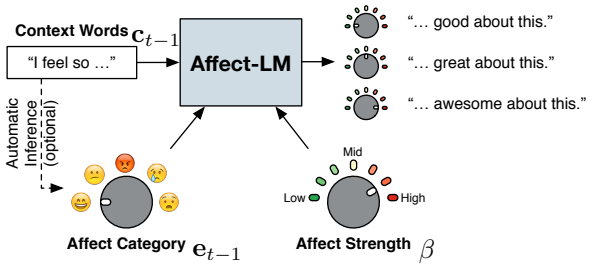


Figure 1: *Affect-LM* is capable of generating emotionally colored conversational text in five specific affect categories ($\mathbf{e}_{t-1}$) with varying affect strengths (β). Three generated example sentences for *happy* affect category are shown in three distinct affect strengths.

Statistical language modeling is an integral component of speech recognition systems, with other applications such as machine translation and information retrieval. There has been a resurgence of research effort in recurrent neural networks for language modeling (Mikolov et al., 2010), which have yielded performances far superior to baseline language models based on n-gram approaches. However, there has not been much effort in building neural language models of text that leverage affective information. Current literature on deep learning for language understanding focuses mainly on representations based on

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 634–642
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1059

word semantics (Mikolov et al., 2013), encoder-decoder models for sentence representations (Cho et al., 2015), language modeling integrated with symbolic knowledge (Ahn et al., 2016) and neural caption generation (Vinyals et al., 2015), but to the best of our knowledge there has been no work on augmenting neural language modeling with affective information, or on data-driven approaches to generate emotional text.

Motivated by these advances in neural language modeling and affective analysis of text, in this paper we propose a model for representation and generation of emotional text, which we call the *Affect-LM*. Our model is trained on conversational speech corpora, common in language modeling for speech recognition applications (Bulyko et al., 2007). Figure 1 provides an overview of our *Affect-LM* and its ability to generate emotionally colored conversational text in a number of affect categories with varying affect strengths. While these parameters can be manually tuned to generate conversational text, the affect category can also be automatically inferred from preceding context words. Specifically for model training, the affect category is derived from features generated using keyword spotting from a dictionary of emotional words, such as the LIWC (Linguistic Inquiry and Word Count) tool (Pennebaker et al., 2001). Our primary research questions in this paper are:
Q1: Can *Affect-LM* be used to generate affective sentences for a target emotion with varying degrees of affect strength through a customizable model parameter?
Q2: Are these generated sentences rated as emotionally expressive as well as grammatically correct in an extensive crowd-sourced perception experiment?
Q3: Does the automatic inference of affect category from the context words improve language modeling performance of the proposed *Affect-LM* over the baseline as measured by perplexity?

The remainder of this paper is organized as follows. In Section 2, we discuss prior work in the fields of neural language modeling, and generation of affective conversational text. In Section 3 we describe the baseline LSTM model and our proposed *Affect-LM* model. Section 4 details the experimental setup, and in Section 5, we discuss results for customizable emotional text generation, perception studies for each affect category, and perplexity improvements over the baseline model before concluding the paper in Section 6.

2 Related Work

Language modeling is an integral component of spoken language systems, and traditionally n-gram approaches have been used (Stolcke et al., 2002) with the shortcoming that they are unable to generalize to word sequences which are not in the training set, but are encountered in unseen data. Bengio et al. (2003) proposed neural language models, which address this shortcoming by generalizing through word representations. Mikolov et al. (2010) and Sundermeyer et al. (2012) extend neural language models to a recurrent architecture, where a target word w_t is predicted from a context of all preceding words $w_1, w_2, ..., w_{t-1}$ with an LSTM (Long Short-Term Memory) neural network. There also has been recent effort on building language models conditioned on other modalities or attributes of the data. For example, Vinyals et al. (2015) introduced the neural image caption generator, where representations learnt from an input image by a CNN (Convolutional Neural Network) are fed to an LSTM language model to generate image captions. Kiros et al. (2014) used an LBL model (Log-Bilinear language model) for two applications - image retrieval given sentence queries, and image captioning. Lower perplexity was achieved on text conditioned on images rather than language models trained only on text.

In contrast, previous literature on affective language generation has not focused sufficiently on customizable state-of-the-art neural network techniques to generate emotional text, nor have they quantitatively evaluated their models on multiple emotionally colored corpora. Mahamood and Reiter (2011) use several NLG (natural language generation) strategies for producing affective medical reports for parents of neonatal infants undergoing healthcare. While they study the difference between affective and non-affective reports, their work is limited only to heuristic based systems and do not include conversational text. Mairesse and Walker (2007) developed PERSONAGE, a system for dialogue generation conditioned on extraversion dimensions. They trained regression models on ground truth judge's selections to automatically determine which of the sentences selected by their model exhibit appropriate extroversion attributes. In Keshtkar and Inkpen (2011), the authors use heuristics and rule-based approaches

for emotional sentence generation. Their generation system is not training on large corpora and they use additional syntactic knowledge of parts of speech to create simple affective sentences. In contrast, our proposed approach builds on state-of-the-art approaches for neural language modeling, utilizes no syntactic prior knowledge, and generates expressive emotional text.

3 Model

3.1 LSTM Language Model

Prior to providing a formulation for our proposed model, we briefly describe a LSTM language model. We have chosen this model as a baseline since it has been reported to achieve state-of-the-art perplexities compared to other approaches, such as n-gram models with Kneser-Ney smoothing (Jozefowicz et al., 2016). Unlike an ordinary recurrent neural network, an LSTM network does not suffer from the vanishing gradient problem which is more pronounced for very long sequences (Hochreiter and Schmidhuber, 1997). Formally, by the chain rule of probability, for a sequence of M words $w_1, w_2, ..., w_M$, the joint probability of all words is given by:

$$P(w_1, w_2, ..., w_M) = \prod_{t=1}^{t=M} P(w_t | w_1, w_2, ..., w_{t-1}) \tag{1}$$

If the vocabulary consists of V words, the conditional probability of word w_t as a function of its context $\mathbf{c_{t-1}} = (w_1, w_2, ..., w_{t-1})$ is given by:

$$P(w_t = i | \mathbf{c_{t-1}}) = \frac{\exp(\mathbf{U_i}^T \mathbf{f}(\mathbf{c_{t-1}}) + b_i)}{\sum_{j=1}^{V} \exp(\mathbf{U_j}^T \mathbf{f}(\mathbf{c_{t-1}}) + b_j)} \tag{2}$$

$\mathbf{f}(.)$ is the output of an LSTM network which takes in the context words $w_1, w_2, ..., w_{t-1}$ as inputs through one-hot representations, $\mathbf{U}$ is a matrix of word representations which on visualization we have found to correspond to POS (Part of Speech) information, while $\mathbf{b_i}$ is a bias term capturing the unigram occurrence of word i. Equation 2 expresses the word w_t as a function of its context for a LSTM language model which does not utilize any additional affective information.

3.2 Proposed Model: *Affect-LM*

The proposed model *Affect-LM* has an additional energy term in the word prediction, and can be described by the following equation:

$$P(w_t = i | \mathbf{c_{t-1}}, \mathbf{e_{t-1}}) = \frac{\exp\left(\mathbf{U_i}^T \mathbf{f}(\mathbf{c_{t-1}}) + \beta \mathbf{V_i}^T \mathbf{g}(\mathbf{e_{t-1}}) + b_i\right)}{\sum_{j=1}^{V} \exp(\mathbf{U_j}^T \mathbf{f}(\mathbf{c_{t-1}}) + \beta \mathbf{V_j}^T \mathbf{g}(\mathbf{e_{t-1}}) + b_j)} \tag{3}$$

$\mathbf{e_{t-1}}$ is an input vector which consists of affect category information obtained from the words in the context during training, and $\mathbf{g}(.)$ is the output of a network operating on $\mathbf{e_{t-1}}$.$\mathbf{V_i}$ is an embedding learnt by the model for the i-th word in the vocabulary and is expected to be discriminative of the affective information conveyed by each word. In Figure 4 we present a visualization of these affective representations.

The parameter β defined in Equation 3, which we call the *affect strength* defines the influence of the affect category information (frequency of emotionally colored words) on the overall prediction of the target word w_t given its context. We can consider the formulation as an energy based model (EBM), where the additional energy term captures the degree of correlation between the predicted word and the affective input (Bengio et al., 2003).

3.3 Descriptors for Affect Category Information

Our proposed model learns a generative model of the next word w_t conditioned not only on the previous words $w_1, w_2, ..., w_{t-1}$ but also on the *affect category* $\mathbf{e_{t-1}}$ which is additional information about emotional content. During model training, the affect category is inferred from the context data itself. Thus we define a suitable feature extractor which can utilize an affective lexicon to *infer* emotion in the context. For our experiments, we have utilized the Linguistic Inquiry and Word Count (LIWC) text analysis program for feature extraction through keyword spotting. Introduced by Pennebaker et al. (2001), LIWC is based on a dictionary, where each word is assigned to a predefined LIWC category. The categories are chosen based on their association with social, affective, and cognitive processes. For example, the dictionary word *worry* is assigned to LIWC category *anxiety*. In our work, we have utilized all word categories of LIWC corresponding to affective processes: *positive emotion, angry, sad, anxious*, and *negative emotion*. Thus the descriptor $\mathbf{e_{t-1}}$ has five features with each feature denoting

Corpus Name	Conversations	Words	% Colored Words	Content
Fisher	11700	21167581	3.79	Conversations
DAIC	688	677389	5.13	Conversations
SEMAINE	959	23706	6.55	Conversations
CMU-MOSI	93	26121	6.54	Monologues

Table 1: Summary of corpora used in this paper. CMU-MOSI and SEMAINE are observed to have higher emotional content than Fisher and DAIC corpora.

presence or absence of a specific emotion, which is obtained by binary thresholding of the features extracted from LIWC. For example, the affective representation of the sentence *i will fight in the war* is $\mathbf{e_{t-1}}$ ={"sad":0, "angry":1, "anxiety":0, "negative emotion":1, "positive emotion":0}.

3.4 *Affect-LM* for Emotional Text Generation

Affect-LM can be used to generate sentences conditioned on the input affect category, the affect strength β, and the context words. For our experiments, we have chosen the following affect categories - *positive emotion, anger, sad, anxiety*, and *negative emotion* (which is a superclass of anger, sad and anxiety). As described in Section 3.2, the affect strength β defines the degree of dominance of the affect-dependent energy term on the word prediction in the language model, consequently after model training we can change β to control the degree of how "emotionally colored" a generated utterance is, varying from $\beta = 0$ (neutral; baseline model) to $\beta = \infty$ (the generated sentences only consist of emotionally colored words, with no grammatical structure).

When *Affect-LM* is used for generation, the affect categories could be either (1) inferred from the context using LIWC (this occurs when we provide sentence beginnings which are emotionally colored themselves), or (2) set to an input emotion descriptor $\mathbf{e}$ (this is obtained by setting $\mathbf{e}$ to a binary vector encoding the desired emotion and works even for neutral sentence beginnings). Given an initial starting set of M words $w_1, w_2, ..., w_M$ to complete, affect strength β, and the number of words N to generate each i-th generated word is obtained by sampling from $P(w_i|w_1, w_2, ..., w_{i-1}, \mathbf{e}; \beta)$ for $i \in \{M+1, M+2, ..., M+N\}$.

4 Experimental Setup

In Section 1, we have introduced three primary research questions related to the ability of the proposed *Affect-LM* model to generate emotionally colored conversational text without sacrificing grammatical correctness, and to obtain lower perplexity than a baseline LSTM language model when evaluated on emotionally colored corpora. In this section, we discuss our experimental setup to address these questions, with a description of *Affect-LM*'s architecture and the corpora used for training and evaluating the language models.

4.1 Speech Corpora

The Fisher English Training Speech Corpus is the main corpus used for training the proposed model, in addition to which we have chosen three emotionally colored conversational corpora. A brief description of each corpus is given below, and in Table 1, we report relevant statistics, such as the total number of words, along with the fraction of emotionally colored words (those belonging to the LIWC affective word categories) in each corpus.

Fisher English Training Speech Parts 1 & 2: The Fisher dataset (Cieri et al., 2004) consists of speech from telephonic conversations of 10 minutes each, along with their associated transcripts. Each conversation is between two strangers who are requested to speak on a randomly selected topic from a set. Examples of conversation topics are *Minimum Wage*, *Time Travel* and *Comedy*.

Distress Assessment Interview Corpus (DAIC): The DAIC corpus introduced by Gratch (2014) consists of 70+ hours of dyadic interviews between a human subject and a virtual human, where the virtual human asks questions designed to diagnose symptoms of psychological distress in the subject such as depression or PTSD (Post Traumatic Stress Disorder).

SEMAINE dataset: SEMAINE (McKeown et al., 2012) is a large audiovisual corpus consisting of interactions between subjects and an operator simulating a SAL (Sensitive Artificial Listener). There are a total of 959 conversations which are approximately 5 minutes each, and are transcribed and annotated with affective dimensions.

Multimodal Opinion-level Sentiment Intensity Dataset (CMU-MOSI): (Zadeh et al., 2016) This is a multimodal annotated corpus of opinion

videos where in each video a speaker expresses his opinion on a commercial product. The corpus consist of speech from 93 videos from 89 distinct speakers (41 male and 48 female speakers). This corpus differs from the others since it contains monologues rather than conversations.

While we find that all corpora contain spoken language, they have the following characteristics different from the Fisher corpus: (1) More emotional content as observed in Table 1, since they have been generated through a human subject's spontaneous replies to questions designed to generate an emotional response, or from conversations on emotion-inducing topics (2) Domain mismatch due to recording environment (for example, the DAIC corpus was created in a mental health setting, while the CMU-MOSI corpus consisted of opinion videos uploaded online). (3) Significantly smaller than the Fisher corpus, which is 25 times the size of the other corpora combined. Thus, we perform training in two separate stages - training of the baseline and *Affect-LM* models on the Fisher corpus, and subsequent adaptation and fine-tuning on each of the emotionally colored corpora.

4.2 *Affect-LM* Neural Architecture

For our experiments, we have implemented a baseline LSTM language model in Tensorflow (Abadi et al., 2016), which follows the non-regularized implementation as described in Zaremba et al. (2014) and to which we have added a separate energy term for the affect category in implementing *Affect-LM*. We have used a vocabulary of 10000 words and an LSTM network with 2 hidden layers and 200 neurons per hidden layer. The network is unrolled for 20 time steps, and the size of each minibatch is 20. The affect category $\mathbf{e_{t-1}}$ is processed by a multi-layer perceptron with a single hidden layer of 100 neurons and sigmoid activation function to yield $\mathbf{g(e_{t-1})}$. We have set the output layer size to 200 for both $\mathbf{f(c_{t-1})}$ and $\mathbf{g(e_{t-1})}$. We have kept the network architecture constant throughout for ease of comparison between the baseline and *Affect-LM*.

4.3 Language Modeling Experiments

Affect-LM can also be used as a language model where the next predicted word is estimated from the words in the context, along with an affect category extracted from the context words themselves (instead of being encoded externally as in generation). To evaluate whether additional emotional information could improve the prediction performance, we train the corpora detailed in Section 4.1 in two stages as described below:

(1) **Training and validation of the language models on Fisher dataset**- The Fisher corpus is split in a 75:15:10 ratio corresponding to the training, validation and evaluation subsets respectively, and following the implementation in Zaremba et al. (2014), we train the language models (both the baseline and *Affect-LM*) on the training split for 13 epochs, with a learning rate of 1.0 for the first four epochs, and the rate decreasing by a factor of 2 after every subsequent epoch. The learning rate and neural architecture are the same for all models. We validate the model over the affect strength $\beta \in [1.0, 1.5, 1.75, 2.0, 2.25, 2.5, 3.0]$. The best performing model on the Fisher validation set is chosen and used as a seed for subsequent adaptation on the emotionally colored corpora.

(2) **Fine-tuning the seed model on other corpora**- Each of the three corpora - CMU-MOSI, DAIC and SEMAINE are split in a 75:15:10 ratio to create individual training, validation and evaluation subsets. For both the baseline and *Affect-LM*, the best performing model from Stage 1 (the seed model) is fine-tuned on each of the training corpora, with a learning rate of 0.25 which is constant throughout, and a validation grid of $\beta \in [1.0, 1.5, 1.75, 2.0]$. For each model adapted on a corpus, we compare the perplexities obtained by *Affect-LM* and the baseline model when evaluated on that corpus.

4.4 Sentence Generation Perception Study

We assess *Affect-LM's* ability to generate emotionally colored text of varying degrees without severely deteriorating grammatical correctness, by conducting an extensive perception study on Amazon's Mechanical Turk (MTurk) platform. The MTurk platform has been successfully used in the past for a wide range of perception experiments and has been shown to be an excellent resource to collect human ratings for large studies (Buhrmester et al., 2011). Specifically, we generated more than 200 sentences for four sentence beginnings (namely the three sentence beginnings listed in Table 2 as well as an end of sentence token indicating that the model should generate a new sentence) in five affect categories *happy(positive emotion), angry, sad, anxiety*, and *negative emotion*. The *Affect-LM* model trained

Beginning	Affect Category	Completed sentence
I feel so	Happy	good because i think that it's important to have a relationship with a friend
	Angry	bad that i hate it and i hate that because they they kill themselves and then they fight
	Sad	sad to miss because i i miss the feelings of family members who i lost feelings with
	Anxious	horrible i mean i think when we're going to you know war and alert alert and we're actually gonna die
	Neutral	bad if i didn't know that the decision was going on
I told him to	Happy	be honest and i said well i hope that i 'm going to be a better person
	Angry	see why he was fighting with my son
	Sad	leave the house because i hurt one and i lost his leg and hurt him
	Anxious	be afraid of him and he he just he just didn't care about the death penalty
	Neutral	do this position i think he is he's got a lot of money he has to pay himself a lot of money
Why did you	Happy	have a best friend
	Angry	say it was only a criminal being killed at a war or something
	Sad	miss your feelings
	Anxious	worry about fear factor
	Neutral	believe in divorce

Table 2: Example sentences generated by the model conditioned on different affect categories

on the Fisher corpus was used for sentence generation. Each sentence was evaluated by two human raters that have a minimum approval rating of 98% and are located in the United States. The human raters were instructed that the sentences should be considered to be taken from a conversational rather than a written context: repetitions and pause fillers (e.g., *um, uh*) are common and no punctuation is provided. The human raters evaluated each sentence on a seven-point Likert scale for the five affect categories, overall *affective valence* as well as the sentence's *grammatical correctness* and were paid 0.05USD per sentence. We measured inter-rater agreement using Krippendorffs α and observed considerable agreement between raters across all categories (e.g., for *valence* $\alpha = 0.510$ and *grammatical correctness* $\alpha = 0.505$).

For each *target emotion* (i.e., intended emotion of generated sentences) we conducted an initial MANOVA, with human ratings of affect categories the DVs (dependent variables) and the affect strength parameter β the IV (independent variable). We then conducted follow-up univariate ANOVAs to identify which DV changes significantly with β. In total we conducted 5 MANOVAs and 30 follow-up ANOVAs, which required us to update the significance level to $p<0.001$ following a Bonferroni correction.

5 Results

5.1 Generation of Emotional Text

In Section 3.4 we have described the process of sampling text from the model conditioned on input affective information (research question **Q1**). Table 2 shows three sentences generated by the model for input sentence beginnings *I feel so ...*, *Why did you ...* and *I told him to ...* for each of five affect categories - *happy(positive emotion), angry, sad anxiety*, and *neutral(no emotion)*. They have been selected from a pool of 20 generated sentences for each category and sentence beginning.

5.2 MTurk Perception Experiments

In the following we address research question **Q2** by reporting the main statistical findings of our MTurk study, which are visualized in Figures 2 and 3.

Positive Emotion Sentences. The multivariate result was significant for *positive emotion* generated sentences (Pillai's Trace=.327, $F(4,437)=6.44$, $p<.0001$). Follow up ANOVAs revealed significant results for all DVs except *angry* with $p<.0001$, indicating that both *affective valence* and *happy* DVs were successfully manipulated with β, as seen in Figure 2(a). *Grammatical correctness* was also significantly influenced by the affect strength parameter β and results show that the correctness deteriorates with increasing β (see Figure 3). However, a post-hoc Tukey test revealed that only the highest β value shows a significant drop in *grammatical correctness* at $p<.05$.

Negative Emotion Sentences. The multivariate result was significant for *negative emotion* generated sentences (Pillai's Trace=.130, $F(4,413)=2.30$, $p<.0005$). Follow up ANOVAs revealed significant results for *affective valence* and *happy* DVs with $p<.0005$, indicating that the *affective valence* DV was successfully manipulated with β, as seen in Figure 2(b). Further, as intended there were no significant differences for DVs *angry, sad* and *anxious*, indicating that the *negative emotion* DV refers to a more general affect related concept rather than a specific negative emotion. This finding is in concordance with the intended LIWC category of *negative affect* that forms a parent category above the more

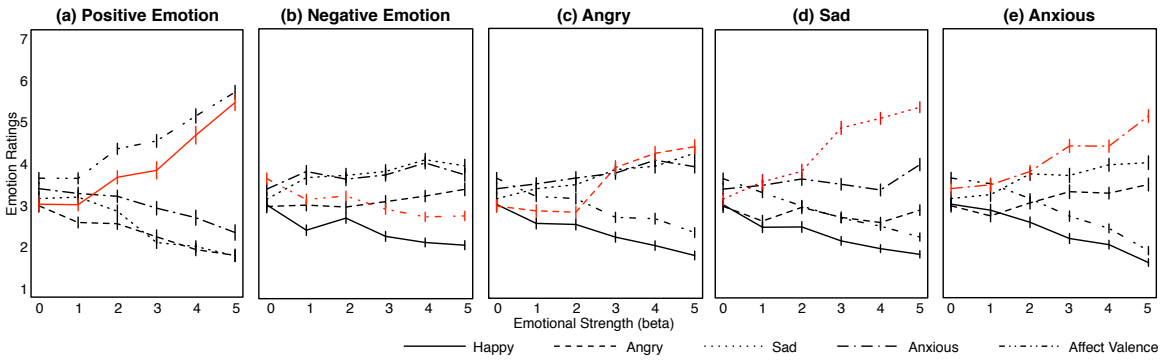


Figure 2: Amazon Mechanical Turk study results for generated sentences in the target affect categories *positive emotion, negative emotion, angry, sad*, and *anxious* (a)-(e). The most relevant human rating curve for each generated emotion is highlighted in red, while less relevant rating curves are visualized in black. Affect categories are coded via different line types and listed in legend below figure.

specific emotions, such as *angry, sad*, and *anxious* (Pennebaker et al., 2001). *Grammatical correctness* was also significantly influenced by the affect strength β and results show that the correctness deteriorates with increasing β (see Figure 3). As for *positive emotion*, a post-hoc Tukey test revealed that only the highest β value shows a significant drop in *grammatical correctness* at p<.05.

Angry Sentences. The multivariate result was significant for *angry* generated sentences (Pillai's Trace=.199, F(4,433)=3.76, p<.0001). Follow up ANOVAs revealed significant results for *affective valence, happy*, and *angry* DVs with p<.0001, indicating that both *affective valence* and *angry* DVs were successfully manipulated with β, as seen in Figure 2(c). *Grammatical correctness* was not significantly influenced by the affect strength parameter β, which indicates that angry sentences are highly stable across a wide range of β (see Figure 3). However, it seems that human raters could not successfully distinguish between *angry, sad*, and *anxious* affect categories, indicating that the generated sentences likely follow a general *negative affect* dimension.

Sad Sentences. The multivariate result was significant for *sad* generated sentences (Pillai's Trace=.377, F(4,425)=7.33, p<.0001). Follow up ANOVAs revealed significant results only for the *sad* DV with p<.0001, indicating that while the *sad* DV can be successfully manipulated with β, as seen in Figure 2(d). The *grammatical correctness* deteriorates significantly with β. Specifically, a post-hoc Tukey test revealed that only the two highest β values show a significant drop in *grammatical correctness* at p<.05 (see Figure 3).

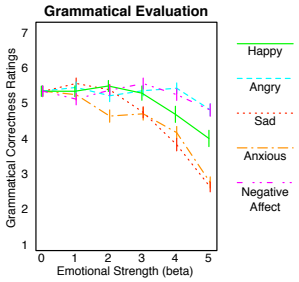


Figure 3: Mechanical Turk study results for *grammatical correctness* for all generated target emotions. Perceived *grammatical correctness* for each affect categories are color-coded.

A post-hoc Tukey test for *sad* reveals that $\beta = 3$ is optimal for this DV, since it leads to a significant jump in the perceived sadness scores at p<.005 for $\beta \in \{0, 1, 2\}$.

Anxious Sentences. The multivariate result was significant for *anxious* generated sentences (Pillai's Trace=.289, F(4,421)=6.44, p<.0001). Follow up ANOVAs revealed significant results for *affective valence, happy* and *anxious* DVs with p<.0001, indicating that both *affective valence* and *anxiety* DVs were successfully manipulated with β, as seen in Figure 2(e). *Grammatical correctness* was also significantly influenced by the affect strength parameter β and results show that the correctness deteriorates with increasing β. Similarly for *sad*, a post-hoc Tukey test revealed that only the two highest β values show a significant drop in *grammatical correctness* at p<.05 (see Figure 3). Again, a post-hoc Tukey test for *anxious* reveals that $\beta = 3$ is optimal for this DV, since it leads to a significant jump in the perceived

	Training (Fisher)		Adaptation	
Perplexity	**Baseline**	**Affect-LM**	**Baseline**	**Affect-LM**
Fisher	37.97	37.89	-	-
DAIC	65.02	64.95	55.86	55.55
SEMAINE	88.18	86.12	57.58	57.26
CMU-MOSI	104.74	101.19	66.72	64.99
Average	73.98	72.54	60.05	59.26

Table 3: Evaluation perplexity scores obtained by the baseline and *Affect-LM* models when trained on Fisher and subsequently adapted on DAIC, SEMAINE and CMU-MOSI corpora

anxiety scores at p<.005 for $\beta \in \{0, 1, 2\}$.

5.3 Language Modeling Results

In Table 3, we address research question **Q3** by presenting the perplexity scores obtained by the baseline model and *Affect-LM*, when trained on the Fisher corpus and subsequently adapted on three emotional corpora (each adapted model is individually trained on CMU-MOSI, DAIC and SEMAINE). The models trained on Fisher are evaluated on all corpora while each adapted model is evaluated only on it's respective corpus. For all corpora, we find that *Affect-LM* achieves lower perplexity on average than the baseline model, implying that affect category information obtained from the context words improves language model prediction. The average perplexity improvement is 1.44 (relative improvement 1.94%) for the model trained on Fisher, while it is 0.79 (1.31%) for the adapted models. We note that larger improvements in perplexity are observed for corpora with higher content of emotional words. This is supported by the results in Table 3, where *Affect-LM* obtains a larger reduction in perplexity for the CMU-MOSI and SEMAINE corpora, which respectively consist of 2.76% and 2.75% more emotional words than the Fisher corpus.

5.4 Word Representations

In Equation 3, *Affect-LM* learns a weight matrix $\mathbf{V}$ which captures the correlation between the predicted word w_t, and the affect category $\mathbf{e_{t-1}}$. Thus, each row of the matrix $\mathbf{V_i}$ is an emotionally meaningful embedding of the i-th word in the vocabulary. In Figure 4, we present a t-SNE visualization of these embeddings, where each data point is a separate word, and words which appear in the LIWC dictionary are colored based on which affect category they belong to (we have labeled only words in categories *positive emotion*, *negative emotion*, *anger*, *sad* and *anxiety* since

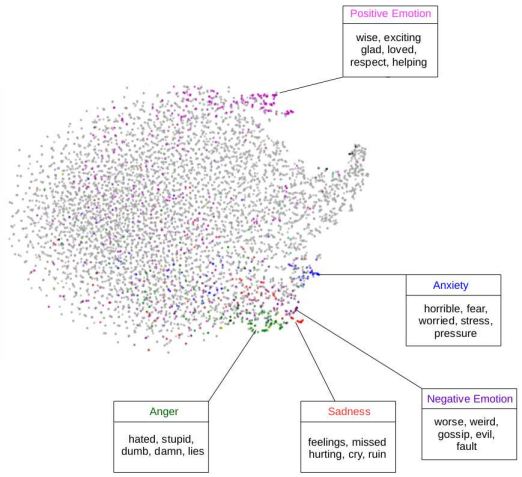


Figure 4: Embeddings learnt by *Affect-LM*

these categories contain the most frequent words). Words colored grey are those not in the LIWC dictionary. In Figure 4, we observe that the embeddings contain affective information, where the positive emotion is highly separated from the negative emotions (*sad*, *angry*, *anxiety*) which are clustered together.

6 Conclusions and Future Work

In this paper, we have introduced a novel language model *Affect-LM* for generating affective conversational text conditioned on context words, an affective category and an affective strength parameter. MTurk perception studies show that the model can generate expressive text at varying degrees of emotional strength without affecting grammatical correctness. We also evaluate *Affect-LM* as a language model and show that it achieves lower perplexity than a baseline LSTM model when the affect category is obtained from the words in the context. For future work, we wish to extend this model by investigating language generation conditioned on other modalities such as facial images and speech, and to applications such as dialogue generation for virtual agents.

Acknowledgments

This material is based upon work supported by the U.S. Army Research Laboratory under contract number W911NF-14-D-0005. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Government, and no official endorsement should be inferred. Sayan Ghosh also acknowledges the Viterbi Graduate School Fellowship for funding his graduate studies.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318* .

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. 2011. Amazon's mechanical turk a new source of inexpensive, yet high-quality, data? *Perspectives on psychological science* 6(1):3–5.

Ivan Bulyko, Mari Ostendorf, Manhung Siu, Tim Ng, Andreas Stolcke, and Özgür Çetin. 2007. Web resources for language modeling in conversational speech recognition. *ACM Transactions on Speech and Language Processing (TSLP)* 5(1):1.

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia* 17(11):1875–1886.

Christopher Cieri, David Miller, and Kevin Walker. 2004. The fisher corpus: a resource for the next generations of speech-to-text. In *LREC*. volume 4, pages 69–71.

Martín Abadi et al. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*.

Jonathan et al. Gratch. 2014. The distress analysis interview corpus of human and computer interviews. In *LREC*. Citeseer, pages 3123–3128.

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

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* .

Justine Kao and Dan Jurafsky. 2012. A computational analysis of style, affect, and imagery in contemporary poetry.

Fazel Keshtkar and Diana Inkpen. 2011. A pattern-based model for generating text to express emotion. In *Affective Computing and Intelligent Interaction*, Springer, pages 11–21.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Multimodal neural language models.

Saad Mahamood and Ehud Reiter. 2011. Generating affective natural language for parents of neonatal infants. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, pages 12–21.

François Mairesse and Marilyn Walker. 2007. Personage: Personality generation for dialogue.

Gary McKeown, Michel Valstar, Roddy Cowie, Maja Pantic, and Marc Schroder. 2012. The semaine database: Annotated multimodal records of emotionally colored conversations between a person and a limited agent. *IEEE Transactions on Affective Computing* 3(1):5–17.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. *Proceedings of SemEval* pages 1–18.

James W Pennebaker. 2011. The secret life of pronouns. *New Scientist* 211(2828):42–45.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates* 71(2001):2001.

Rosalind Picard. 1997. *Affective computing*, volume 252. MIT press Cambridge.

Klaus R Scherer, Tanja Bänziger, and Etienne Roesch. 2010. *A Blueprint for Affective Computing: A sourcebook and manual*. Oxford University Press.

Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *Interspeech*. volume 2002, page 2002.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Interspeech*. pages 194–197.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems* 31(6):82–88.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Domain Attention with an Ensemble of Experts

Young-Bum Kim[†] **Karl Stratos**[‡] **Dongchan Kim**[†]

[†]Microsoft AI and Research
[‡]Bloomberg L. P.
{ybkim, dongchan.kim}@microsoft.com
me@karlstratos.com

Abstract

An important problem in domain adaptation is to quickly generalize to a new domain with limited supervision given K existing domains. One approach is to retrain a global model across all $K + 1$ domains using standard techniques, for instance Daumé III (2009). However, it is desirable to adapt without having to reestimate a global model from scratch each time a new domain with potentially new intents and slots is added. We describe a solution based on attending an ensemble of domain experts. We assume K domain-specific intent and slot models trained on respective domains. When given domain $K + 1$, our model uses a weighted combination of the K domain experts' feedback along with its own opinion to make predictions on the new domain. In experiments, the model significantly outperforms baselines that do not use domain adaptation and also performs better than the full retraining approach.

1 Introduction

An important problem in domain adaptation is to quickly generalize to a new domain with limited supervision given K existing domains. In spoken language understanding, new domains of interest for categorizing user utterances are added on a regular basis[1]. For instance, we may add `ORDERPIZZA` domain and desire a domain-specific intent and semantic slot tagger with a limited amount of training data. Training only on the target domain fails to utilize the existing resources in other domains that are relevant (e.g., labeled data for `PLACES` domain with `place_name`, `location` as the slot types), but naively training on the union of all domains does not work well since different domains can have widely varying distributions.

Domain adaptation offers a balance between these extremes by using all data but simultaneously distinguishing domain types. A common approach for adapting to a new domain is to retrain a global model across all $K + 1$ domains using well-known techniques, for example the feature augmentation method of Daumé III (2009) which trains a single model that has one domain-invariant component along with $K + 1$ domain-specific components each of which is specialized in a particular domain. While such a global model is effective, it requires re-estimating a model from scratch on all $K + 1$ domains each time a new domain is added. This is burdensome particularly in our scenario in which new domains can arise frequently.

In this paper, we present an alternative solution based on attending an ensemble of domain experts. We assume that we have already trained K domain-specific models on respective domains. Given a new domain $K+1$ with a small amount of training data, we train a model on that data alone but queries the K experts as part of the training procedure. We compute an attention weight for each of these experts and use their combined feedback along with the model's own opinion to make predictions. This way, the model is able to selectively capitalize on relevant domains much like in

[1]A scenario frequently arising in practice is having a request for creating a new virtual domain targeting a specific application. One typical use case is that of building natural language capability through intent and slot modeling (without actually building a domain classifier) targeting a specific application.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 643–653
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1060

standard domain adaptation but without explicitly re-training on all domains together.

In experiments, we show clear gains in a domain adaptation scenario across 7 test domains, yielding average error reductions of 44.97% for intent classification and 32.30% for slot tagging compared to baselines that do not use domain adaptation. Moreover we have higher accuracy than the full re-training approach of Kim et al. (2016c), a neural analog of Daumé III (2009).

2 Related Work

2.1 Domain Adaptation

There is a venerable history of research on domain adaptation (Daume III and Marcu, 2006; Daumé III, 2009; Blitzer et al., 2006, 2007; Pan et al., 2011) which is concerned with the shift in data distribution from one domain to another. In the context of NLP, a particularly successful approach is the feature augmentation method of Daumé III (2009) whose key insight is that if we partition the model parameters to those that handle *common patterns* and those that handle *domain-specific patterns*, the model is forced to learn from all domains yet preserve domain-specific knowledge. The method is generalized to the neural paradigm by Kim et al. (2016c) who jointly use a domain-specific LSTM and also a global LSTM shared across all domains. In the context of SLU, Jaech et al. (2016) proposed K domain-specific feedforward layers with a shared word-level LSTM layer across domains; Kim et al. (2016c) instead employed $K + 1$ LSTMs. Hakkani-Tür et al. (2016) proposed to employ a sequence-to-sequence model by introducing a fictitious symbol at the end of an utterance of which tag represents the corresponding domain and intent.

All these methods require one to re-train a model from scratch to make it learn the correlation and invariance between domains. This becomes difficult to scale when there is a new domain coming in at high frequency. We address this problem by proposing a method that only calls K trained domain experts; we do not have to re-train these domain experts. This gives a clear computational advantage over the feature augmentation method.

2.2 Spoken Language Understanding

Recently, there has been much investment on the personal digital assistant (PDA) technology in industry (Sarikaya, 2015; Sarikaya et al., 2016). Apples Siri, Google Now, Microsofts Cortana, and Amazons Alexa are some examples of personal digital assistants. Spoken language understanding (SLU) is an important component of these examples that allows natural communication between the user and the agent (Tur, 2006; El-Kahky et al., 2014). PDAs support a number of scenarios including creating reminders, setting up alarms, note taking, scheduling meetings, finding and consuming entertainment (i.e. movie, music, games), finding places of interest and getting driving directions to them (Kim et al., 2016a).

Naturally, there has been an extensive line of prior studies for domain scaling problems to easily scale to a larger number of domains: pre-training (Kim et al., 2015c), transfer learning (Kim et al., 2015d), constrained decoding with a single model (Kim et al., 2016a), multi-task learning (Jaech et al., 2016), neural domain adaptation (Kim et al., 2016c), domainless adaptation (Kim et al., 2016b), a sequence-to-sequence model (Hakkani-Tür et al., 2016), adversary domain training (Kim et al., 2017) and zero-shot learning(Chen et al., 2016; Ferreira et al., 2015).

There are also a line of prior works on enhancing model capability and features: jointly modeling intent and slot predictions (Jeong and Lee, 2008; Xu and Sarikaya, 2013; Guo et al., 2014; Zhang and Wang, 2016; Liu and Lane, 2016a,b), modeling SLU models with web search click logs (Li et al., 2009; Kim et al., 2015a) and enhancing features, including representations (Anastasakos et al., 2014; Sarikaya et al., 2014; Celikyilmaz et al., 2016, 2010; Kim et al., 2016d) and lexicon (Liu and Sarikaya, 2014; Kim et al., 2015b).

3 Method

We use an LSTM simply as a mapping $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ that takes an input vector x and a state vector h to output a new state vector $h' = \phi(x, h)$. See Hochreiter and Schmidhuber (1997) for a detailed description. At a high level, the individual model consists of builds on several ingredients shown in Figure 1: character and word embedding, a bidirectional LSTM (BiLSTM) at a character layer, a BiLSTM at word level, and feedfoward network at the output.

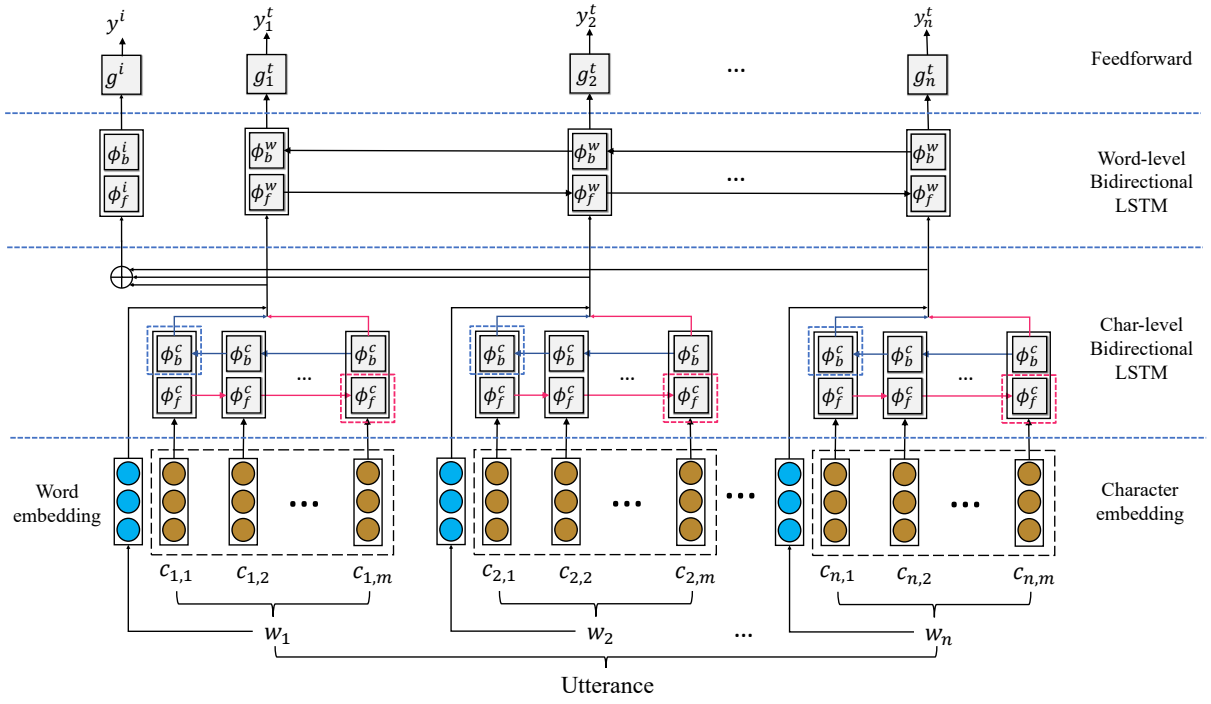


Figure 1: The overall network architecture of the individual model.

3.1 Individual Model Architecture

Let $\mathcal{C}$ denote the set of character types and $\mathcal{W}$ the set of word types. Let $\oplus$ denote the vector concatenation operation. A wildly successful architecture for encoding a sentence $(w_1 \dots w_n) \in \mathcal{W}^n$ is given by bidirectional LSTMs (BiLSTMs) (Schuster and Paliwal, 1997; Graves, 2012). Our model first constructs a network over an utterance closely following Lample et al. (2016). The model parameters Θ associated with this BiLSTM layer are

- Character embedding $e_c \in \mathbb{R}^{25}$ for each $c \in \mathcal{C}$
- Character LSTMs $\phi^{\mathcal{C}}_f, \phi^{\mathcal{C}}_b : \mathbb{R}^{25} \times \mathbb{R}^{25} \to \mathbb{R}^{25}$
- Word embedding $e_w \in \mathbb{R}^{100}$ for each $w \in \mathcal{W}$
- Word LSTMs $\phi^{\mathcal{W}}_f, \phi^{\mathcal{W}}_b : \mathbb{R}^{150} \times \mathbb{R}^{100} \to \mathbb{R}^{100}$

Let $w_1 \dots w_n \in \mathcal{W}$ denote a word sequence where word w_i has character $w_i(j) \in \mathcal{C}$ at position j. First, the model computes a character-sensitive word representation $v_i \in \mathbb{R}^{150}$ as

$$
\begin{aligned}
f^{\mathcal{C}}_j &= \phi^{\mathcal{C}}_f\left(e_{w_i(j)}, f^{\mathcal{C}}_{j-1}\right) && \forall j = 1 \dots |w_i| \\
b^{\mathcal{C}}_j &= \phi^{\mathcal{C}}_b\left(e_{w_i(j)}, b^{\mathcal{C}}_{j+1}\right) && \forall j = |w_i| \dots 1 \\
v_i &= f^{\mathcal{C}}_{|w_i|} \oplus b^{\mathcal{C}}_1 \oplus e_{w_i}
\end{aligned}
$$

for each $i = 1 \dots n$.[2] Next, the model computes

$$
\begin{aligned}
f^{\mathcal{W}}_i &= \phi^{\mathcal{W}}_f\left(v_i, f^{\mathcal{W}}_{i-1}\right) && \forall i = 1 \dots n \\
b^{\mathcal{W}}_i &= \phi^{\mathcal{W}}_b\left(v_i, b^{\mathcal{W}}_{i+1}\right) && \forall i = n \dots 1
\end{aligned}
$$

and induces a character- and context-sensitive word representation $h_i \in \mathbb{R}^{200}$ as

$$h_i = f^{\mathcal{W}}_i \oplus b^{\mathcal{W}}_i \tag{1}$$

for each $i = 1 \dots n$. These vectors can be used to perform intent classification or slot tagging on the utterance.

Intent Classification We can predict the intent of the utterance using $(h_1 \dots h_n) \in \mathbb{R}^{200}$ in (1) as follows. Let $\mathcal{I}$ denote the set of intent types. We introduce a single-layer feedforward network $g^i : \mathbb{R}^{200} \to \mathbb{R}^{|\mathcal{I}|}$ whose parameters are denoted by Θ^i. We compute a $|\mathcal{I}|$-dimensional vector

$$\mu^i = g^i\left(\sum_{i=1}^{n} h_i\right)$$

and define the conditional probability of the correct intent τ as

$$p(\tau | h_1 \dots h_n) \propto \exp\left(\mu^i_\tau\right) \tag{2}$$

[2] For simplicity, we assume some random initial state vectors such as $f^{\mathcal{C}}_0$ and $b^{\mathcal{C}}_{|w_i|+1}$ when we describe LSTMs.

The intent classification loss is given by the negative log likelihood:

$$L^i\left(\Theta, \Theta^i\right) = -\sum_l \log p\left(\tau^{(l)}|h^{(l)}\right) \quad (3)$$

where l iterates over intent-annotated utterances.

Slot Tagging We predict the semantic slots of the utterance using $(h_1 \ldots h_n) \in \mathbb{R}^{200}$ in (1) as follows. Let $\mathcal{S}$ denote the set of semantic types and $\mathcal{L}$ the set of corresponding BIO label types [3] that is, $\mathcal{L} = \{\texttt{B}\text{-}e : e \in \mathcal{E}\} \cup \{\texttt{I}\text{-}e : e \in \mathcal{E}\} \cup \{\texttt{O}\}$. We add a transition matrix $T \in \mathbb{R}^{|\mathcal{L}|\times|\mathcal{L}|}$ and a single-layer feedforward network $g^t : \mathbb{R}^{200} \rightarrow \mathbb{R}^{|\mathcal{L}|}$ to the network; denote these additional parameters by Θ^t. The conditional random field (CRF) tagging layer defines a joint distribution over label sequences of $y_1 \ldots y_n \in \mathcal{L}$ of $w_1 \ldots w_n$ as

$$p(y_1 \ldots y_n|h_1 \ldots h_n) \propto \exp\left(\sum_{i=1}^{n} T_{y_{i-1},y_i} \times g^t_{y_i}(h_i)\right) \quad (4)$$

The tagging loss is given by the negative log likelihood:

$$L^t\left(\Theta, \Theta^t\right) = -\sum_l \log p\left(y^{(l)}|h^{(l)}\right) \quad (5)$$

where l iterates over tagged sentences in the data. Alternatively, we can optimize the local loss:

$$L^{t-loc}\left(\Theta, \Theta^t\right) = -\sum_l \sum_i \log p\left(y_i^{(l)}|h_i^{(l)}\right) \quad (6)$$

where $p(y_i|h_i) \propto \exp\left(g^t_{y_i}(h_i)\right)$.

4 Method

4.1 Domain Attention Architecture

Now we assume that for each of the K domains we have an individual model described in Section 3.1. Denote these domain experts by $\Theta^{(1)} \ldots \Theta^{(K)}$. We now describe our model for a new domain $K+1$. Given an utterance $w_1 \ldots w_n$, it uses a BiLSTM layer to induce a feature representation $h_1 \ldots h_n$ as specified in (1). It further invokes K domain experts $\Theta^{(1)} \ldots \Theta^{(K)}$ on this utterance to obtain the feature representations $h_1^{(k)} \ldots h_n^{(k)}$ for

[3] For example, to/O San/B-Source Francisco/I-Source airport/O.

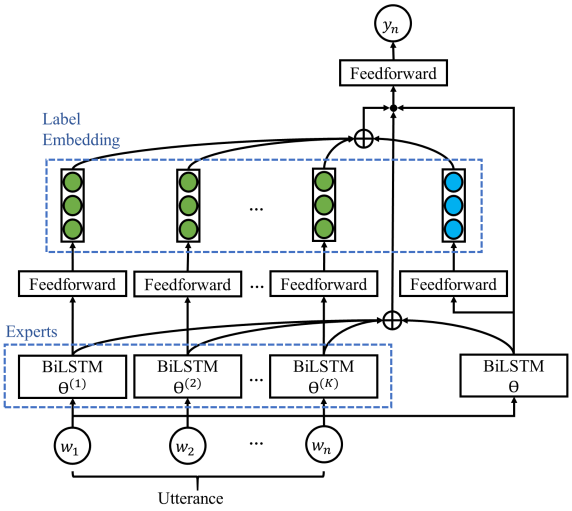


Figure 2: The overall network architecture of the domain attention, which consists of three components: (1) K domain experts + 1 target BiLSTM layer to induce a feature representation, (2) K domain experts + 1 target feedfoward layer to output pre-trained label embedding (3) a final feedforward layer to output an intent or slot. We have two separate attention mechanisms to combine feedback from domain experts.

$k = 1 \ldots K$. For each word w_i, the model computes an attention weight for each domain $k = 1 \ldots K$ domains as

$$q^{\text{dot}}_{i,k} = h_i^\top h^{(k)} \quad (7)$$

in the simplest case. We also experiment with the bilinear function

$$q^{\text{bi}}_{i,k} = h_i^\top B h^{(k)} \quad (8)$$

where B is an additional model parameter, and also the feedforward function

$$q^{\text{feed}}_{i,k} = W \tanh\left(U h_i^\top + V h^{(k)} + b^1\right) + b^2 \quad (9)$$

where U, V, W, b^1, b^2 are additional model parameters. The final attention weights $a_i^{(1)} \ldots a_i^{(1)}$ are obtained by using a softmax layer

$$a_{i,k} = \frac{\exp(q_{i,k})}{\sum_{k=1}^{K} \exp(q_{i,k})} \quad (10)$$

The weighted combination of the experts' feedback is given by

$$h_i^{\text{experts}} = \sum_{k=1}^{K} a_{i,k} h_i^{(k)} \quad (11)$$

and the model makes predictions by using $\bar{h}_1 \ldots \bar{h}_n$ where

$$\bar{h}_i = h_i \oplus h_i^{\text{experts}} \tag{12}$$

These vectors replace the original feature vectors h_i in defining the intent or tagging losses.

4.2 Domain Attention Variants

We also consider two variants of the domain attention architecture in Section 4.1.

Label Embedding In addition to the state vectors $h^{(1)} \ldots h^{(K)}$ produced by K experts, we further incorporate their final (discrete) label predictions using pre-trained label embeddings. We induce embeddings e^y for labels y from all domains using the method of Kim et al. (2015d). At the i-th word, we predict the most likely label $y^{(k)}$ under the k-th expert and compute an attention weight as

$$\bar{q}_{i,k}^{\text{dot}} = h_i^\top e^{y^{(k)}} \tag{13}$$

Then we compute an expectation over the experts' predictions

$$\bar{a}_{i,k} = \frac{\exp(\bar{q}_{i,k})}{\sum_{k=1}^{K} \exp(\bar{q}_{i,k})} \tag{14}$$

$$h_i^{\text{label}} = \sum_{k=1}^{K} \bar{a}_{i,k} e_i^{y^{(k)}} \tag{15}$$

and use it in conjunction with $\bar{h}_i$. Note that this makes the objective a function of discrete decision and thus non-differentiable, but we can still optimize it in a standard way treating it as learning a stochastic policy.

Selective Attention Instead of computing attention over all K experts, we only consider the top $K' \leq K$ that predict the highest label scores. We only compute attention over these K' vectors. We experiment with various values of K'

5 Experiments

In this section, we describe the set of experiments conducted to evaluate the performance of our model. In order to fully assess the contribution of our approach, we also consider several baselines and variants besides our primary expert model.

Domain	$\lvert I\rvert$	$\lvert S\rvert$	Description
EVENTS	10	12	Buy event tickets
FITNESS	10	9	Track health
M-TICKET	8	15	Buy movie tickets
ORDERPIZZA	19	27	Order pizza
REMINDER	19	20	Remind task
TAXI	8	13	Find/book an cab
TV	7	5	Control TV

Table 1: The number of intent types ($|I|$), the number of slot types ($|S|$), and a short description of the test domains.

	Overlapping	
Domain	Intents	Slots
EVENTS	70.00%	75.00%
FITNESS	30.00%	77.78%
M-TICKET	37.50%	100.00%
ORDERPIZZA	47.37%	74.07%
REMINDER	68.42%	85.00%
TAXI	50.00%	100.00%
TV	57.14%	60.00%
AVG	51.49%	81.69%

Table 2: The overlapping percentage of intent types and slot types with experts or source domains.

5.1 Test domains and Tasks

To test the effectiveness of our proposed approach, we apply it to a suite of 7 Microsoft Cortana domains with 2 separate tasks in spoken language understanding: (1) intent classification and (2) slot (label) tagging. The intent classification task is a multi-class classification problem with the goal of determining to which one of the $|I|$ intents a user utterance belongs within a given domain. The slot tagging task is a sequence labeling problem with the goal of identifying entities and chunking of useful information snippets in a user utterance. For example, a user could say "*reserve a table at joeys grill for thursday at seven pm for five people*". Then the goal of the first task would be to classify this utterance as "*make_reservation*" intent given the places domain, and the goal of the second task would be to tag "*joeys grill*" as `restaurant`, "*thursday*" as `date`, "*seven pm*" as `time`, and "*five*" as `number_people`.

The short descriptions on the 7 test domains are shown in Table 1. As the table shows, the test domains have different granularity and diverse semantics. For each personal assistant test domain,

we only used 1000 training utterances to simulate scarcity of newly labeled data. The amount of development and test utterance was 100 and 10k respectively.

The similarities of test domains, represented by overlapping percentage, with experts or source domains are represented in Table 2. The intent overlapping percentage ranges from 30% on FITNESS domain to 70% on EVENTS, which averages out at 51.49%. And the slots for test domains overlaps more with those of source domains ranging from 60% on TV domain to 100% on both M-TICKET and TAXI domains, which averages out at 81.69%.

5.2 Experimental Setup

Category	$\|D\|$	Example
Trans.	4	BUS, FLIGHT
Time	4	ALARM, CALENDAR
Media	5	MOVIE, MUSIC
Action	5	HOMEAUTO, PHONE
Loc.	3	HOTEL, BUSINESS
Info	4	WEATHER, HEALTH
TOTAL	25	

Table 3: Overview of experts or source domains: Domain categories which have been created based on the label embeddings. These categorizations are solely for the purpose of describing domains because of the limited space and they are completely unrelated to the model. The number of sentences in each domain is in the range of 50k to 660k and the number of unique intents and slots are 200 and 500 respectively. In total, we have 25 domain-specific expert models. For the average performance, intent accuracy is 98% and slot F1 score is 96%.

In testing our approach, we consider a domain adaptation (DA) scenario, where a target domain has a limited training data and the source domain has a sufficient amount of labeled data. We further consider a scenario, creating a new virtual domain targeting a specific scenario given a large inventory of intent and slot types and underlying models build for many different applications and scenarios. One typical use case is that of building natural language capability through intent and slot modeling (without actually building a domain classifier) targeting a specific application. Therefore, our experimental settings are rather different from previously considered settings for domain adaptation in two aspects:

- *Multiple source domains:* In most previous works, only a pair of domains (source vs. target) have been considered, although they can be easily generalized to $K > 2$. Here, we experiment with $K = 25$ domains shown in Table 3.

- *Variant output:* In a typical setting for domain adaptation, the label space is invariant across all domains. Here, the label space can be different in different domains, which is a more challenging setting. See Kim et al. (2015d) for details of this setting.

For this DA scenario, we test whether our approach can effectively make a system to quickly generalize to a new domain with limited supervision given K existing domain experts shown in 3 .

In summary, our approach is tested with 7 Microsoft Cortana personal assistant domains across 2 tasks of intent classification and slot tagging. Below shows more detail of our baselines and variants used in our experiments.

Baselines: All models below use same underlying architecture described in Section 3.1

- TARGET: a model trained on a targeted domain without DA techniques.

- UNION: a model trained on the union of a targeted domain and 25 domain experts.

- DA: a neural domain adaptation method of Kim et al. (2016c) which trains domain specific K LSTMs with a generic LSTM on all domain training data.

Domain Experts (DE) variants: All models below are based on attending on an ensemble of 25 domain experts (DE) described in Section 4.1, where a specific set of intent and slots models are trained for each domain. We have two feedback from domain experts: (1) feature representation from LSTM, and (2) label embedding from feedfoward described in Section 4.1 and Section 4.2, respectively.

- DE^B: DE without domain attention mechanism. It uses the unweighted combination of first feedback from experts like bag-of-word model.

- DE^1: DE with domain attention with the weighted combination of the first feedbacks from experts.
- DE^2: DE^1 with additional weighted combination of second feedbacks.
- DE^{S2}: DE^2 with selected attention mechanism, described in Section 4.2.

In our experiments, all the models were implemented using Dynet (Neubig et al., 2017) and were trained using Stochastic Gradient Descent (SGD) with Adam (Kingma and Ba, 2015)—an adaptive learning rate algorithm. We used the initial learning rate of 4×10^{-4} and left all the other hyper parameters as suggested in Kingma and Ba (2015). Each SGD update was computed without a minibatch with Intel MKL (Math Kernel Library)[4]. We used the dropout regularization (Srivastava et al., 2014) with the keep probability of 0.4 at each LSTM layer.

To encode user utterances, we used bidirectional LSTMs (BiLSTMs) at the character level and the word level, along with 25 dimensional character embedding and 100 dimensional word embedding. The dimension of both the input and output of the character LSTMs were 25, and the dimensions of the input and output of the word LSTMs were 150[5] and 100, respectively. The dimension of the input and output of the final feedforward network for intent, and slot were 200 and the number of their corresponding task. Its activation was rectified linear unit (ReLU).

To initialize word embedding, we used word embedding trained from (Lample et al., 2016). In the following sections, we report intent classification results in accuracy percentage and slot results in F1-score. To compute slot F1-score, we used the standard CoNLL evaluation script[6]

5.3 Results

We show our results in the DA setting where we had a sufficient labeled dataset in the 25 source domains shown in Table 3, but only 1000 labeled data in the target domain. The performance of the baselines and our domain experts DE variants are shown in Table 4. The top half of the table shows the results of intent classification and the results of slot tagging is in the bottom half.

The baseline which trained only on the target domain (TARGET) shows a reasonably good performance, yielding on average 87.7% on the intent classification and 83.9% F1-score on the slot tagging. Simply training a single model with aggregated utterance across all domains (UNION) brings the performance down to 77.4% and 75.3%. Using DA approach of Kim et al. (2016c) shows a significant increase in performance in all 7 domains, yielding on average 90.3% intent accuracy and 86.2%.

The DE without domain attention (DE^B) shows similar performance compared to DA. Using DE model with domain attention (DE^1) shows another increase in performance, yielding on average 90.9% intent accuracy and 86.9%. The performance increases again when we use both feature representation and label embedding (DE^2), yielding on average 91.4% and 88.2% and observe nearly 93.6% and 89.1% when using selective attention (DE^{S2}). Note that DE^{S2} selects the appropriate number of experts per layer by evaluation on a development set.

The results show that our expert variant approach (DE^{S2}) achieves a significant performance gain in all 7 test domains, yielding average error reductions of 47.97% for intent classification and 32.30% for slot tagging. The results suggest that our expert approach can quickly generalize to a new domain with limited supervision given K existing domains by having only a handful more data of 1k newly labeled data points. The poor performance of using the union of both source and target domain data might be due to the relatively very small size of the target domain data, overwhelmed by the data in the source domain. For example, a word such as "*home*" can be labeled as place_type under the TAXI domain, but in the source domains can be labeled as either home_screen under the PHONE domain or contact_name under the CALENDAR domain.

5.4 Training Time

The Figure 3 shows the time required for training DE^{S2} and DA of Kim et al. (2016c). The training time for DE^{S2} stays almost constant as the number of source domains increases. However, the training time for DA grows exponentially in the number of source domains. Specifically, when trained

[4]https://software.intel.com/en-us/articles/intelr-mkl-and-c-template-libraries

[5]We concatenated last two outputs from the character LSTM and word embedding, resulting in 150 (25+25+100)

[6]http://www.cnts.ua.ac.be/conll2000/chunking/output.html

Task	Domain	TARGET	UNION	DA	DE^{B}	DE^{1}	DE^{2}	DE^{S2}
Intent	EVENTS	88.3	78.5	89.9	93.1	92.5	92.7	**94.5**
	FITNESS	88.0	77.7	92.0	92.0	91.2	91.8	**94.0**
	M-TICKET	88.2	79.2	91.9	94.4	91.5	92.7	**93.4**
	ORDERPIZZA	85.8	76.6	87.8	89.3	89.4	90.8	**92.8**
	REMINDER	87.2	76.3	91.2	90.0	90.5	90.2	**93.1**
	TAXI	87.3	76.8	89.3	89.9	89.6	89.2	**93.7**
	TV	88.9	76.4	90.3	81.5	91.5	92.0	**94.0**
	AVG	87.7	77.4	90.3	90.5	90.9	91.4	**93.6**
Slot	EVENTS	84.8	76.1	87.1	87.4	88.1	89.4	**90.2**
	FITNESS	84.0	75.6	86.4	86.3	87.0	88.1	**88.9**
	M-TICKET	84.2	75.6	86.4	86.1	86.8	88.4	**89.7**
	ORDERPIZZA	82.3	73.6	84.2	84.4	85.0	86.3	**87.1**
	REMINDER	83.5	75.0	85.9	86.3	87.0	88.3	**89.2**
	TAXI	83.0	74.6	85.6	85.5	86.3	87.5	**88.6**
	TV	85.4	76.7	87.7	87.6	88.3	89.3	**90.1**
	AVG	83.9	75.3	86.2	86.2	86.9	88.2	**89.1**

Table 4: Intent classification accuracy (%) and slot tagging F1-score (%) of our baselines and variants of DE. The numbers in boldface indicate the best performing methods.

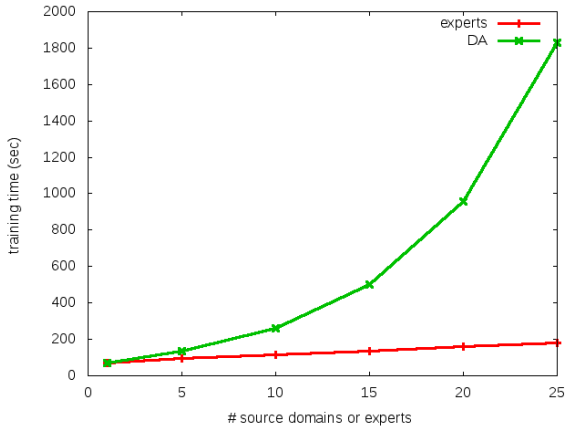


Figure 3: Comparison of training time between our DE^{S2} model and DA model of Kim et al. (2016c) as the number of domains increases. The horizontal axis means the number of domains, the vertical axis is training time per epoch in seconds. Here we use CALENDAR as the target domain, which has 1k training data.

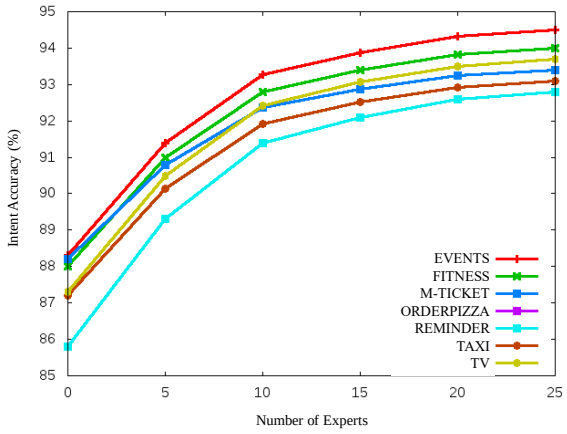


Figure 4: Learning curves in accuracy across all seven test domains as the number of expert domains increases.

with 1 source or expert domain, both took around a minute per epoch on average. When training with full 25 source domains, DE^{S2} took 3 minutes per epoch while DA took 30 minutes per epoch. Since we need to iterate over all 25+1 domains to re-train the global model, the net training time ratio could be over 250.

5.5 Learning Curve

We also measured the performance of our methods as a function of the number of domain experts. For each test domain, we consider all possible sizes of experts ranging from 1 to 25 and we then take the average of the resulting performances obtained from the expert sets of all different sizes. Figure 4 shows the resulting learning curves for each test domain. The overall trend is clear: as the more expert domains are added, the more the test performance improves. With ten or more expert domains added, our method starts to get saturated achiev-

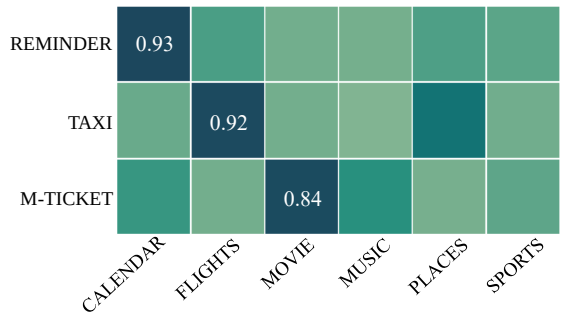


Figure 5: Heatmap visualizing attention weights.

ing more than 90% in accuracy across all seven domains.

5.6 Attention weights

From the heatmap shown in Figure 5, we can see that the attention strength generally agrees with common sense. For example, the M-TICKET and TAXI domain selected MOVIE and PLACES as their top experts, respectively.

5.7 Oracle Expert

Domain	TARGET	DE^2	Top 1
ALARM	70.1	**98.2**	ALARM (.99)
HOTEL	65.2	**96.9**	HOTEL (.99)

Table 5: Intent classification accuracy with an oracle expert in the expert pool.

The results in Table 5 show the intent classification accuracy of DE^2 when we already have the same domain expert in the expert pool. To simulate such a situation, we randomly sampled 1,000, 100, and 100 utterances from each domain as training, development and test data, respectively. In both ALARM and HOTEL domains, the trained models only on the 1,000 training utterances (TARGET) achieved only 70.1%and 65.2% in accuracy, respectively. Whereas, with our method (DE^2) applied, we reached almost the full training performance by selectively paying attention to the oracle expert, yielding 98.2% and 96.9%, respectively. This result again confirms that the behavior of the trained attention network indeed matches the semantic closeness between different domains.

5.8 Selective attention

The results in Table 6 examines how the intent prediction accuracy of DE^{S2} varies with respect to the number of experts in the pool. The rationale behind DE^{S2} is to alleviate the downside of soft attention, namely distributing probability mass over all items even if some are bad items. To deal with such issues, we apply a hard cut-off at top k domains. From the result, a threshold at top 3 or 5 yielded better results than that of either 1 or 25 experts. This matches our common sense that their are only a few of domains that are close enough to be of help to a test domain. Thus it is advisable to find the optimal k value through several rounds of experiments on a development dataset.

Domain	Top 1	Top 3	Top 5	Top 25
EVENTS	98.1	98.8	**99.2**	96.4
TV	81.4	**82.0**	81.7	80.9
AVG	89.8	90.4	**90.5**	88.7

Table 6: Accuracies of DE^{S2} using different number of experts.

6 Conclusion

In this paper, we proposed a solution for scaling domains and experiences potentially to a large number of use cases by reusing existing data labeled for different domains and applications. Our solution is based on attending an ensemble of domain experts. When given a new domain, our model uses a weighted combination of domain experts' feedback along with its own opinion to make prediction on the new domain. In both intent classification and slot tagging tasks, the model significantly outperformed baselines that do not use domain adaptation and also performed better than the full re-training approach. This approach enables creation of new virtual domains through a weighted combination of domain experts' feedback reducing the need to collect and annotate the similar intent and slot types multiple times for different domains. Future work can include an extension of domain experts to take into account dialog history aiming for a holistic framework that can handle contextual interpretation as well.

References

Tasos Anastasakos, Young-Bum Kim, and Anoop Deoras. 2014. Task specific continuous word representations for mono and multi-lingual spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, pages 3246–3250.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*. volume 7, pages 440–447.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 120–128.

Asli Celikyilmaz, Ruhi Sarikaya, Minwoo Jeong, and Anoop Deoras. 2016. An empirical investigation of word class-based features for natural language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24(6):994–1005.

Asli Celikyilmaz, Silicon Valley, and Dilek Hakkani-Tur. 2010. Convolutional neural network based semantic tagging with entity embeddings. *genre* .

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, pages 6045–6049.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815* .

Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research* 26:101–126.

Ali El-Kahky, Derek Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2014. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. IEEE, Proceedings of the ICASSP.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015. Zero-shot semantic parser for spoken language understanding. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Alex Graves. 2012. Neural networks. In *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer, pages 15–35.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pages 554–559.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*.

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

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain adaptation of recurrent neural networks for natural language understanding. *arXiv preprint arXiv:1604.00117* .

Minwoo Jeong and Gary Geunbae Lee. 2008. Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing* 16(7):1287–1302.

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015a. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *Proceedings of the NAACL*. Association for Computational Linguistics.

Young-Bum Kim, Alexandre Rochette, and Ruhi Sarikaya. 2016a. Natural language model reusability for scaling to different domains. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics* .

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Adversarial adaptation of synthetic or stale data. In *Annual Meeting of the Association for Computational Linguistics*.

Young-Bum Kim, Karl Stratos, Xiaohu Liu, and Ruhi Sarikaya. 2015b. Compact lexicon selection with spectral methods. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2015c. Pre-training of hidden-unit crfs. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 192–198.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016b. Domainless adaptation by constrained decoding on a schema lattice. *Proceedings of the 26th International Conference on Computational Linguistics (COLING)* .

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016c. Frustratingly easy neural domain adaptation. *Proceedings of the 26th International Conference on Computational Linguistics (COLING)* .

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016d. Scalable semi-supervised query classification using matrix sketching. In *The 54th Annual Meeting of the Association for Computational Linguistics*. page 8.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015d. New transfer learning techniques for disparate label sets. *ACL. Association for Computational Linguistics* .

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*. .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* .

Xiao Li, Ye-Yi Wang, and Alex Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*.

Bing Liu and Ian Lane. 2016a. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016*. pages 685–689.

Bing Liu and Ian Lane. 2016b. Joint online spoken language understanding and language modeling with recurrent neural networks. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Los Angeles.

Xiaohu Liu and Ruhi Sarikaya. 2014. A discriminative model based entity dictionary weighting approach for spoken language understanding. In *Spoken Language Technology Workshop (SLT)*. IEEE, pages 195–199.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks* 22(2):199–210.

Ruhi Sarikaya. 2015. *The technology powering personal digital assistants*. Keynote at Interspeech, Dresden, Germany.

Ruhi Sarikaya, Asli Celikyilmaz, Anoop Deoras, and Minwoo Jeong. 2014. Shrinkage based features for slot tagging with conditional random fields. In *INTERSPEECH*. pages 268–272.

Ruhi Sarikaya, Paul Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiuahu Liu, et al. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *IEEE Workshop on Spoken Language Technology*.

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

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Gokhan Tur. 2006. Multitask learning for spoken language understanding. In *In Proceedings of the ICASSP*. Toulouse, France.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, pages 78–83.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. IJCAI.

Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders

Tiancheng Zhao, Ran Zhao and Maxine Eskenazi
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
{tianchez,ranzhao1,max+}@cs.cmu.edu

Abstract

While recent neural encoder-decoder models have shown great promise in modeling open-domain conversations, they often generate dull and generic responses. Unlike past work that has focused on diversifying the output of the decoder at word-level to alleviate this problem, we present a novel framework based on conditional variational autoencoders that captures the discourse-level diversity in the encoder. Our model uses latent variables to learn a distribution over potential conversational intents and generates diverse responses using only greedy decoders. We have further developed a novel variant that is integrated with linguistic prior knowledge for better performance. Finally, the training procedure is improved by introducing a *bag-of-word* loss. Our proposed models have been validated to generate significantly more diverse responses than baseline approaches and exhibit competence in discourse-level decision-making.

1 Introduction

The dialog manager is one of the key components of dialog systems, which is responsible for modeling the decision-making process. Specifically, it typically takes a new utterance and the dialog context as input, and generates discourse-level decisions (Bohus and Rudnicky, 2003; Williams and Young, 2007). Advanced dialog managers usually have a list of potential actions that enable them to have diverse behavior during a conversation, e.g. different strategies to recover from non-understanding (Yu et al., 2016). However, the conventional approach of designing a dialog manager (Williams and Young, 2007) does not scale well to open-domain conversation models because of the vast quantity of possible decisions. Thus, there has been a growing interest in applying encoder-decoder models (Sutskever et al., 2014) for modeling open-domain conversation (Vinyals and Le, 2015; Serban et al., 2016a). The basic approach treats a conversation as a transduction task, in which the dialog history is the source sequence and the next response is the target sequence. The model is then trained end-to-end on large conversation corpora using the maximum-likelihood estimation (MLE) objective without the need for manual crafting.

However recent research has found that encoder-decoder models tend to generate generic and dull responses (e.g., *I don't know*), rather than meaningful and specific answers (Li et al., 2015; Serban et al., 2016b). There have been many attempts to explain and solve this limitation, and they can be broadly divided into two categories (see Section 2 for details): (1) the first category argues that the dialog history is only one of the factors that decide the next response. Other features should be extracted and provided to the models as conditionals in order to generate more specific responses (Xing et al., 2016; Li et al., 2016a); (2) the second category aims to improve the encoder-decoder model itself, including decoding with beam search and its variations (Wiseman and Rush, 2016), encouraging responses that have long-term payoff (Li et al., 2016b), etc.

Building upon the past work in dialog managers and encoder-decoder models, the key idea of this paper is to model dialogs as a *one-to-many* problem at the discourse level. Previous studies indicate that there are many factors in open-domain dialogs that decide the next response, and it is non-trivial to extract all of them. Intuitively, given a similar dialog history (and other observed inputs), there may exist many valid responses (at the

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 654–664
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1061

discourse-level), each corresponding to a certain configuration of the latent variables that are not presented in the input. To uncover the potential responses, we strive to model a probabilistic distribution over the distributed utterance embeddings of the potential responses using a latent variable (Figure 1). This allows us to generate diverse responses by drawing samples from the learned distribution and reconstruct their words via a decoder neural network.

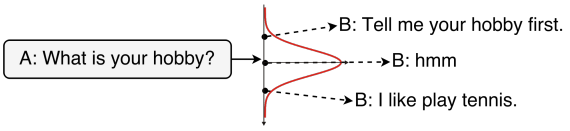


Figure 1: Given A's question, there exists many valid responses from B for different assumptions of the latent variables, e.g., B's hobby.

Specifically, our contributions are three-fold: 1. We present a novel neural dialog model adapted from conditional variational autoencoders (CVAE) (Yan et al., 2015; Sohn et al., 2015), which introduces a latent variable that can capture discourse-level variations as described above 2. We propose Knowledge-Guided CVAE (kgCVAE), which enables easy integration of expert knowledge and results in performance improvement and model interpretability. 3. We develop a training method in addressing the difficulty of optimizing CVAE for natural language generation (Bowman et al., 2015). We evaluate our models on human-human conversation data and yield promising results in: (a) generating appropriate and discourse-level diverse responses, and (b) showing that the proposed training method is more effective than the previous techniques.

2 Related Work

Our work is related to both recent advancement in encoder-decoder dialog models and generative models based on CVAE.

2.1 Encoder-decoder Dialog Models

Since the emergence of the neural dialog model, the problem of output diversity has received much attention in the research community. Ideal output responses should be both coherent and diverse. However, most models end up with generic and dull responses. To tackle this problem, one line of research has focused on augmenting the input of encoder-decoder models with richer context information, in order to generate more specific responses. Li et al., (2016a) captured speakers' characteristics by encoding background information and speaking style into the distributed embeddings, which are used to re-rank the generated response from an encoder-decoder model. Xing et al., (2016) maintain topic encoding based on Latent Dirichlet Allocation (LDA) (Blei et al., 2003) of the conversation to encourage the model to output more topic coherent responses.

On the other hand, many attempts have also been made to improve the architecture of encoder-decoder models. Li et al,. (2015) proposed to optimize the standard encoder-decoder by maximizing the mutual information between input and output, which in turn reduces generic responses. This approach penalized unconditionally high frequency responses, and favored responses that have high conditional probability given the input. Wiseman and Rush (2016) focused on improving the decoder network by alleviating the biases between training and testing. They introduced a search-based loss that directly optimizes the networks for beam search decoding. The resulting model achieves better performance on word ordering, parsing and machine translation. Besides improving beam search, Li et al., (2016b) pointed out that the MLE objective of an encoder-decoder model is unable to approximate the real-world goal of the conversation. Thus, they initialized a encoder-decoder model with MLE objective and leveraged reinforcement learning to fine tune the model by optimizing three heuristic rewards functions: informativity, coherence, and ease of answering.

2.2 Conditional Variational Autoencoder

The variational autoencoder (VAE) (Kingma and Welling, 2013; Rezende et al., 2014) is one of the most popular frameworks for image generation. The basic idea of VAE is to encode the input x into a probability distribution z instead of a point encoding in the autoencoder. Then VAE applies a decoder network to reconstruct the original input using samples from z. To generate images, VAE first obtains a sample of z from the prior distribution, e.g. $\mathcal{N}(0, \mathbf{I})$, and then produces an image via the decoder network. A more advanced model, the conditional VAE (CVAE), is a recent modification of VAE to generate diverse images conditioned on certain attributes, e.g. generating different human faces given skin color (Yan et al., 2015; Sohn et al., 2015). Inspired by CVAE, we view the dialog contexts as the conditional attributes and adapt CVAE

to generate diverse responses instead of images.

Although VAE/CVAE has achieved impressive results in image generation, adapting this to natural language generators is non-trivial. Bowman et al., (2015) have used VAE with Long-Short Term Memory (LSTM)-based recognition and decoder networks to generate sentences from a latent Gaussian variable. They showed that their model is able to generate diverse sentences with even a greedy LSTM decoder. They also reported the difficulty of training because the LSTM decoder tends to ignore the latent variable. We refer to this issue as the *vanishing latent variable problem*. Serban et al., (2016b) have applied a latent variable hierarchical encoder-decoder dialog model to introduce utterance-level variations and facilitate longer responses. To improve upon the past models, we firstly introduce a novel mechanism to leverage linguistic knowledge in training end-to-end neural dialog models, and we also propose a novel training technique that mitigates the vanishing latent variable problem.

3 Proposed Models

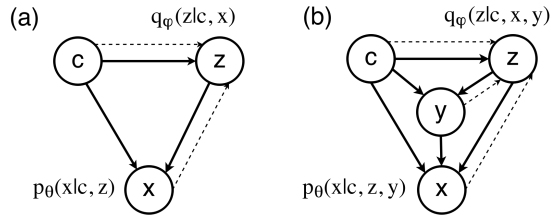


Figure 2: Graphical models of CVAE (a) and kgCVAE (b)

3.1 Conditional Variational Autoencoder (CVAE) for Dialog Generation

Each dyadic conversation can be represented via three random variables: the dialog context c (context window size $k-1$), the response utterance x (the k^{th} utterance) and a latent variable z, which is used to capture the latent distribution over the valid responses. Further, c is composed of the dialog history: the preceding k-1 utterances; conversational floor (1 if the utterance is from the same speaker of x, otherwise 0) and meta features m (e.g. the topic). We then define the conditional distribution $p(x, z|c) = p(x|z, c)p(z|c)$ and our goal is to use deep neural networks (parametrized by θ) to approximate $p(z|c)$ and $p(x|z, c)$. We refer to $p_\theta(z|c)$ as the *prior network* and $p_\theta(x, |z, c)$ as the *response decoder*. Then the generative process of x is (Figure 2 (a)):

1. Sample a latent variable z from the prior network $p_\theta(z|c)$.
2. Generate x through the response decoder $p_\theta(x|z, c)$.

CVAE is trained to maximize the conditional log likelihood of x given c, which involves an intractable marginalization over the latent variable z. As proposed in (Sohn et al., 2015; Yan et al., 2015), CVAE can be efficiently trained with the *Stochastic Gradient Variational Bayes* (SGVB) framework (Kingma and Welling, 2013) by maximizing the variational lower bound of the conditional log likelihood. We assume the z follows multivariate Gaussian distribution with a diagonal covariance matrix and introduce a *recognition network* $q_\phi(z|x, c)$ to approximate the true posterior distribution $p(z|x, c)$. Sohn and et al,. (2015) have shown that the variational lower bound can be written as:

$$\begin{aligned}\mathcal{L}(\theta, \phi; x, c) &= -KL(q_\phi(z|x, c)\|p_\theta(z|c)) \\ &\quad + \mathbf{E}_{q_\phi(z|c,x)}[\log p_\theta(x|z, c)] \\ &\leq \log p(x|c)\end{aligned} \tag{1}$$

Figure 3 demonstrates an overview of our model. The utterance encoder is a bidirectional recurrent neural network (BRNN) (Schuster and Paliwal, 1997) with a gated recurrent unit (GRU) (Chung et al., 2014) to encode each utterance into fixed-size vectors by concatenating the last hidden states of the forward and backward RNN $u_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$. x is simply u_k. The context encoder is a 1-layer GRU network that encodes the preceding k-1 utterances by taking $u_{1:k-1}$ and the corresponding conversation floor as inputs. The last hidden state h^c of the context encoder is concatenated with meta features and $c = [h^c, m]$. Since we assume z follows isotropic Gaussian distribution, the recognition network $q_\phi(z|x, c) \sim \mathcal{N}(\mu, \sigma^2\mathbf{I})$ and the prior network $p_\theta(z|c) \sim \mathcal{N}(\mu', \sigma'^2\mathbf{I})$, and then we have:

$$\begin{bmatrix}\mu \\ \log(\sigma^2)\end{bmatrix} = W_r \begin{bmatrix}x \\ c\end{bmatrix} + b_r \tag{2}$$

$$\begin{bmatrix}\mu' \\ \log(\sigma'^2)\end{bmatrix} = \text{MLP}_p(c) \tag{3}$$

We then use the reparametrization trick (Kingma and Welling, 2013) to obtain samples of z either

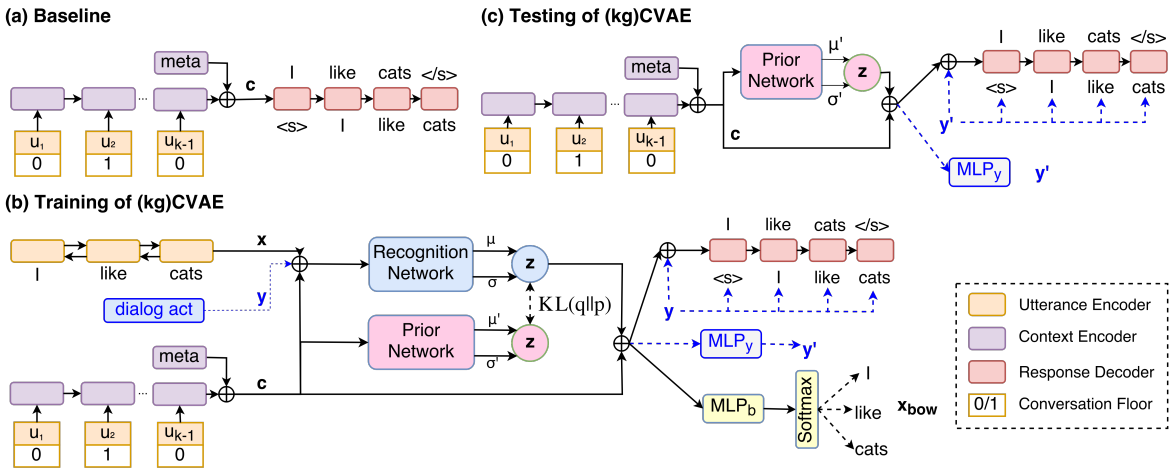


Figure 3: The neural network architectures for the baseline and the proposed CVAE/kgCVAE models. $\bigoplus$ denotes the concatenation of the input vectors. The dashed blue connections only appear in kgCVAE.

from $\mathcal{N}(z; \mu, \sigma^2\mathbf{I})$ predicted by the recognition network (training) or $\mathcal{N}(z; \mu', \sigma'^2\mathbf{I})$ predicted by the prior network (testing). Finally, the response decoder is a 1-layer GRU network with initial state $s_0 = W_i[z, c]+b_i$. The response decoder then predicts the words in x sequentially.

3.2 Knowledge-Guided CVAE (kgCVAE)

In practice, training CVAE is a challenging optimization problem and often requires large amount of data. On the other hand, past research in spoken dialog systems and discourse analysis has suggested that many linguistic cues capture crucial features in representing natural conversation. For example, dialog acts (Poesio and Traum, 1998) have been widely used in the dialog managers (Litman and Allen, 1987; Raux et al., 2005; Zhao and Eskenazi, 2016) to represent the propositional function of the system. Therefore, we conjecture that it will be beneficial for the model to learn meaningful latent z if it is provided with explicitly extracted discourse features during the training.

In order to incorporate the linguistic features into the basic CVAE model, we first denote the set of linguistic features as y. Then we assume that the generation of x depends on c, z and y. y relies on z and c as shown in Figure 2. Specifically, during training the initial state of the response decoder is $s_0 = W_i[z, c, y] + b_i$ and the input at every step is $[e_t, y]$ where e_t is the word embedding of t^{th} word in x. In addition, there is an MLP to predict $y' = \text{MLP}_y(z, c)$ based on z and c. In the testing stage, the predicted y' is used by the response decoder instead of the oracle decoders. We denote the modified model as knowledge-guided CVAE (kgCVAE) and developers can add desired discourse features that they wish the latent variable z to capture. KgCVAE model is trained by maximizing:

$$\begin{aligned}\mathcal{L}(\theta, \phi; x, c, y) = &-KL(q_\phi(z|x, c, y)\|P_\theta(z|c)) \\ &+ \mathbf{E}_{q_\phi(z|c,x,y)}[\log p(x|z, c, y)] \\ &+ \mathbf{E}_{q_\phi(z|c,x,y)}[\log p(y|z, c)] \quad (4)\end{aligned}$$

Since now the reconstruction of y is a part of the loss function, kgCVAE can more efficiently encode y-related information into z than discovering it only based on the surface-level x and c. Another advantage of kgCVAE is that it can output a high-level label (e.g. dialog act) along with the word-level responses, which allows easier interpretation of the model's outputs.

3.3 Optimization Challenges

A straightforward VAE with RNN decoder fails to encode meaningful information in z due to the *vanishing latent variable problem* (Bowman et al., 2015). Bowman et al., (2015) proposed two solutions: (1) *KL annealing*: gradually increasing the weight of the KL term from 0 to 1 during training; (2) *word drop decoding*: setting a certain percentage of the target words to 0. We found that CVAE suffers from the same issue when the decoder is an RNN. Also we did not consider word drop decoding because Bowman et al,. (2015) have shown that it may hurt the performance when the drop rate is too high.

As a result, we propose a simple yet novel technique to tackle the vanishing latent variable problem: *bag-of-word loss*. The idea is to introduce

an auxiliary loss that requires the decoder network to predict the bag-of-words in the response x as shown in Figure 3(b). We decompose x into two variables: x_o with word order and x_{bow} without order, and assume that x_o and x_{bow} are conditionally independent given z and c: $p(x, z|c) = p(x_o|z, c)p(x_{bow}|z, c)p(z|c)$. Due to the conditional independence assumption, the latent variable is forced to capture global information about the target response. Let $f = \text{MLP}_b(z, x) \in \mathcal{R}^V$ where V is vocabulary size, and we have:

$$\log p(x_{bow}|z, c) = \log \prod_{t=1}^{|x|} \frac{e^{f_{x_t}}}{\sum_j^V e^{f_j}} \quad (5)$$

where $|x|$ is the length of x and x_t is the word index of t_{th} word in x. The modified variational lower bound for CVAE with bag-of-word loss is (see Appendix A for kgCVAE):

$$\mathcal{L}'(\theta, \phi; x, c) = \mathcal{L}(\theta, \phi; x, c) + \mathbf{E}_{q_\phi(z|c,x,y)}[\log p(x_{bow}|z, c)] \quad (6)$$

We will show that the bag-of-word loss in Equation 6 is very effective against the vanishing latent variable and it is also complementary to the KL annealing technique.

4 Experiment Setup

4.1 Dataset

We chose the Switchboard (SW) 1 Release 2 Corpus (Godfrey and Holliman, 1997) to evaluate the proposed models. SW has 2400 two-sided telephone conversations with manually transcribed speech and alignment. In the beginning of the call, a computer operator gave the callers recorded prompts that define the desired topic of discussion. There are 70 available topics. We randomly split the data into $2316/60/62$ dialogs for train/validate/test. The pre-processing includes (1) tokenize using the NLTK tokenizer (Bird et al., 2009); (2) remove non-verbal symbols and repeated words due to false starts; (3) keep the top 10K frequent word types as the vocabulary. The final data have $207,833/5,225/5,481$ (c, x) pairs for train/validate/test. Furthermore, a subset of SW was manually labeled with dialog acts (Stolcke et al., 2000). We extracted dialog act labels based on the dialog act recognizer proposed in (Ribeiro et al., 2015). The features include the uni-gram and bi-gram of the utterance, and the contextual features of the last 3 utterances. We trained a Support Vector Machine (SVM) (Suykens and Vandewalle, 1999) with linear kernel on the subset of SW with human annotations. There are 42 types of dialog acts and the SVM achieved 77.3% accuracy on held-out data. Then the rest of SW data are labelled with dialog acts using the trained SVM dialog act recognizer.

4.2 Training

We trained with the following hyperparameters (according to the loss on the validate dataset): word embedding has size 200 and is shared across everywhere. We initialize the word embedding from Glove embedding pre-trained on Twitter (Pennington et al., 2014). The utterance encoder has a hidden size of 300 for each direction. The context encoder has a hidden size of 600 and the response decoder has a hidden size of 400. The prior network and the MLP for predicting y both have 1 hidden layer of size 400 and $tanh$ non-linearity. The latent variable z has a size of 200. The context window k is 10. All the initial weights are sampled from a uniform distribution [-0.08, 0.08]. The mini-batch size is 30. The models are trained end-to-end using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 and gradient clipping at 5. We selected the best models based on the variational lower bound on the validate data. Finally, we use the BOW loss along with *KL annealing* of 10,000 batches to achieve the best performance. Section 5.4 gives a detailed argument for the importance of the BOW loss.

5 Results

5.1 Experiments Setup

We compared three neural dialog models: a strong baseline model, CVAE, and kgCVAE. The **baseline model** is an encoder-decoder neural dialog model without latent variables similar to (Serban et al., 2016a). The baseline model's encoder uses the same context encoder to encode the dialog history and the meta features as shown in Figure 3. The encoded context c is directly fed into the decoder networks as the initial state. The hyperparameters of the baseline are the same as the ones reported in Section 4.2 and the baseline is trained to minimize the standard cross entropy loss of the decoder RNN model without any auxiliary loss.

Also, to compare the diversity introduced by the stochasticity in the proposed latent variable versus the softmax of RNN at each decoding step, we generate N responses from the baseline by sam-

pling from the softmax. For CVAE/kgCVAE, we sample N times from the latent z and only use greedy decoders so that the randomness comes entirely from the latent variable z.

5.2 Quantitative Analysis

Automatically evaluating an open-domain generative dialog model is an open research challenge (Liu et al., 2016). Following our *one-to-many* hypothesis, we propose the following metrics. We assume that for a given dialog context c, there exist M_c reference responses r_j, $j \in [1, M_c]$. Meanwhile a model can generate N hypothesis responses h_i, $i \in [1, N]$. The generalized response-level precision/recall for a given dialog context is:

$$\text{precision(c)} = \frac{\sum_{i=1}^{N} max_{j \in [1,M_c]} d(r_j, h_i)}{N}$$

$$\text{recall(c)} = \frac{\sum_{j=1}^{M_c} max_{i \in [1,N]} d(r_j, h_i))}{M_c}$$

where $d(r_j, h_i)$ is a distance function which lies between 0 to 1 and measures the similarities between r_j and h_i. The final score is averaged over the entire test dataset and we report the performance with 3 types of distance functions in order to evaluate the systems from various linguistic points of view:

1. Smoothed Sentence-level BLEU (Chen and Cherry, 2014): BLEU is a popular metric that measures the geometric mean of modified n-gram precision with a length penalty (Papineni et al., 2002; Li et al., 2015). We use BLEU-1 to 4 as our lexical similarity metric and normalize the score to 0 to 1 scale.

2. Cosine Distance of Bag-of-word Embedding: a simple method to obtain sentence embeddings is to take the average or extrema of all the word embeddings in the sentences (Forgues et al., 2014; Adi et al., 2016). The $d(r_j, h_i)$ is the cosine distance of the two embedding vectors. We used Glove embedding described in Section 4 and denote the average method as A-bow and extrema method as E-bow.

3. Dialog Act Match: to measure the similarity at the discourse level, the same dialog-act tagger from 4.1 is applied to label all the generated responses of each model. We set $d(r_j, h_i) = 1$ if r_j and h_i have the same dialog acts, otherwise $d(r_j, h_i) = 0$.

One challenge of using the above metrics is that there is only one, rather than multiple reference responses/contexts. This impacts reliability of our measures. Inspired by (Sordoni et al., 2015), we utilized information retrieval techniques (see Appendix A) to gather 10 extra candidate reference responses/context from other conversations with the same topics. Then the 10 candidate references are filtered by two experts, which serve as the ground truth to train the reference response classifier. The result is 6.69 extra references in average per context. The average number of distinct reference dialog acts is 4.2. Table 1 shows the results.

Metrics	Baseline	CVAE	kgCVAE
perplexity (KL)	35.4	20.2	16.02
	(n/a)	(11.36)	(13.08)
BLEU-1 prec	0.405	0.372	**0.412**
BLEU-1 recall	0.336	0.381	**0.411**
BLEU-2 prec	0.300	0.295	**0.350**
BLEU-2 recall	0.281	0.322	**0.356**
BLEU-3 prec	0.272	0.265	**0.310**
BLEU-3 recall	0.254	0.292	**0.318**
BLEU-4 prec	0.226	0.223	**0.262**
BLEU-4 recall	0.215	0.248	**0.272**
A-bow prec	0.387	**0.389**	0.373
A-bow recall	0.337	**0.361**	0.336
E-bow prec	0.701	0.705	**0.711**
E-bow recall	0.684	0.709	**0.712**
DA prec	**0.736**	0.704	0.721
DA recall	0.514	**0.604**	0.598

Table 1: Performance of each model on automatic measures. The highest score in each row is in bold. Note that our BLEU scores are normalized to $[0, 1]$.

The proposed models outperform the baseline in terms of recall in all the metrics with statistical significance. This confirms our hypothesis that generating responses with discourse-level diversity can lead to a more comprehensive coverage of the potential responses than promoting only word-level diversity. As for precision, we observed that the baseline has higher or similar scores than CVAE in all metrics, which is expected since the baseline tends to generate the mostly likely and safe responses repeatedly in the N hypotheses. However, kgCVAE is able to achieve the highest precision and recall in the 4 metrics at the same time (BLEU1-4, E-BOW). One reason

for kgCVAE's good performance is that the predicted dialog act label in kgCVAE can regularize the generation process of its RNN decoder by forcing it to generate more coherent and precise words. We further analyze the precision/recall of BLEU-4 by looking at the average score versus the number of distinct reference dialog acts. A low number of distinct dialog acts represents the situation where the dialog context has a strong constraint on the range of the next response (low entropy), while a high number indicates the opposite (high-entropy). Figure 4 shows that CVAE/kgCVAE achieves significantly higher recall than the baseline in higher entropy contexts. Also it shows that CVAE suffers from lower precision, especially in low entropy contexts. Finally, kgCVAE gets higher precision than both the baseline and CVAE in the full spectrum of context entropy.

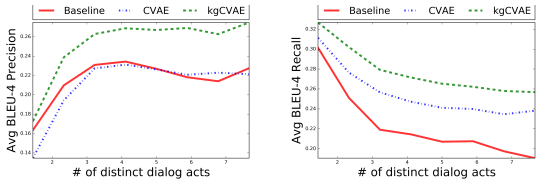


Figure 4: BLEU-4 precision/recall vs. the number of distinct reference dialog acts.

5.3 Qualitative Analysis

Table 2 shows the outputs generated from the baseline and kgCVAE. In example 1, caller A begins with an open-ended question. The kgCVAE model generated highly diverse answers that cover multiple plausible dialog acts. Further, we notice that the generated text exhibits similar dialog acts compared to the ones predicted separately by the model, implying the consistency of natural language generation based on y. On the contrary, the responses from the baseline model are limited to local n-gram variations and share a similar prefix, i.e. "I'm". Example 2 is a situation where caller A is telling B stories. The ground truth response is a back-channel and the range of valid answers is more constrained than example 1 since B is playing the role of a listener. The baseline successfully predicts "uh-huh". The kgCVAE model is also able to generate various ways of back-channeling. This implies that the latent z is able to capture context-sensitive variations, i.e. in low-entropy dialog contexts modeling lexical diversity while in high-entropy ones modeling discourse-level diversity. Moreover, kgCVAE is occasionally able to generate more sophisticated grounding (sample 4) beyond a simple back-channel, which is also an acceptable response given the dialog context.

In addition, past work (Kingma and Welling, 2013) has shown that the recognition network is able to learn to cluster high-dimension data, so we conjecture that posterior z outputted from the recognition network should cluster the responses into meaningful groups. Figure 5 visualizes the posterior z of responses in the test dataset in 2D space using t-SNE (Maaten and Hinton, 2008). We found that the learned latent space is highly correlated with the dialog act and length of responses, which confirms our assumption.

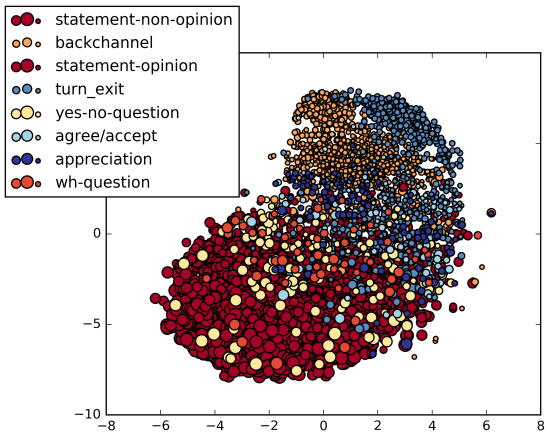


Figure 5: t-SNE visualization of the posterior z for test responses with top 8 frequent dialog acts. The size of circle represents the response length.

5.4 Results for Bag-of-Word Loss

Finally, we evaluate the effectiveness of bag-of-word (BOW) loss for training VAE/CVAE with the RNN decoder. To compare with past work (Bowman et al., 2015), we conducted the same language modelling (LM) task on Penn Treebank using VAE. The network architecture is same except we use GRU instead of LSTM. We compared four different training setups: (1) standard VAE without any heuristics; (2) VAE with KL annealing (KLA); (3) VAE with BOW loss; (4) VAE with both BOW loss and KLA. Intuitively, a well trained model should lead to a low reconstruction loss and small but non-trivial KL cost. For all models with KLA, the KL weight increases linearly from 0 to 1 in the first 5000 batches.

Table 3 shows the reconstruction perplexity and the KL cost on the test dataset. The standard VAE fails to learn a meaningful latent variable by hav-

Example 1-Topic: Recycling **Context**: A: are they doing a lot of recycling out in Georgia? **Target-B** (statement): well at my workplace we have palaces for aluminium cans and we have a separate trash can for recyclable paper	
Baseline+Sampling	**kgCVAE+Greedy**
1. well I'm a graduate student and have two kids	1. (non-understand) pardon
2. well I was in last year and so we've had lots of recycling	2. (statement) oh you're not going to have a curbside pick up here
3. I'm not sure	3. (statement) okay I am sure about a recycling center
4. well I don't know I just moved here in new york	4. (yes-answer) yeah so

Example 2-Topic: Child Care **Context**: A: you know a private home to take their children to when they're young until they hit the preschool age and they **Target-B** (backchannel): uh-huh	
Baseline+Sampling	**kgCVAE+Greedy**
1. um - hum	1. (backchannel) uh-huh
2. yeah	2. (turn-exit) um-hum
3. um - hum	3. (backchannel) yeah
4. uh-huh	4. (statement) oh yeah I think that's part of the problem

Table 2: Generated responses from the baselines and kgCVAE in two examples. KgCVAE also provides the predicted dialog act for each response. The context only shows the last utterance due to space limit (the actual context window size is 10).

ing a KL cost close to 0 and a reconstruction perplexity similar to a small LSTM LM (Zaremba et al., 2014). KLA helps to improve the reconstruction loss, but it requires early stopping since the models will fall back to the standard VAE after the KL weight becomes 1. At last, the models with BOW loss achieved significantly lower perplexity and larger KL cost.

Model	Perplexity	KL cost
Standard	122.0	0.05
KLA	111.5	2.02
BOW	97.72	7.41
BOW+KLA	73.04	15.94

Table 3: The reconstruction perplexity and KL terms on Penn Treebank test set.

Figure 6 visualizes the evolution of the KL cost. We can see that for the standard model, the KL cost crashes to 0 at the beginning of training and never recovers. On the contrary, the model with only KLA learns to encode substantial information in latent z when the KL cost weight is small. However, after the KL weight is increased to 1 (after 5000 batch), the model once again decides to ignore the latent z and falls back to the naive implementation. The model with BOW loss, however, consistently converges to a non-trivial KL cost even without KLA, which confirms the importance of BOW loss for training latent variable models with the RNN decoder. Last but not least, our experiments showed that the conclusions drawn from LM using VAE also apply to training CVAE/kgCVAE, so we used BOW loss together with KLA for all previous experiments.

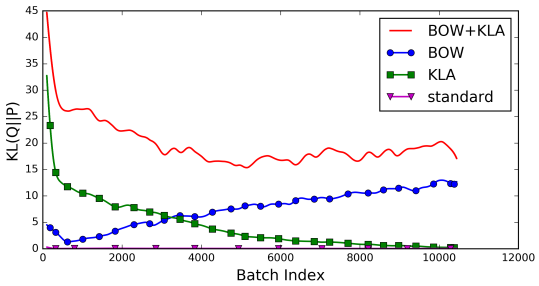


Figure 6: The value of the KL divergence during training with different setups on Penn Treebank.

6 Conclusion and Future Work

In conclusion, we identified the *one-to-many* nature of open-domain conversation and proposed two novel models that show superior performance in generating diverse and appropriate responses at the discourse level. While the current paper addresses diversifying responses in respect to dialogue acts, this work is part of a larger research direction that targets leveraging both past linguistic findings and the learning power of deep neural networks to learn better representation of the latent factors in dialog. In turn, the output of this novel neural dialog model will be easier to explain and control by humans. In addition to dialog acts, we plan to apply our kgCVAE model to capture other different linguistic phenomena including sentiment, named entities,etc. Last but not least, the recognition network in our model will serve as the foundation for designing a data-driven dialog manager, which automatically discovers useful high-level intents. All of the above suggest a promising research direction.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207* .

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. " O'Reilly Media, Inc.".

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Dan Bohus and Alexander I Rudnicky. 2003. Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda .

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* .

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. *ACL 2014* page 362.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *NIPS, Modern Machine Learning and Natural Language Processing Workshop*.

John J Godfrey and Edward Holliman. 1997. Switchboard-1 release 2. *Linguistic Data Consortium, Philadelphia* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* .

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155* .

Jiwei Li, Will Monroe, Alan Ritter, and Dan Jurafsky. 2016b. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* .

Diane J Litman and James F Allen. 1987. A plan recognition model for subdialogues in conversations. *Cognitive science* 11(2):163–200.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023* .

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–43.

Massimo Poesio and David Traum. 1998. Towards an axiomatization of dialogue acts. In *Proceedings of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogues (13th Twente Workshop on Language Technology*. Citeseer.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Lets go public! taking a spoken dialog system to the real world. In *in Proc. of Interspeech 2005*. Citeseer.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* .

Eugénio Ribeiro, Ricardo Ribeiro, and David Martins de Matos. 2015. The influence of context on dialogue act recognition. *arXiv preprint arXiv:1506.00839* .

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24(5):513–523.

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

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016a. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016b. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069* .

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*. pages 3483–3491.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics* 26(3):339–373.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters* 9(3):293–300.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language* 21(2):393–422.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* .

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2016. Topic augmented neural response generation with a joint attention mechanism. *arXiv preprint arXiv:1606.08340* .

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2015. Attribute2image: Conditional image generation from visual attributes. *arXiv preprint arXiv:1512.00570* .

Zhou Yu, Ziyu Xu, Alan W Black, and Alex I Rudnicky. 2016. Strategy and policy learning for non-task-oriented conversational systems. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. volume 2, page 7.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560* .

A Supplemental Material

Variational Lower Bound for kgCVAE

We assume that even with the presence of linguistic feature y regarding x, the prediction of x_{bow} still only depends on the z and c. Therefore, we have:

$$\begin{aligned}\mathcal{L}(\theta, \phi; x, c, y) = &-KL(q_\phi(z|x, c, y)\|P_\theta(z|c)) \\ &+ \mathbf{E}_{q_\phi(z|c,x,y)}[\log p(x|z, c, y)] \\ &+ \mathbf{E}_{q_\phi(z|c,x,y)}[\log p(y|z, c)] \\ &+ \mathbf{E}_{q_\phi(z|c,x,y)}[\log p(x_{bow}|z, c)]\end{aligned} \tag{7}$$

Collection of Multiple Reference Responses

We collected multiple reference responses for each dialog context in the test set by information retrieval techniques combining with traditional a machine learning method. First, we encode the dialog history using Term Frequency-Inverse Document Frequency (TFIDF) (Salton and Buckley, 1988) weighted bag-of-words into vector representation h. Then we denote the topic of the conversation as t and denote f as the conversation floor, i.e. if the speakers of the last utterance in the dialog history and response utterance are the same $f = 1$ otherwise $f = 0$. Then we computed the similarity $d(c_i, c_j)$ between two dialog contexts using:

$$d(c_i, c_j) = \mathbb{1}(t_i = t_j)\mathbb{1}(t_i = t_j)\frac{h_i \cdot h_j}{||h_i||||h_j||} \tag{8}$$

Unlike past work (Sordoni et al., 2015), this similarity function only cares about the distance in the context and imposes no constraints on the response, therefore is suitbale for finding diverse responses regarding to the same dialog context. Secondly, for each dialog context in the test set, we retrieved the 10 nearest neighbors from the training set and treated the responses from the training set as candidate reference responses. Thirdly, we further sampled 240 context-responses pairs from 5481 pairs in the total test set and post-processed the selected candidate responses by two human computational linguistic experts who were told to give a binary label for each candidate response about whether the response is appropriate regarding its dialog context. The filtered lists then served as the ground truth to train our reference response classifier. For the next step, we extracted bigrams, part-of-speech bigrams and word part-of-speech

pairs from both dialogue contexts and candidate reference responses with rare threshold for feature extraction being set to 20. Then L2-regularized logistic regression with 10-fold cross validation was applied as the machine learning algorithm. Cross validation accuracy on the human-labelled data was 71%. Finally, we automatically annotated the rest of test set with this trained classifier and the resulting data were used for model evaluation.

Hybrid Code Networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning

Jason D. Williams
Microsoft Research
jason.williams@microsoft.com

Kavosh Asadi
Brown University
kavosh@brown.edu

Geoffrey Zweig*
Microsoft Research
g2zweig@gmail.com

Abstract

End-to-end learning of recurrent neural networks (RNNs) is an attractive solution for dialog systems; however, current techniques are data-intensive and require thousands of dialogs to learn simple behaviors. We introduce Hybrid Code Networks (HCNs), which combine an RNN with *domain-specific knowledge encoded as software* and *system action templates*. Compared to existing end-to-end approaches, HCNs considerably reduce the amount of training data required, while retaining the key benefit of inferring a latent representation of dialog state. In addition, HCNs can be optimized with supervised learning, reinforcement learning, or a mixture of both. HCNs attain state-of-the-art performance on the bAbI dialog dataset (Bordes and Weston, 2016), and outperform two commercially deployed customer-facing dialog systems.

1 Introduction

Task-oriented dialog systems help a user to accomplish some goal using natural language, such as making a restaurant reservation, getting technical support, or placing a phonecall. Historically, these dialog systems have been built as a pipeline, with modules for language understanding, state tracking, action selection, and language generation. However, dependencies between modules introduce considerable complexity – for example, it is often unclear how to define the dialog state and what history to maintain, yet action selection relies exclusively on the state for input. Moreover, training each module requires specialized labels.

Recently, end-to-end approaches have trained recurrent neural networks (RNNs) directly on text transcripts of dialogs. A key benefit is that the RNN infers a latent representation of state, obviating the need for state labels. However, end-to-end methods lack a general mechanism for injecting domain knowledge and constraints. For example, simple operations like sorting a list of database results or updating a dictionary of entities can expressed in a few lines of software, yet may take thousands of dialogs to learn. Moreover, in some practical settings, programmed constraints are essential – for example, a banking dialog system would require that a user is logged in before they can retrieve account information.

This paper presents a model for end-to-end learning, called *Hybrid Code Networks* (HCNs) which addresses these problems. In addition to learning an RNN, HCNs also allow a developer to express domain knowledge via software and action templates. Experiments show that, compared to existing recurrent end-to-end techniques, HCNs achieve the same performance with considerably less training data, while retaining the key benefit of end-to-end trainability. Moreover, the neural network can be trained with supervised learning or reinforcement learning, by changing the gradient update applied.

This paper is organized as follows. Section 2 describes the model, and Section 3 compares the model to related work. Section 4 applies HCNs to the bAbI dialog dataset (Bordes and Weston, 2016). Section 5 then applies the method to real customer support domains at our company. Section 6 illustrates how HCNs can be optimized with reinforcement learning, and Section 7 concludes.

* Currently at JPMorgan Chase

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 665–677
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1062

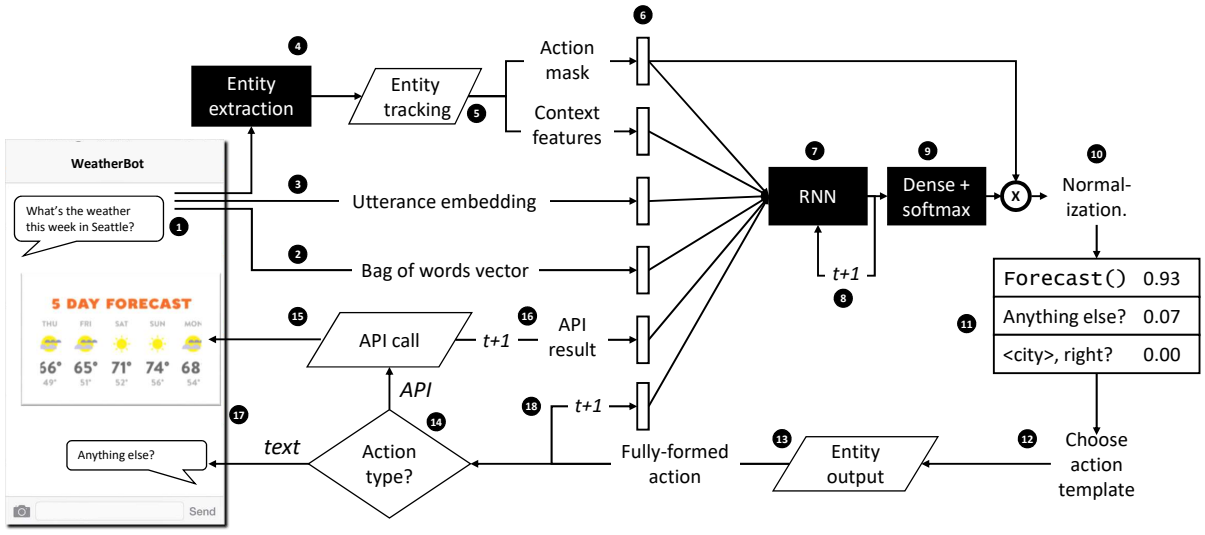


Figure 1: Operational loop. Trapezoids refer to programmatic code provided by the software developer, and shaded boxes are trainable components. Vertical bars under "6" represent concatenated vectors which form the input to the RNN.

2 Model description

At a high level, the four components of a Hybrid Code Network are a recurrent neural network; domain-specific software; domain-specific action templates; and a conventional entity extraction module for identifying entity mentions in text. Both the RNN and the developer code maintain state. Each action template can be a textual communicative action or an API call. The HCN model is summarized in Figure 1.

The cycle begins when the user provides an utterance, as text (step 1). The utterance is featurized in several ways. First, a bag of words vector is formed (step 2). Second, an utterance embedding is formed, using a pre-built utterance embedding model (step 3). Third, an entity extraction module identifies entity mentions (step 4) – for example, identifying "Jennifer Jones" as a `<name>` entity. The text and entity mentions are then passed to "Entity tracking" code provided by the developer (step 5), which grounds and maintains entities – for example, mapping the text "Jennifer Jones" to a specific row in a database. This code can optionally return an "action mask", indicating actions which are permitted at the current timestep, as a bit vector. For example, if a target phone number has not yet been identified, the API action to place a phone call may be masked. It can also optionally return "context features" which are features the developer thinks will be useful for distinguishing among actions, such as which entities are currently present and which are absent.

The feature components from steps 1-5 are concatenated to form a feature vector (step 6). This vector is passed to an RNN, such as a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) or gated recurrent unit (GRU) (Chung et al., 2014). The RNN computes a hidden state (vector), which is retained for the next timestep (step 8), and passed to a dense layer with a softmax activation, with output dimension equal to the number of distinct system action templates (step 9).[1] Thus the output of step 9 is a distribution over action templates. Next, the action mask is applied as an element-wise multiplication, and the result is normalized back to a probability distribution (step 10) – this forces non-permitted actions to take on probability zero. From the resulting distribution (step 11), an action is selected (step 12). When RL is active, exploration is required, so in this case an action is *sampled* from the distribution; when RL is not active, the best action should be chosen, and so the action with the *highest probability* is always selected.

The selected action is next passed to "Entity output" developer code that can substitute in entities (step 13) and produce a fully-formed action – for example, mapping the template "`<city>`,

[1]Implementation details for the RNN such as size, loss, etc. are given with each experiment in Sections 4-6.

right?" to "Seattle, right?". In step 14, control branches depending on the type of the action: if it is an API action, the corresponding API call in the developer code is invoked (step 15) – for example, to render rich content to the user. APIs can act as sensors and return features relevant to the dialog, so these can be added to the feature vector in the next timestep (step 16). If the action is text, it is rendered to the user (step 17), and cycle then repeats. The action taken is provided as a feature to the RNN in the next timestep (step 18).

3 Related work

Broadly there are two lines of work applying machine learning to dialog control. The first decomposes a dialog system into a pipeline, typically including language understanding, dialog state tracking, action selection policy, and language generation (Levin et al., 2000; Singh et al., 2002; Williams and Young, 2007; Williams, 2008; Hori et al., 2009; Lee et al., 2009; Griol et al., 2008; Young et al., 2013; Li et al., 2014). Specifically related to HCNs, past work has implemented the policy as feed-forward neural networks (Wen et al., 2016), trained with supervised learning followed by reinforcement learning (Su et al., 2016). In these works, the policy has not been *recurrent* – i.e., the policy depends on the state tracker to summarize observable dialog history into state features, which requires design and specialized labeling. By contrast, HCNs use an RNN which automatically infers a representation of state. For learning efficiency, HCNs use an external lightweight process for tracking entity values, but the policy is not strictly dependent on it: as an illustration, in Section 5 below, we demonstrate an HCN-based dialog system which has no external state tracker. If there is context which is not apparent in the text in the dialog, such as database status, this can be encoded as a context feature to the RNN.

The second, more recent line of work applies recurrent neural networks (RNNs) to learn "end-to-end" models, which map from an observable dialog history directly to a sequence of output words (Sordoni et al., 2015; Shang et al., 2015; Vinyals and Le, 2015; Yao et al., 2015; Serban et al., 2016; Li et al., 2016a,c; Luan et al., 2016; Xu et al., 2016; Li et al., 2016b; Mei et al., 2016; Lowe et al., 2017; Serban et al., 2017). These systems can be applied to task-oriented domains by adding special "API call" actions, enumerating database output as a sequence of tokens (Bordes and Weston, 2016), then learning an RNN using Memory Networks (Sukhbaatar et al., 2015), gated memory networks (Liu and Perez, 2016), query reduction networks (Seo et al., 2016), and copy-augmented networks (Eric and Manning, 2017). In each of these architectures, the RNN learns to manipulate entity values, for example by saving them in a memory. Output is produced by generating a sequence of tokens (or ranking all possible surface forms), which can also draw from this memory. HCNs also use an RNN to accumulate dialog state and choose actions. However, HCNs differ in that they use developer-provided action templates, which can contain entity references, such as "`<city>`, right?". This design reduce learning complexity, and also enable the software to limit which actions are available via an action mask, at the expense of developer effort. To further reduce learning complexity in a practical system, entities are tracked separately, outside the the RNN, which also allows them to be substituted into action templates. Also, past end-to-end recurrent models have been trained using supervised learning, whereas we show how HCNs can also be trained with reinforcement learning.

4 Supervised learning evaluation I

In this section we compare HCNs to existing approaches on the public "bAbI dialog" dataset (Bordes and Weston, 2016). This dataset includes two end-to-end dialog learning tasks, in the restaurant domain, called task5 and task6.[2] Task5 consists of synthetic, simulated dialog data, with highly regular user behavior and constrained vocabulary. Dialogs include a database access action which retrieves relevant restaurants from a database, with results included in the dialog transcript. We test on the "OOV" variant of Task5, which includes entity values not observed in the training set. Task6 draws on human-computer dialog data from the second dialog state tracking challenge (DSTC2), where usability subjects (crowd-workers) interacted with several variants of a spoken dialog system (Henderson et al., 2014a). Since the database from DSTC2 was not provided, database calls have been inferred from the data and inserted into the dialog transcript. Example dialogs are provided in the Appendix Sections A.2 and A.3.

To apply HCNs, we wrote simple domain-

[2]Tasks 1-4 are sub-tasks of Task5.

specific software, as follows. First, for entity extraction (step 4 in Figure 1), we used a simple string match, with a pre-defined list of entity names – i.e., the list of restaurants available in the database. Second, in the context update (step 5), we wrote simple logic for tracking entities: when an entity is recognized in the user input, it is retained by the software, over-writing any previously stored value. For example, if the `price` "cheap" is recognized in the first turn, it is retained as `price=cheap`. If "expensive" is then recognized in the third turn, it over-writes "cheap" so the code now holds `price=expensive`. Third, system actions were templatized: for example, system actions of the form "prezzo is a nice restaurant in the west of town in the moderate price range" all map to the template "`<name>` is a nice restaurant in the `<location>` of town in the `<price>` price range". This results in 16 templates for Task5 and 58 for Task6.[3] Fourth, when database results are received into the entity state, they are sorted by rating. Finally, an action mask was created which encoded common-sense dependencies. These are implemented as simple if-then rules based on the presence of entity values: for example, only allow an API call if pre-conditions are met; only offer a restaurant if database results have already been received; do not ask for an entity if it is already known; etc.

For Task6, we noticed that the system can say that no restaurants match the current query *without* consulting the database (for an example dialog, see Section A.3 in the Appendix). In a practical system this information would be retrieved from the database and not encoded in the RNN. So, we mined the training data and built a table of search queries known to yield no results. We also added context features that indicated the state of the database – for example, whether there were any restaurants matching the current query. The complete set of context features is given in Appendix Section A.4. Altogether this code consisted of about 250 lines of Python.

We then trained an HCN on the training set, employing the domain-specific software described above. We selected an LSTM for the recurrent layer (Hochreiter and Schmidhuber, 1997), with the AdaDelta optimizer (Zeiler, 2012). We used the development set to tune the number of hidden units (128), and the number of epochs (12). Utterance embeddings were formed by averaging word embeddings, using a publicly available 300-dimensional word embedding model trained using word2vec on web data (Mikolov et al., 2013).[4] The word embeddings were static and not updated during LSTM training. In training, each dialog formed one minibatch, and updates were done on full rollouts (i.e., non-truncated back propagation through time). The training loss was categorical cross-entropy. Further low-level implementation details are in the Appendix Section A.1.

We ran experiments with four variants of our model: with and without the utterance embeddings, and with and without the action mask (Figure 1, steps 3 and 6 respectively).

Following past work, we report average turn accuracy – i.e., for each turn in each dialog, present the (true) history of user and system actions to the network and obtain the network's prediction as a string of characters. The turn is correct if the string matches the reference exactly, and incorrect if not. We also report dialog accuracy, which indicates if all turns in a dialog are correct.

We compare to four past end-to-end approaches (Bordes and Weston, 2016; Liu and Perez, 2016; Eric and Manning, 2017; Seo et al., 2016). We emphasize that past approaches have applied purely sequence-to-sequence models, or (as a baseline) purely programmed rules (Bordes and Weston, 2016). By contrast, Hybrid Code Networks are a hybrid of hand-coded rules and learned models.

Results are shown in Table 1. Since Task5 is synthetic data generated using rules, it is possible to obtain perfect accuracy using rules (line 1). The addition of domain knowledge greatly simplifies the learning task and enables HCNs to also attain perfect accuracy. On Task6, rules alone fare poorly, whereas HCNs outperform past learned models.

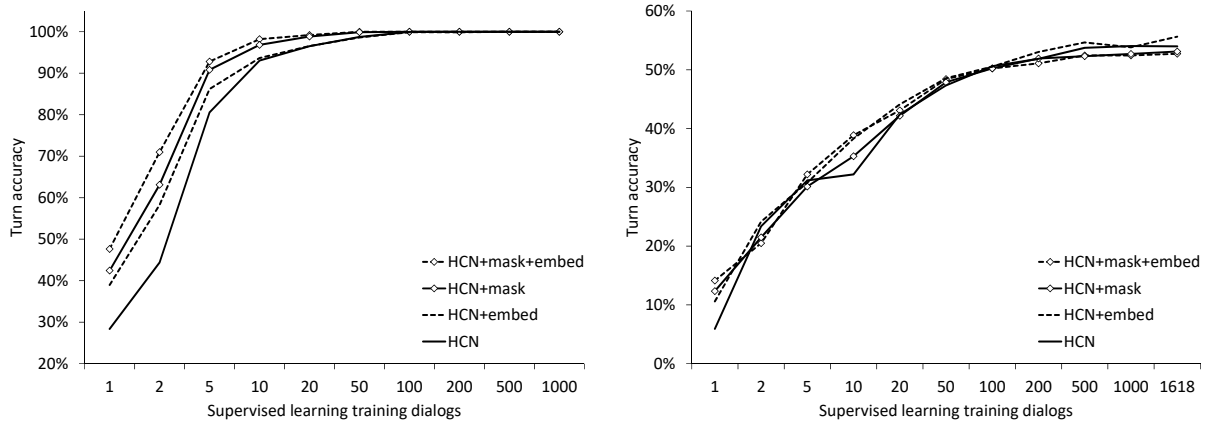
We next examined learning curves, training with increasing numbers of dialogs. To guard against bias in the ordering of the training set, we averaged over 5 runs, randomly permuting the order of the training dialogs in each run. Results are in Figure 2. In Task5, the action mask and utterance embeddings substantially reduce the number of training dialogs required (note the horizontal axis scale is logarithmic). For Task6, the bene-

[3] A handful of actions in Task6 seemed spurious; for these, we replaced them with a special "UNK" action in the training set, and masked this action at test time.

[4] Google News 100B model from `https://github.com/3Top/word2vec-api`

Model	Task5-OOV		Task6	
	Turn Acc.	Dialog Acc.	Turn Acc.	Dialog Acc.
Rules	**100%**	**100%**	33.3%	0.0%
Bordes and Weston (2016)	77.7%	0.0%	41.1%	0.0%
Liu and Perez (2016)	79.4%	0.0%	48.7%	1.4%
Eric and Manning (2017)	—	—	48.0%	1.5%
Seo et al. (2016)	96.0%	—	51.1%	—
HCN	**100%**	**100%**	54.0%	1.2%
HCN+embed	**100%**	**100%**	**55.6%**	1.3%
HCN+mask	**100%**	**100%**	53.1%	**1.9%**
HCN+embed+mask	**100%**	**100%**	52.7%	1.5%

Table 1: Results on bAbI dialog Task5-OOV and Task6 (Bordes and Weston, 2016). Results for "Rules" taken from Bordes and Weston (2016). Note that, unlike cited past work, HCNs make use of domain-specific procedural knowledge.



(a) bAbI dialog Task5-OOV.

(b) bAbI dialog Task6.

Figure 2: Training dialog count vs. turn accuracy for bAbI dialog Task5-OOV and Task6. "embed" indicates whether utterance embeddings were included; "mask" indicates whether the action masking code was active.

fits of the utterance embeddings are less clear. An error analysis showed that there are several systematic differences between the training and testing sets. Indeed, DSTC2 intentionally used different dialog policies for the training and test sets, whereas our goal is to mimic the policy in the training set.

Nonetheless, these tasks are the best public benchmark we are aware of, and HCNs exceed performance of existing sequence-to-sequence models. In addition, they match performance of past models using an order of magnitude less data (200 vs. 1618 dialogs), which is crucial in practical settings where collecting realistic dialogs for a new domain can be expensive.

5 Supervised learning evaluation II

We now turn to comparing with purely hand-crafted approaches. To do this, we obtained logs from our company's text-based customer support dialog system, which uses a sophisticated rule-based dialog manager. Data from this system is attractive for evaluation because it is used by real customers – not usability subjects – and because its rule-based dialog manager was developed by customer support professionals at our company, and not the authors. This data is not publicly available, but we are unaware of suitable human-computer dialog data in the public domain which uses rules.

Customers start using the dialog system by entering a brief description of their problem, such

as "I need to update my operating system". They are then routed to one of several hundred domains, where each domain attempts to resolve a particular problem. In this study, we collected human-computer transcripts for the high-traffic domains "reset password" and "cannot access account".

We labeled the dialog data as follows. First, we enumerated unique system actions observed in the data. Then, for each dialog, starting from the beginning, we examined each system action, and determined whether it was "correct". Here, correct means that it was the most appropriate action among the set of existing system actions, given the history of that dialog. If multiple actions were arguably appropriate, we broke ties in favor of the existing rule-based dialog manager. Example dialogs are provided in the Appendix Sections A.5 and A.6.

If a system action was labeled as correct, we left it as-is and continued to the next system action. If the system action was not correct, we replaced it with the correct system action, and discarded the rest of the dialog, since we do not know how the user would have replied to this new system action. The resulting dataset contained a mixture of complete and partial dialogs, containing only correct system actions. We partitioned this set into training and test dialogs. Basic statistics of the data are shown in Table 2.

In this domain, no entities were relevant to the control flow, and there was no obvious mask logic since any question could follow any question. Therefore, we wrote no domain-specific software for this instance of the HCN, and relied purely on the recurrent neural network to drive the conversation. The architecture and training of the RNN was the same as in Section 4, except that here we did not have enough data for a validation set, so we instead trained until we either achieved 100% accuracy on the training set or reached 200 epochs.

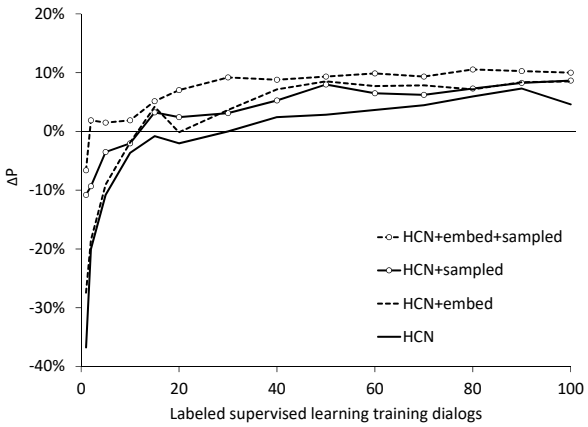
To evaluate, we observe that conventional measures like average dialog accuracy unfairly penalize the system used to collect the dialogs – in our case, the rule-based system. If the system used for collection makes an error at turn t, the labeled dialog only includes the sub-dialog up to turn t, and the system being evaluated off-line is only evaluated on that sub-dialog. In other words, in our case, reporting dialog accuracy would favor the HCN because it would be evaluated on fewer turns than the rule-based system. We therefore use a comparative measure that examines which method produces longer continuous sequences of correct system actions, starting from the beginning of the dialog. Specifically, we report $\Delta P = \frac{C(\text{HCN-win}) - C(\text{rule-win})}{C(\text{all})}$, where $C(\text{HCN-win})$ is the number of test dialogs where the rule-based approach output a wrong action before the HCN; $C(\text{rule-win})$ is the number of test dialogs where the HCN output a wrong action before the rule-based approach; and $C(\text{all})$ is the number of dialogs in the test set. When $\Delta P > 0$, there are more dialogs in which HCNs produce longer continuous sequences of correct actions starting from the beginning of the dialog. We run all experiments 5 times, each time shuffling the order of the training set. Results are in Figure 3. HCNs exceed performance of the existing rule-based system after about 30 dialogs.

	Forgot password	Account Access
Av. sys. turns/dialog	2.2	2.2
Max. sys. turns/dialog	5	9
Av. words/user turn	7.7	5.4
Unique sys. actions	7	16
Train dialogs	422	56
Test dialogs	148	60
Test acc. (rules)	64.9%	42.1%

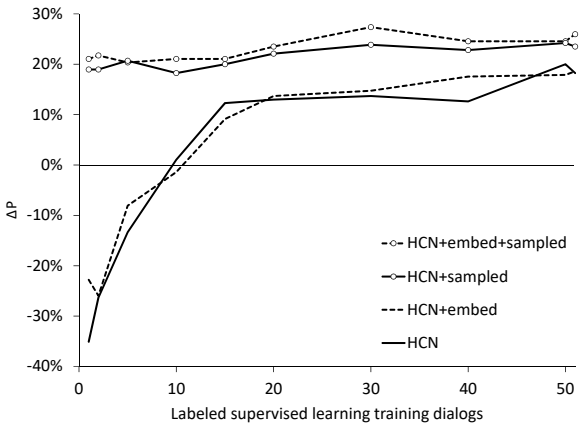
Table 2: Basic statistics of labeled customer support dialogs. Test accuracy refers to whole-dialog accuracy of the existing rule-based system.

In these domains, we have a further source of knowledge: the rule-based dialog managers themselves can be used to generate example "sunny-day" dialogs, where the user provides purely expected inputs. From each rule-based controller, synthetic dialogs were sampled to cover each expected user response at least once, and added to the set of labeled real dialogs. This resulted in 75 dialogs for the "Forgot password" domain, and 325 for the "Can't access account" domain. Training was repeated as described above. Results are also included in Figure 3, with the suffix "sampled". In the "Can't access account" domain, the sampled dialogs yield a large improvement, probably because the flow chart for this domain is large, so the sampled dialogs increase coverage. The gain in the "forgot password" domain is present but smaller.

In summary, HCNs can out-perform



(a) "Forgot password" domain.



(b) "Can't access account" domain.

Figure 3: Training dialogs vs. ΔP, where ΔP is the fraction of test dialogs where HCNs produced longer initial correct sequences of system actions than the rules, minus the fraction where rules produced longer initial correct sequences than the HCNs. "embed" indicates whether utterance embeddings were included; "sampled" indicates whether dialogs sampled from the rule-based controller were included in the training set.

production-grade rule-based systems with a reasonable number of labeled dialogs, and adding synthetic "sunny-day" dialogs improves performance further. Moreover, unlike existing pipelined approaches to dialog management that rely on an explicit state tracker, this HCN used no explicit state tracker, highlighting an advantage of the model.

6 Reinforcement learning illustration

In the previous sections, supervised learning (SL) was applied to train the LSTM to mimic dialogs provided by the system developer. Once a system operates at scale, interacting with a large number of users, it is desirable for the system to continue to learn *autonomously* using reinforcement learning (RL). With RL, each turn receives a measurement of goodness called a *reward*; the agent explores different sequences of actions in different situations, and makes adjustments so as to maximize the expected discounted sum of rewards, which is called the *return*, denoted G.

For optimization, we selected a *policy gradient* approach (Williams, 1992), which has been successfully applied to dialog systems (Jurčíček et al., 2011), robotics (Kohl and Stone, 2004), and the board game Go (Silver et al., 2016). In policy gradient-based RL, a model π is parameterized by $\mathbf{w}$ and outputs a distribution from which actions are sampled at each timestep. At the end of a trajectory – in our case, dialog – the return G for that trajectory is computed, and the gradients of the probabilities of the actions taken with respect to the model weights are computed. The weights are then adjusted by taking a gradient step proportional to the return:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \Big(\sum_t \nabla_{\mathbf{w}} \log \pi(a_t|\mathbf{h_t}; \mathbf{w})\Big)(G - b) \quad (1)$$

where α is a learning rate; a_t is the action taken at timestep t; $\mathbf{h_t}$ is the dialog history at time t; G is the return of the dialog; $\nabla_{\mathbf{x}} F$ denotes the Jacobian of F with respect to $\mathbf{x}$; b is a baseline described below; and $\pi(a|\mathbf{h}; \mathbf{w})$ is the LSTM – i.e., a stochastic policy which outputs a distribution over a given a dialog history $\mathbf{h}$, parameterized by weights $\mathbf{w}$. The baseline b is an estimate of the average return of the current policy, estimated on the last 100 dialogs using weighted importance sampling.[5] Intuitively, "better" dialogs receive a positive gradient step, making the actions selected more likely; and "worse" dialogs receive a negative gradient step, making the actions selected less likely.

SL and RL correspond to different methods of updating weights, so both can be applied to the same network. However, there is no guarantee that the optimal RL policy will agree with the SL training set; therefore, after each RL gradient step, we

[5]The choice of baseline does not affect the long-term convergence of the algorithm (i.e., the bias), but can dramatically affect the speed of convergence (i.e., the variance) (Williams, 1992).

check whether the updated policy reconstructs the training set. If not, we re-run SL gradient steps on the training set until the model reproduces the training set. Note that this approach allows new training dialogs to be added at any time during RL optimization.

We illustrate RL optimization on a simulated dialog task in the name dialing domain. In this system, a contact's name may have synonyms ("Michael" may also be called "Mike"), and a contact may have more than one phone number, such as "work" or "mobile", which may in turn have synonyms like "cell" for "mobile". This domain has a database of names and phone numbers taken from the Microsoft personnel directory, 5 entity types – `firstname`, `nickname`, `lastname`, `phonenumber`, and `phonetype` – and 14 actions, including 2 API call actions. Simple entity logic was coded, which retains the most recent copy of recognized entities. A simple action mask suppresses impossible actions, such as placing a phonecall before a phone number has been retrieved from the database. Example dialogs are provided in Appendix Section A.7.

To perform optimization, we created a simulated user. At the start of a dialog, the simulated user randomly selected a name and phone type, including names and phone types not covered by the dialog system. When speaking, the simulated user can use the canonical name or a nickname; usually answers questions but can ignore the system; can provide additional information not requested; and can give up. The simulated user was parameterized by around 10 probabilities, set by hand.

We defined the reward as being 1 for successfully completing the task, and 0 otherwise. A discount of 0.95 was used to incentivize the system to complete dialogs faster rather than slower, yielding return 0 for failed dialogs, and $G = 0.95^{T-1}$ for successful dialogs, where T is the number of system turns in the dialog. Finally, we created a set of 21 labeled dialogs, which will be used for supervised learning.

For the RNN in the HCN, we again used an LSTM with AdaDelta, this time with 32 hidden units. RL policy updates are made after each dialog. Since a simulated user was employed, we did not have real user utterances, and instead relied on context features, omitting bag-of-words and utterance embedding features.

We first evaluate RL by randomly initializing an LSTM, and begin RL optimization. After 10 RL updates, we freeze the policy, and run 500 dialogs with the user simulation to measure task completion. We repeat all of this for 100 runs, and report average performance. In addition, we also report results by initializing the LSTM using supervised learning on the training set, consisting of 1, 2, 5, or 10 dialogs sampled randomly from the training set, then running RL as described above.

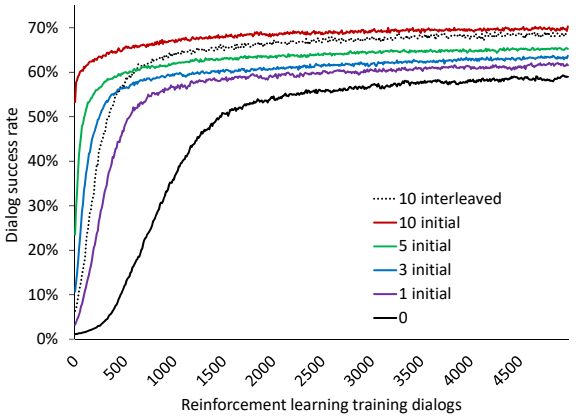


Figure 4: Dialog success rate vs. reinforcement learning training dialogs. Curve marked "0" begins with a randomly initialized LSTM. Curves marked "N initial" are pre-trained with N labeled dialogs. Curve marked "10, interleaved" adds one SL training dialog before RL dialog 0, 100, 200, ... 900.

Results are in Figure 4. Although RL alone can find a good policy, pre-training with just a handful of labeled dialogs improves learning speed dramatically. Additional experiments, not shown for space, found that ablating the action mask slowed training, agreeing with Williams (2008).

Finally, we conduct a further experiment where we sample 10 training dialogs, then add one to the training set just before RL dialog 0, 100, 200, ... , 900. Results are shown in Figure 4. This shows that SL dialogs can be introduced as RL is in progress – i.e., that it is possible to interleave RL and SL. This is an attractive property for practical systems: if a dialog error is spotted by a developer while RL is in progress, it is natural to add a training dialog to the training set.

7 Conclusion

This paper has introduced Hybrid Code Networks for end-to-end learning of task-oriented dialog

systems. HCNs support a separation of concerns where procedural knowledge and constraints can be expressed in software, and the control flow is learned. Compared to existing end-to-end approaches, HCNs afford more developer control and require less training data, at the expense of a small amount of developer effort.

Results in this paper have explored three different dialog domains. On a public benchmark in the restaurants domain, HCNs exceeded performance of purely learned models. Results in two troubleshooting domains exceeded performance of a commercially deployed rule-based system. Finally, in a name-dialing domain, results from dialog simulation show that HCNs can also be optimized with a mixture of reinforcement and supervised learning.

In future work, we plan to extend HCNs by incorporating lines of existing work, such as integrating the entity extraction step into the neural network (Dhingra et al., 2017), adding richer utterance embeddings (Socher et al., 2013), and supporting text generation (Sordoni et al., 2015). We will also explore using HCNs with automatic speech recognition (ASR) input, for example by forming features from n-grams of the ASR n-best results (Henderson et al., 2014b). Of course, we also plan to deploy the model in a live dialog system. More broadly, HCNs are a general model for stateful control, and we would be interested to explore applications beyond dialog systems – for example, in NLP medical settings or human-robot NL interaction tasks, providing domain constraints are important for safety; and in resource-poor settings, providing domain knowledge can amplify limited data.

References

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR* abs/1605.07683. http://arxiv.org/abs/1605.07683.

Franois Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proc NIPS 2014 Deep Learning and Representation Learning Workshop*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proc Association for Computational Linguistics, Vancouver, Canada*.

Mihail Eric and Christopher D Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *CoRR* abs/1701.04024. https://arxiv.org/abs/1701.04024.

David Griol, Llus F. Hurtado, Encarna Segarra, and Emilio Sanchis. 2008. A statistical approach to spoken dialog systems design and evaluation. *Speech Communication* 50(8–9).

Matthew Henderson, Blaise Thomson, and Jason Williams. 2014a. The second dialog state tracking challenge. In *Proc SIGdial Workshop on Discourse and Dialogue, Philadelphia, USA*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Word-based Dialog State Tracking with Recurrent Neural Networks. In *Proc SIGdial Workshop on Discourse and Dialogue, Philadelphia, USA*.

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

Chiori Hori, Kiyonori Ohtake, Teruhisa Misu, Hideki Kashioka, and Satoshi Nakamura. 2009. Statistical dialog management applied to WFST-based dialog systems. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. pages 4793–4796. https://doi.org/10.1109/ICASSP.2009.4960703.

Filip Jurčíček, Blaise Thomson, and Steve Young. 2011. Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as pomdps. *ACM Transactions on Speech and Language Processing (TSLP)* 7(3):6.

Nate Kohl and Peter Stone. 2004. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. IEEE, volume 3, pages 2619–2624.

Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. 2009. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication* 51(5):466–484.

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

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proc HLT-NAACL, San Diego, California, USA*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proc Association for Computational Linguistics, Berlin, Germany*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016c. Deep reinforcement learning for dialogue generation. In *Proc Conference on Empirical Methods in Natural Language Processing, Austin, Texas, USA*.

Lihong Li, He He, and Jason D. Williams. 2014. Temporal supervised learning for inferring a dialog policy from example conversations. In *Proc IEEE Workshop on Spoken Language Technologies (SLT), South Lake Tahoe, Nevada, USA*.

Fei Liu and Julien Perez. 2016. Gated end-to-end memory networks. *CoRR* abs/1610.04211. http://arxiv.org/abs/1610.04211.

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue and Discourse* 8(1).

Yi Luan, Yangfeng Ji, and Mari Ostendorf. 2016. LSTM based conversation models. *CoRR* abs/1603.09457. http://arxiv.org/abs/1603.09457.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. Coherent dialogue with attention-based language models. *CoRR* abs/1611.06997. http://arxiv.org/abs/1611.06997.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc Advances in Neural Information Processing Systems, Lake Tahoe, USA*. pages 3111–3119.

Min Joon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. Query-regression networks for machine comprehension. *CoRR* abs/1606.04582. http://arxiv.org/abs/1606.04582.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'16, pages 3776–3783. http://dl.acm.org/citation.cfm?id=3016387.3016435.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues.

Lifeng Shang, Zhengdong Lu, , and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proc Association for Computational Linguistics, Beijing, China*.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Satinder Singh, Diane J Litman, Michael Kearns, and Marilyn A Walker. 2002. Optimizing dialogue management with reinforcement leaning: experiments with the NJFun system. *Journal of Artificial Intelligence* 16:105–133.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Chris Manning, Andrew Ng, and Chris Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, USA*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proc HLT-NAACL, Denver, Colorado, USA*.

Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. In *arXiv preprint: 1606.02689*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proc Advances in Neural Information Processing Systems (NIPS), Montreal, Canada*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proc ICML Deep Learning Workshop*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *CoRR* abs/1604.04562. http://arxiv.org/abs/1604.04562.

Jason D. Williams. 2008. The best of both worlds: Unifying conventional dialog systems and POMDPs. In *Proc Intl Conf on Spoken Language Processing (ICSLP), Brisbane, Australia*.

Jason D. Williams and Steve Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language* 21(2):393–422.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2016. Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling. *CoRR* abs/1605.05110. http://arxiv.org/abs/1605.05110.

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. In *Proc NIPS workshop on Machine Learning for Spoken Language Understanding and Interaction*.

Steve Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. POMDP-based Statistical Spoken Dialogue Systems: a Review. *Proceedings of the IEEE* PP(99):1–20.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

A Supplemental Material

A.1 Model implementation details

The RNN was specified using Keras version 0.3.3, with back-end computation in Theano version 0.8.0.dev0 (Theano Development Team, 2016; Chollet, 2015). The Keras model specification is given below. The input variable `obs` includes all features from Figure 1 step 6 except for the previous action (step 18) and the action mask (step 6, top-most vector).

```
# Given:
# obs_size, action_size, nb_hidden

g = Graph()
g.add_input(
  name='obs',
  input_shape=(None, obs_size)
)
g.add_input(
  name='prev_action',
  input_shape=(None, action_size)
)
g.add_input(
  name='avail_actions',
  input_shape=(None, action_size)
)
g.add_node(
  LSTM(
    n_hidden,
    return_sequences=True,
    activation='tanh',
  ),
  name='h1',
  inputs=[
    'obs',
    'prev_action',
    'avail_actions'
  ]
)
g.add_node(
  TimeDistributedDense(
    action_size,
    activation='softmax',
  ),
  name='h2',
  input='h1'
)
g.add_node(
  Activation(
    activation=normalize,
  ),
  name='action',
  inputs=['h2','avail_actions'],
  merge_mode='mul',
  create_output=True
)
g.compile(
  optimizer=Adadelta(clipnorm=1.),
  sample_weight_modes={
    'action': 'temporal'
  },
  loss={
    'action':'categorical_crossentropy'
  }
)
```

Model sizes are given in Table 3. Example dialogs are given below for each of the 5 dialog systems. For space and readability, the entity tags that appear in the user and system sides of the dialogs have been removed – for example, *Call* `<name>`*Joan*`</name>` is shown as *Call Joan*.

A.2 bAbI Task5 example dialog

good morning
hello what can i help you with today
i'd like to book a table with italian food
i'm on it
[silence]
where should it be
in paris
how many people would be in your party
for six people please
which price range are looking for
in a cheap price range please
ok let me look into some options for you
[silence]
`api_call italian paris six cheap`
`[database results returned]`
[silence]
what do you think of this option: resto_madrid_cheap_spanish_8stars
no this does not work for me
sure let me find an other option for you
[silence]
what do you think of this option: resto_madrid_cheap_spanish_6stars
do you have something else
sure let me find an other option for you
[silence]
what do you think of this option: resto_madrid_cheap_spanish_1stars
it's perfect
great let me do the reservation
may i have the phone number of the restaurant
here it is resto_madrid_cheap_spanish_1stars_phone

	bAbI Task5	bAbI Task6	Reset password	Account access	Dialer
Utterance embed.	300	300	300	300	0
Bag of words	85	523	1124	278	0
Context features	4	14	0	0	17
`obs_size`	389	837	1424	578	17
`action_size`	16	58	7	16	14
`nb_hidden`	128	128	128	128	32

Table 3: Dimensions of the 5 HCNs in this paper.

thanks
is there anything i can help you with
no thank you
you're welcome

A.3 bAbI Task6 example dialog

In the system turn marked [*], the system provided venue information without first making an API call.

Hello , welcome to the Cambridge restaurant system . You can ask for restaurants by area , price range or food type . How may I help you ?
im looking for canapes food in any price range
I am sorry but there is no canapes restaurant that matches your request [*]
how about european
`api_call european R_location R_price`
`[database results returned]`
[silence]
michaelhouse_cafe is a great restaurant serving european food and it is in the expensive price range
food an
What part of town do you have in mind?
any any part of town
michaelhouse_cafe serves european food .
may i have the address and phone number please
The phone number of michaelhouse_cafe is michaelhouse_cafe_phone
thank you good bye
you are welcome

A.4 Task5 and Task6 context features

Context features used in Section 4 are in Table 4.

A.5 Forgot password example dialog

Some system actions have been shortened for space.

How do I recover my password
Okay, you don't need to remember your password, we can reset it. Would you like to try that?
Yes
To reset your password, please see this URL ... Was that helpful?
I don't have a flashdrive
Let's connect you to a person who can help you

A.6 Account access example dialog

Some system actions have been shortened for space.

cannot sign in to my email
Please check caps lock is off, and that your email is spelled correctly. Are you able to access your account now?
no
If you still can't sign in, please go the the following page and check that the email server is running ok... Is everything up and running?
yes
You need to reset your password - here's how ... Did this solve your problem?
still cannot get in
Did you receive the security code?
yes
Check you entered the right numbers. Did this solve your problem?
no
Have you seen a message that your profile is temporarily blocked?
just your message of update on sept 15
Did you turn on two-step verification recently?
whats that?
Sorry, I can't help. Rephrase your question, or click below to talk to a person.

A.7 Name dialing example dialogs

How can I help you?
Call Joe
There's more than one person named Joe. Can you say their full name?
Call Joe Adamson
`SavePhonetypeavail()`
Calling Joe Adamson, work
`PlaceCall()`

How can I help you?
Call Sally on her home

Feature	**bAbI Task5**	**bAbI Task6**
Presence of each entity in dialog state	4	3
Presence of each entity in current utterance	–	3
Whether DB has been queried yet	–	1
Whether DB results are empty	–	1
Whether DB results are non-empty	–	1
Whether any DB results have been presented	–	1
Whether all DB results have been presented	–	1
Whether any DB results are available to present	–	1
Whether curent query yields no results in training set	–	1
Whether cuisine in current query yields no results in training set	–	1
Total context features	4	14

Table 4: Binary context features used to convey entity and database state in Section 4.

Sorry, I don't have a home number for Sally Smith. I only have a work phone. Do you want to call that number?
no
Oh, sorry about that. Goodbye.

Generating Contrastive Referring Expressions

Martín Villalba and **Christoph Teichmann** and **Alexander Koller**
Department of Language Science and Technology
Saarland University, Germany
{villalba|cteichmann|koller}@coli.uni-saarland.de

Abstract

The referring expressions (REs) produced by a natural language generation (NLG) system can be misunderstood by the hearer, even when they are semantically correct. In an interactive setting, the NLG system can try to recognize such misunderstandings and correct them. We present an algorithm for generating corrective REs that use contrastive focus ("no, the BLUE button") to emphasize the information the hearer most likely misunderstood. We show empirically that these contrastive REs are preferred over REs without contrast marking.

1 Introduction

Interactive natural language generation (NLG) systems face the task of detecting when they have been misunderstood, and reacting appropriately to fix the problem. For instance, even when the system generated a semantically correct referring expression (RE), the user may still misunderstand it, i.e. resolve it to a different object from the one the system intended. In an interactive setting, such as a dialogue system or a pedestrian navigation system, the system can try to detect such misunderstandings – e.g. by predicting what the hearer understood from their behavior (Engonopoulos et al., 2013) – and to produce further utterances which resolve the misunderstanding and get the hearer to identify the intended object after all.

When humans correct their own REs, they routinely employ *contrastive focus* (Rooth, 1992; Krifka, 2008) to clarify the relationship to the original RE. Say that we originally described an object b as "the blue button", but the hearer approaches a button b' which is green, thus providing evidence that they misunderstood the RE to mean b'. In this case, we would like to say "no, the BLUE button", with the contrastive focus realized by an appropriate pitch accent on "BLUE". This utterance alerts the hearer to the fact that they misunderstood the original RE; it reiterates the information from the original RE; and it marks the attribute "blue" as a salient difference between b' and the object the original RE was intended to describe.

In this paper, we describe an algorithm for generating REs with contrastive focus. We start from the modeling assumption that misunderstandings arise because the RE r_s the system uttered was corrupted by a noisy channel into an RE r_u which the user "heard" and then resolved correctly; in the example above, we assume the user literally heard "the green button". We compute this (hypothetical) RE r_u as the RE which refers to b' and has the lowest edit distance from r_s. Based on this, we mark the contrastive words in r_s, i.e. we transform "the blue button" into "the BLUE button". We evaluate our system empirically on REs from the GIVE Challenge (Koller et al., 2010) and the TUNA Challenge (van der Sluis et al., 2007), and show that the contrastive REs generated by our system are preferred over a number of baselines.

The paper is structured as follows. We first review related work in Section 2 and define the problem of generating contrastive REs in Section 3. Section 4 sketches the general architecture for RE generation on which our system is based. In Section 5, we present the corruption model and show how to use it to reconstruct r_u. Section 6 describes how we use this information to generate contrastive markup in r_s, and in Section 7 we evaluate our approach.

2 Related Work

The notion of focus has been extensively studied in the literature on theoretical semantics and prag-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 678–687
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1063

matics, see e.g. Krifka (2008) and Rooth (1997) for overview papers. Krifka follows Rooth (1992) in taking focus as "indicat(ing) the presence of *alternatives* that are relevant for the interpretation of linguistic expressions"; focus then establishes a contrast between an object and these alternatives. Bornkessel and Schlesewsky (2006) find that corrective focus can even override syntactic requirements, on the basis of "its extraordinarily high communicative saliency". This literature is purely theoretical; we offer an algorithm for automatically generating contrastive focus.

In speech, focus is typically marked through intonation and pitch accents (Levelt, 1993; Pierrehumbert and Hirschberg, 1990; Steube, 2001), while concepts that can be taken for granted are deaccented and/or deleted. Developing systems which realize precise pitch contours for focus in text-to-speech settings is an ongoing research effort. We therefore realize focus in written language in this paper, by capitalizing the focused word. We also experiment with deletion of background words.

There is substantial previous work on interactive systems that detect and respond to misunderstandings. Misu et al. (2014) present an error analysis of an in-car dialogue system which shows that more than half the errors can only be resolved through further clarification dialogues, as opposed to better sensors and/or databases; that is, by improved handling of misunderstandings. Engonopoulos et al. (2013) detect misunderstandings of REs in interactive NLG through the use of a statistical model. Their model also predicts the object to which a misunderstood RE was incorrectly resolved. Moving from misunderstanding detection to error correction, Zarrieß and Schlangen (2016) present an interactive NLG algorithm which is capable of referring in installments, in that it can generate multiple REs that are designed to correct misunderstandings of earlier REs to the same object. The interactive NLG system developed by Akkersdijk et al. (2011) generates both reflective and anticipative feedback based on what a user does and sees. Their error detection and correction strategy distinguishes a fixed set of possible situations where feedback is necessary, and defines custom, hard-coded RE generation sub-strategies for each one. None of these systems generate REs marked for focus.

We are aware of two items of previous work that address the generation of contrastive REs directly. Milosavljevic and Dale (1996) outline strategies for generating clarificatory comparisons in encyclopedic descriptions. Their surface realizer can generate contrastive REs, but the attributes that receive contrastive focus have to be specified by hand. Krahmer and Theune (2002) extend the Incremental Algorithm (Dale and Reiter, 1995) so it can mark attributes as contrastive. This is a fully automatic algorithm for contrastive REs, but it inherits all the limitations of the Incremental Algorithm, such as its reliance on a fixed attribute order. Neither of these two approaches evaluates the quality of the contrastive REs it generates.

Finally, some work has addressed the issue of generating texts that realize the discourse relation *contrast*. For instance, Howcroft et al. (2013) show how to choose contrastive discourse connectives (*but, while,* ...) when generating restaurant descriptions, thus increasing human ratings for naturalness. Unlike their work, the research presented in this paper is not about discourse relations, but about assigning focus in contrastive REs.

3 Interactive NLG

We start by introducing the problem of generating corrective REs in an interactive NLG setting. We use examples from the GIVE Challenge (Koller et al., 2010) throughout the paper; however, the algorithm itself is domain-independent.

GIVE is a shared task in which an NLG system (the instruction giver, IG) must guide a human user (the instruction follower, IF) through a virtual 3D environment. The IF needs to open a safe and steal a trophy by clicking on a number of buttons in the right order without triggering alarms. The job of the NLG system is to generate natural-language instructions which guide the IF to complete this task successfully.

The generation of REs has a central place in the GIVE Challenge because the system frequently needs to identify buttons in the virtual environment to the IF. Figure 1 shows a screenshot of a GIVE game in progress; here b_1 and b_4 are blue buttons, b_2 and b_3 are yellow buttons, and w_1 is a window. If the next button the IF needs to press is b_4 – the *intended object*, o_s – then one good RE for b_4 would be "the blue button below the window", and the system should utter:

(1) Press the blue button below the window.

After uttering this sentence, the system can

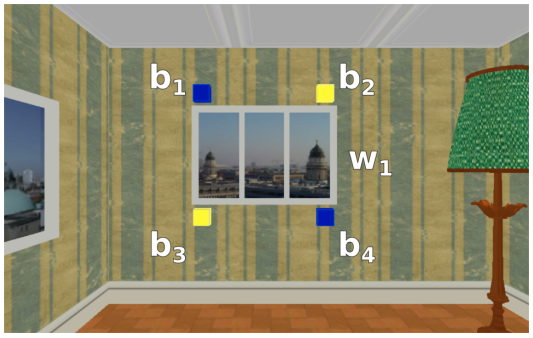


Figure 1: Example scene from the GIVE Challenge.

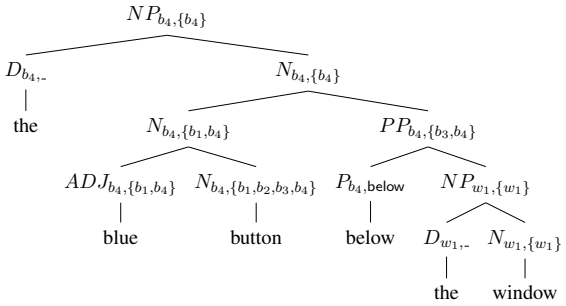


Figure 2: Example syntax tree for an RE for b_4.

track the IF's behavior to see whether the IF has understood the RE correctly. If the wrong button is pressed, or if a model of IF's behavior suggests that they are about to press the wrong button (Engonopoulos et al., 2013), the original RE has been misunderstood. However, the system still gets a second chance, since it can utter a *corrective* RE, with the goal of identifying b_4 to the IF after all. Examples include simply repeating the original RE, or generating a completely new RE from scratch. The system can also explicitly take into account which part of the original RE the IF misunderstood. If it has reason to believe that the IF resolved the RE to b_3, it could say:

(2) No, the BLUE button below the window.

This use of contrastive focus distinguishes the attributes the IF misunderstood (blue) from those that they understood correctly (below the window), and thus makes it easier for the IF to resolve the misunderstanding. In speech, contrastive focus would be realized with a pitch accent; we approximate this accent in written language by capitalizing the focused word. We call an RE that uses contrastive focus to highlight the difference between the misunderstood and the intended object, a *contrastive RE*. The aim of this paper is to present an algorithm for computing contrastive REs.

4 Generating Referring Expressions

While we make no assumptions on how the original RE r_s was generated, our algorithm for reconstructing the corrupted RE r_u requires an RE generation algorithm that can represent all semantically correct REs for a given object compactly in a chart. Here we sketch the RE generation of Engonopoulos and Koller (2014), which satisfies this requirement.

This algorithm assumes a synchronous grammar which relates strings with the sets of objects they refer to. Strings and their referent sets are constructed in parallel from lexicon entries and grammar rules; each grammar rule specifies how the referent set of the parent is determined from those of the children. For the scene in Figure 1, we assume lexicon entries which express, among other things, that the word "blue" denotes the set $\{b_1, b_4\}$ and the word "below" denotes the relation $\{(w_1, b_1), (w_1, b_2), (b_3, w_1), (b_4, w_1)\}$. We combine these lexicons entries using rules such as

$$\text{"N} \rightarrow \text{button() } | \text{button } | \{b_1, b_2, b_3, b_4\}\text{"}$$

which generates the string "button" and associates it with the set of all buttons or

$$\text{"N} \rightarrow \text{N1(N,PP) } | w_1 \bullet w_2 \; | R_1 \cap R_2\text{"}$$

which states that a phrase of type noun can be combined with a prepositional phrase and their denotations will be intersected. Using these rules we can determine that "the window" denotes $\{w_1\}$, that "below the window" can refer to $\{b_3, b_4\}$ and that "blue button below the window" uniquely refers to $\{b_4\}$. The syntax tree in Fig. 2 represents a complete derivation of an RE for $\{b_4\}$.

The algorithm of Engonopoulos and Koller computes a chart which represents the set of all possible REs for a given set of input objects, such as $\{b_4\}$, according to the grammar. This is done by building a chart containing all derivations of the grammar which correspond to the desired set. They represent this chart as a finite tree automaton (Comon et al., 2007). Here we simply write the chart as a Context-Free Grammar. The strings produced by this Context-Free Grammar are then exactly the REs for the intended object. For example, the syntax tree in Fig. 2 is generated by the parse chart for the set $\{b_4\}$. Its nonterminal symbols consist of three parts: a syntactic category

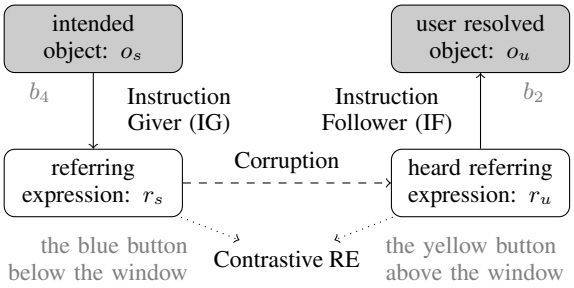


Figure 3: The corruption model.

(given by the synchronous grammar), the referent for which an RE is currently being constructed, and the set of objects to which the entire subtree refers. The grammar may include recursion and therefore allow for an infinite set of possible REs. If it is weighted, one can use the Viterbi algorithm to compute the best RE from the chart.

5 Listener Hypotheses and Edit Distance

5.1 Corruption model

Now let us say that the system has generated and uttered an RE r_s with the intention of referring to the object o_s, but it has then found that the IF has misunderstood the RE and resolved it to another object, o_u (see Fig. 3). We assume for the purposes of this paper that such a misunderstanding arises because r_s was corrupted by a noisy channel when it was transmitted to the IF, and the IF "heard" a different RE, r_u. We further assume that the IF then resolved r_u correctly, i.e. the corruption in the transmission is the only source of misunderstandings.

In reality, there are of course many other reasons why the IF might misunderstand r_s, such as lack of attention, discrepancies in the lexicon or the world model of the IG and IF, and so on. We make a simplifying assumption in order to make the misunderstanding explicit at the level of the RE strings, while still permitting meaningful corrections for a large class of misunderstandings.

An NLG system that builds upon this idea in order to generate a corrective RE has access to the values of o_s, r_s and o_u; but it needs to infer the most likely corrupted RE r_u. To do this, we model the corruption using the edit operations used for the familiar *Levenshtein edit distance* (Mohri, 2003) over the alphabet Σ: S_a, substitution of a word with a symbol $a \in \Sigma$; D, deletion of a word; I_a, insertion of the symbol $a \in \Sigma$; or K, keeping the word. The noisy channel passes over each word in r_s and applies either D, K or one of the S operations to it. It may also apply I operations before or after a word. We call any sequence s of edit operations that could apply to r_s an *edit sequence for* r_s.

An example for an edit sequence which corrupts r_s = "the blue button below the window" into r_u = "the yellow button above the window" is shown in Figure 4. The same r_u could also have been generated by the edit operation sequence $K\, S_{\text{yellow}}\, K\, S_{\text{above}}\, K\, K$, and there is generally a large number of edit sequences that could transform between any two REs. If an edit sequence s maps x to y, we write $\text{apply}(s, x) = y$.

We can now define a probability distribution $P(s \mid r_s)$ over edit sequences s that the noisy channel might apply to the string r_s, as follows:

$$P(s \mid r_s) = \frac{1}{Z} \prod_{s_i \in s} \exp(-c(s_i)),$$

where $c(s_i)$ is a cost for using the edit operation s_i. We set $c(K) = 0$, and for any a in our alphabet we set $c(S_a) = c(I_a) = c(D) = \mathcal{C}$, for some fixed $\mathcal{C} > 0$. Z is a normalizing constant which is independent of s and ensures that the probabilities sum to 1. It is finite for sufficiently high values of $\mathcal{C}$, because no sequence for r_s can ever contain more K, S and D operations than there are words in r_s, and the total weight of sequences generated by adding more and more I operations will converge.

Finally, let L be the set of referring expressions that the IF would resolve to o_u, i.e. the set of candidates for r_u. Then the most probable edit sequence for r_s which generates an $r_u \in L$ is given by

$$\begin{aligned} s^* &= \underset{s\,:\,\text{apply}(s,r_s)\in L}{\arg\max} P(s \mid r_s) \\ &= \arg\min_s \textstyle\sum_{s_i \in s} c(s_i), \end{aligned}$$

i.e. s^* is the edit sequence that maps r_s to an RE in L with minimal cost. We will assume that s^* is the edit sequence that corrupted r_s, i.e. that $r_u = \text{apply}(s^*, r_s)$.

5.2 Finding the most likely corruption

It remains to compute s^*; we will then show in Section 6 how it can be used to generate a corrective RE. Attempting to find s^* by enumeration is impractical, as the set of edit sequences for a given r_s and r_u may be large and the set of possible r_u for a given o_u may be infinite. Instead

r_s	the	blue		button	below	the	window
edit operation sequence	K	D	I_{yellow}	K	S_{above}	K	K
r_u	the		yellow	button	above	the	window

Figure 4: Example edit sequence for a given corruption.

we will use the algorithm from Section 4 to compute a chart for all the possible REs for o_u, represented as a context-free grammar G whose language $L = L(G)$ consists of these REs. We will then intersect it with a finite-state automaton which keeps track of the edit costs, obtaining a second context-free grammar G'. These operations can be performed efficiently, and s^* can be read off of the minimum-cost syntax tree of G'.

Edit automaton. The possible edit sequences for a given r_s can be represented compactly in the form of a weighted finite-state automaton $F(r_s)$ (Mohri, 2003). Each run of the automaton on a string w corresponds to a specific edit sequence that transforms r_s into w, and the sum of transition weights of the run is the cost of that edit sequence. We call $F(r_s)$ the *edit automaton*. It has a state q_i for every position i in r_s; the start state is q_0 and the final state is $q_{|r_s|}$. For each i, it has a "keep" transition from q_i to q_{i+1} that reads the word at position i with cost 0. In addition, there are transitions from q_i to q_{i+1} with cost $\mathcal{C}$ that read any symbol in Σ (for substitution) and ones that read the empty string ϵ (for deletion). Finally, there is a loop with cost $\mathcal{C}$ from each q_i to itself and for any symbol in Σ, implementing insertion.

An example automaton for r_s = "the blue button below the window" is shown in Figure 5. The transitions are written in the form $\langle$word in w : associated cost$\rangle$. Note that every path through the edit transducer corresponds to a specific edit sequence s, and the sum of the costs along the path corresponds to $-\log P(s \mid r_s) - \log Z$.

Combining G and $F(r_s)$. Now we can combine G with $F(r_s)$ to obtain G', by intersecting them using the Bar-Hillel construction (Bar-Hillel et al., 1961; Hopcroft and Ullman, 1979). For the purposes of our presentation we assume that G is in Chomsky Normal Form, i.e. all rules have the form $A \to a$, where a is a word, or $A \to B\ C$, where both symbols on the right hand side are nonterminals. The resulting grammar G' uses nonterminal symbols of the form $N_{b,A,\langle q_i,q_k\rangle}$, where b, A are as in Section 4, and q_i, q_k indicate that the string derived by this nonterminal was generated by editing the substring of r_s from position i to k.

Let $N_{b,A} \to a$ be a production rule of G with a word a on the right-hand side; as explained above, b is the object to which the subtree should refer, and A is the set of objects to which the subtree actually might refer. Let $t = q_i \to \langle a{:}c\rangle q_k$ be a transition in $F(r_s)$, where q, q' are states of $F(r_s)$ and c is the edit cost. From these two, we create a context-free rule $N_{b,A,\langle q_i,q_k\rangle} \to a$ with weight c and add it to G'. If $k = i + 1$, these rules represent K and S operations; if $k = i$, they represent insertions.

Now let $N_{b,A} \to X_{b_1,A_1}\, Y_{b_2,A_2}$ be a binary rule in G, and let q_i, q_j, q_k be states of $F(r_s)$ with $i \le j \le k$. We then add a rule $N_{b,A,\langle q_i,q_k\rangle} \to X_{b_1,A_1,\langle q_i,q_j\rangle}\, Y_{b_2,A_2,\langle q_j,q_k\rangle}$ to G'. These rules are assigned weight 0, as they only combine words according to the grammar structure of G and do not encode any edit operations.

Finally, we deal with deletion. Let $N_{b,A}$ be a nonterminal symbol in G and let q_h, q_i, q_j, q_k be states of $F(r_s)$ with $h \le i \le j \le k$. We then add a rule $N_{b,A,\langle q_h,q_k\rangle} \to N_{b,A,\langle q_i,q_j\rangle}$ to G'. This rule deletes the substrings from positions h to i and j to k from r_s; thus we assign it the cost $((i-h)+(k-j))\mathcal{C}$, i.e. the cost of the corresponding ϵ transitions.

If the start symbol of G is $S_{b,A}$, then the start symbol of G' is $S_{b,A,\langle q_0,q_{|r_s|}\rangle}$. This construction intersects the languages of G and $F(r_s)$, but because $F(r_s)$ accepts all strings over the alphabet, the languages of G' and G will be the same (namely, all REs for o_u). However, the weights in G' are inherited from $F(r_s)$; thus the weight of each RE in $L(G')$ is the edit cost from r_s.

Example. Fig. 6 shows an example tree for the G' we obtain from the automaton in Fig. 5. We can read the string w = "the yellow button above the window" off of the leaves; by construction, this is an RE for o_u. Furthermore, we can reconstruct the edit sequence that maps from r_s to w from the rules of G' that

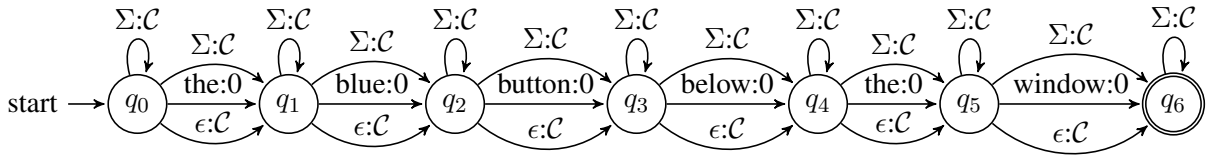


Figure 5: Edit automaton $F(r_s)$ for $r_s =$ "the blue button below the window".

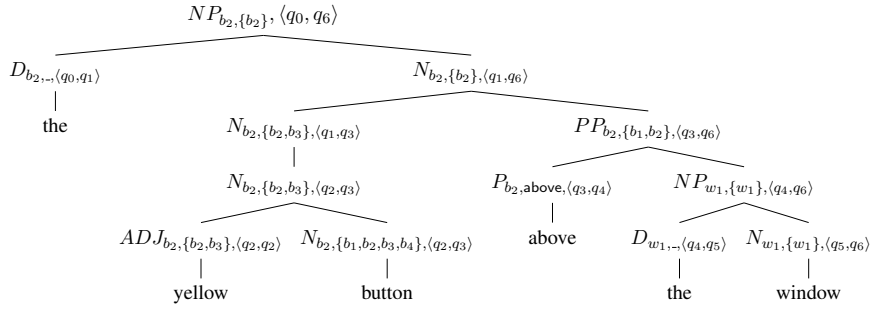
Tree	
s	$K\ D\ I_{\text{yellow}}\ K\ S_{\text{above}}\ K\ K$
Emphasis	No, press the BLUE button BELOW the window

Figure 6: A syntax tree described by G', together with its associated edit sequence and contrastive RE.

were used to derive w. We can see that "yellow" was created by an insertion because the two states of $F(r_s)$ in the preterminal symbol just above it are the same. If the two states are different, then the word was either substituted ("above", if the rule had weight $\mathcal{C}$) or kept ("the", if the rule had weight 0). By contrast, unary rules indicate deletions, in that they make "progress" in r_s without adding new words to w.

We can compute the minimal-cost tree of G' using the Viterbi algorithm. Thus, to summarize, we can calculate s^* from the intersection of a context-free grammar G representing the REs to o_u with the automaton $F(r_s)$ representing the edit distance to r_s. From this, we obtain $r_u = \text{apply}(s^*, r_s)$. This is efficient in practice.

6 Generating Contrastive REs

6.1 Contrastive focus

We are now ready to generate a contrastive RE from r_s and s^*. We assign focus to the words in r_s which were changed by the corruption – that is, the ones to which s^* applied Substitute or Delete operations. For instance, the edit sequence in Fig. 6 deleted "blue" and substituted "below" with "above". Thus, we mark these words with focus, and obtain the contrastive RE "the BLUE button BELOW the window". We call this strategy *Emphasis*, and write ${r_s}^E$ for the RE obtained by applying the Emphasis strategy to the RE r_s.

6.2 Shortening

We also investigate a second strategy, which generates more succinct contrastive REs than the *Emphasis* strategy. Most research on RE generation (e.g. Dale and Reiter (1995)) has assumed that hearers should prefer succinct REs, which in particular do not violate the Maxim of Quantity (Grice, 1975). When we utter a contrastive RE, the user has previously heard the RE r_s, so some of the information in ${r_s}^E$ is redundant. Thus we might obtain a more succinct, and possibly better, RE by dropping such redundant information from the RE.

For the grammars we consider here, ${r_s}^E$ often combines an NP and a PP, e.g. "[blue button]$_{NP}$ [below the window]$_{PP}$". If errors occur only in one of these constituents, then it might be sufficient to generate a contrastive RE using only that constituent. We call this strategy *Shortening* and define it as follows.

If all the words that are emphasized in ${r_s}^E$ are in the NP, the Shortening RE is "the" plus the NP, with emphasis as in ${r_s}^E$. So if r_s is "the [blue button] [above the window]" and $s^* = K\ S_{yellow}\ K\ K\ K\ K$, corresponding to a ${r_s}^E$ of "the [BLUE button] [above the window]", then the RE would be "the [BLUE button]".

If all the emphasis in ${r_s}^E$ is in the PP, we use

We wanted our player to select this button:

So we told them: *press the red button to the right of the blue button*.
But they selected this button instead:

Which correction is better for this scene?

- ○ No, press the red **BUTTON** to the right of the **BLUE BUTTON**
- ○ No, press the red button to the **RIGHT** of the blue button

Figure 7: A sample scene from Experiment 1.

We wanted our player to select the person circled in green:

So we told them: *the light haired old man in a suit looking straight*.

But they selected the person circled in red instead.
Which correction is better for this scene?

- ○ No, the light haired old man **IN A SUIT LOOKING STRAIGHT**
- ○ No, the **LIGHT HAIRED OLD** man in a suit looking straight

Figure 8: A sample scene from Experiment 2.

"the one" plus the PP and again capitalize as in ${r_s}^E$. So if we have $s^* = K\,K\,K\,S_{below}\,K\,K$, where ${r_s}^E$ is "the [blue button] [ABOVE the window]", we obtain "the one [ABOVE the window]." If there is no PP or if ${r_s}^E$ emphasizes words in both the NP and the PP, then we just use ${r_s}^E$.

7 Evaluation

To test whether our algorithm for contrastive REs assigns contrastive focus correctly, we evaluated it against several baselines in crowdsourced pairwise comparison overhearer experiments. Like Buß et al. (2010), we opted for an overhearer experiment to focus our evaluation on the effects of contrastive feedback, as opposed to the challenges presented by the navigational and timing aspects of a fully interactive system.

7.1 Domains and stimuli

We created the stimuli for our experiments from two different domains. We performed a first experiment with scenes from the GIVE Challenge, while a second experiment replaced these scenes with stimuli from the "People" domain of the TUNA Reference Corpus (van der Sluis et al., 2007). This corpus consists of photographs of men annotated with nine attributes, such as whether the person has a beard, a tie, or is looking straight. Six of these attributes were included in the corpus to better reflect human RE generation strategies. Many human-generated REs in the corpus are overspecific, in that they contain attributes that are not necessary to make the RE semantically unique.

We chose the GIVE environment in order to test REs referring both to attributes of an object, i.e. color, and to its spatial relation to other visible objects in the scene. The TUNA Corpus was chosen as a more challenging domain, due to the greater number of available properties for each object on a scene.

Each experimental subject was presented with screenshots containing a marked object and an RE. Subjects were told that we had previously referred to the marked object with the given RE, but an (imaginary) player misunderstood this RE and selected a different object, shown in a second screenshot. They were then asked to select which one of two corrections they considered better, where "better" was intentionally left unspecific. Figs. 7 and 8 show examples for each domain. The full set of stimuli is available as supplementary material.

To maintain annotation quality in our crowdsourcing setting, we designed test items with a

clearly incorrect answer, such as REs referring to the wrong target or a nonexistent one. These test items were randomly interspersed with the real stimuli, and only subjects with a perfect score on the test items were taken into account. Experimental subjects were asked to rate up to 12 comparisons, shown in groups of 3 scenes at a time, and were automatically disqualified if they evaluated any individual scene in less than 10 seconds. The order in which the pairs of strategies were shown was randomized, to avoid effects related to the order in which they were presented on screen.

7.2 Experiment 1

Our first experiment tested four strategies against each other. Each experimental subject was presented with two screenshots of 3D scenes with a marked object and an RE (see Fig. 7 for an example). Each subject was shown a total of 12 scenes, selected at random from 16 test scenes. We collected 10 judgments for each possible combination of GIVE scene and pair of strategies, yielding a total of 943 judgements from 142 subjects after removing fake answers.

We compared the *Emphasis* and *Shortening* strategies from Section 6 against two baselines. The *Repeat* strategy simply presented r_s as a "contrastive" RE, without any capitalization. Comparisons to *Repeat* test the hypothesis that subjects prefer explicit contrastive focus. The *Random* strategy randomly capitalized adjectives, adverbs, and/or prepositions that were not capitalized by the *Emphasis* strategy. Comparisons to *Random* verify that any preference for *Emphasis* is not only due to the presence of contrastive focus, but also because our method identifies precisely where that focus should be.

Table 1a shows the results of all pairwise comparisons. For each row strategy $Strat_R$ and each column strategy $Strat_C$, the table value corresponds to

$$\frac{(\#Strat_R \text{ pref. over } Strat_C) - (\#Strat_C \text{ pref. over } Strat_R)}{(\#\text{ tests between } Strat_R \text{ and } Strat_C)}$$

Significance levels are taken from a two-tailed binomial test over the counts of preferences for each strategy. We find a significant preference for the *Emphasis* strategy over all others, providing evidence that our algorithm assigns contrastive focus to the right words in the corrective RE.

While the *Shortening* strategy is numerically preferred over both baselines, the difference is not significant, and it is significantly worse than the *Emphasis* strategy. This is surprising, given our initial assumption that listeners prefer succinct REs. It is possible that a different strategy for shortening contrastive REs would work better; this bears further study.

7.3 Experiment 2

In our second experiment, we paired the *Emphasis*, *Repeat*, and *Random* strategies against each other, this time evaluating each strategy in the TUNA people domain. Due to its poor performance in Experiment 1, which was confirmed in pilot experiments for Experiment 2, the *Shortening* strategy was not included.

The experimental setup for the TUNA domain used 3x4 grids of pictures of people chosen at random from the TUNA Challenge, as shown in Fig. 8. We generated 8 such grids, along with REs ranging from two to five attributes and requiring one or two attributes to establish the correct contrast. The larger visual size of objects in the the TUNA scenes allowed us to mark both o_s and o_u in a single picture without excessive clutter.

The REs for Experiment 2 were designed to only include attributes from the referred objects, but no information about its position in relation to other objects. The benefit is twofold: we avoid taxing our subjects' memory with extremely long REs, and we ensure that the overall length of the second set of REs is comparable to those in the previous experiment.

We obtained 240 judgements from 65 subjects (after removing fake answers). Table 1b shows the results of all pairwise comparisons. We find that even in the presence of a larger number of attributes, our algorithm assigns contrastive focus to the correct words of the RE.

7.4 Discussion

Our experiments confirm that the strategy for computing contrastive REs presented in this paper works in practice. This validates the corruption model, which approximates semantic mismatches between what the speaker said and what the listener understood as differences at the level of words in strings. Obviously, this model is still an approximation, and we will test its limits in future work.

We find that users generally prefer REs with an emphasis over simple repetitions. In the more challenging scenes of the TUNA corpus, users even have a significant preference of Random over

	Repeat	Random	Emphasis	Shortening
Repeat	–	0.041	-0.570***	-0.141
Random	-0.041	–	-0.600***	-0.109
Emphasis	0.570***	0.600***	–	0.376***
Shortening	0.141	0.109	-0.376***	–

(a) Results for Experiment 1

	Repeat	Random	Emphasis
Repeat	–	-0.425***	-0.575***
Random	0.425***	–	-0.425***
Emphasis	0.575***	0.425***	–

(b) Results for Experiment 2

Table 1: Pairwise comparisons between feedback strategies for experiments 1 and 2. A positive value shows preference for the row strategy, significant at *** $p < 0.001$.

Repeat, although this makes no semantic sense. This preference may be due to the fact that emphasizing *anything* at least publically acknowledges the presence of a misunderstanding that requires correction. It will be interesting to explore whether this preference holds up in an interactive setting, rather than an overhearer experiment, where listeners will have to act upon the corrective REs.

The poor performance of the Shortening strategy is a surprising negative result. We would expect a shorter RE to always be preferred, following the Gricean Maxim of Quantity (Grice, 1975). This may because our particular Shortening strategy can be improved, or it may be because listeners interpret the shortened REs not with respect to the original instructions, but rather with respect to a "refreshed" context (as observed, for instance, in Gotzner et al. (2016)). In this case the shortened REs would not be unique with respect to the refreshed, wider context.

8 Conclusion

In this paper, we have presented an algorithm for generating contrastive feedback for a hearer who has misunderstood a referring expression. Our technique is based on modeling likely user misunderstandings and then attempting to give feedback that contrasts with the most probable incorrect understanding. Our experiments show that this technique accurately predicts which words to mark as focused in a contrastive RE.

In future work, we will complement the overhearer experiment presented here with an end-to-end evaluation in an interactive NLG setting. This will allow us to further investigate the quality of the correction strategies and refine the Shortening strategy. It will also give us the opportunity to investigate empirically the limits of the corruption model. Furthermore, we could use this data to refine the costs $c(D), c(I_a)$ etc. for the edit operations, possibly assigning different costs to different edit operations.

Finally, it would be interesting to combine our algorithm with a speech synthesis system. In this way, we will be able to express focus with actual pitch accents, in contrast to the typographic approximation we made here.

References

Saskia Akkersdijk, Marin Langenbach, Frieder Loch, and Mariët Theune. 2011. The thumbs up! twente system for give 2.5. In *The 13th European Workshop on Natural Language Generation (ENLG 2011)*.

Yehoshua Bar-Hillel, Micha Perles, and Eli Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 14:143–172.

Ina Bornkessel and Matthias Schlesewsky. 2006. The role of contrast in the local licensing of scrambling in german: Evidence from online comprehension. *Journal of Germanic Linguistics* 18(01):1–43.

Okko Buß, Timo Baumann, and David Schlangen. 2010. Collaborating on utterances with a spoken dialogue system using an isu–based approach to incremental dialogue management. In *Proceedings of the Special Interests Group on Discourse and Dialogue Conference (SIGdial 2010)*.

Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, Marc Tommasi, and Christof Löding. 2007. *Tree Automata techniques and applications*. published online - `http://tata.gforge.inria.fr/`. http://tata.gforge.inria.fr/.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* 19(2):233–263.

Nikos Engonopoulos and Alexander Koller. 2014. Generating effective referring expressions using charts. In *Proceedings of the INLG and SIGdial 2014 Joint Session*.

Nikos Engonopoulos, Martín Villalba, Ivan Titov, and Alexander Koller. 2013. Predicting the resolution of referring expressions from user behavior. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*.

Nicole Gotzner, Isabell Wartenburger, and Katharina Spalek. 2016. The impact of focus particles on the recognition and rejection of contrastive alternatives. *Language and Cognition* 8(1):59–95.

H. Paul Grice. 1975. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, Academic Press, pages 41–58.

John Edward Hopcroft and Jeffrey Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

David Howcroft, Crystal Nakatsu, and Michael White. 2013. Enhancing the expression of contrast in the SPaRKy restaurant corpus. In *Proceedings of the 14th European Workshop on Natural Language Generation (ENLG 2013)*.

Alexander Koller, Kristina Striegnitz, Andrew Gargett, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010. Report on the Second NLG Challenge on Generating Instructions in Virtual Environments (GIVE-2). In *Proceedings of the Sixth International Natural Language Generation Conference (Special session on Generation Challenges)*.

E. Krahmer and M. Theune. 2002. Efficient context-sensitive generation of referring expressions. In K. van Deemter and R. Kibble, editors, *Information Sharing: Reference and Presupposition in Language Generation and Interpretation, Center for the Study of Language and Information-Lecture Notes*, CSLI Publications, volume 143, pages 223–263.

Manfred Krifka. 2008. Basic notions of information structure. *Acta Linguistica Hungarica* 55:243–276.

Willem J.M. Levelt. 1993. *Speaking: From Intention to Articulation*. MIT University Press Group.

Maria Milosavljevic and Robert Dale. 1996. Strategies for comparison in encyclopædia descriptions. In *Proceedings of the 8th International Natural Language Generation Workshop (INLG 1996)*.

Teruhisa Misu, Antoine Raux, Rakesh Gupta, and Ian Lane. 2014. Situated language understanding at 25 miles per hour. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGdial 2014)*.

Mehryar Mohri. 2003. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science* 14(6):957–982.

Janet B. Pierrehumbert and Julia Hirschberg. 1990. The meaning of intonational contours in the interpretation of discourse. In Philip R. Cohen, Jerry Morgan, and Martha E. Pollack, editors, *Intentions in Communication*, MIT University Press Group, chapter 14.

Mats Rooth. 1992. A theory of focus interpretation. *Natural Language Semantics* 1:75–116.

Mats Rooth. 1997. Focus. In Shalom Lappin, editor, *The Handbook of Contemporary Semantic Theory*, Blackwell Publishing, chapter 10, pages 271–298.

Anita Steube. 2001. Correction by contrastive focus. *Theoretical Linguistics* 27(2-3):215–250.

Ielka van der Sluis, Albert Gatt, and Kees van Deemter. 2007. Evaluating algorithms for the generation of referring expressions: Going beyond toy domains. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*.

Sina Zarrieß and David Schlangen. 2016. Easy Things First: Installments Improve Referring Expression Generation for Objects in Photographs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Modeling Source Syntax for Neural Machine Translation

Junhui Li[†] **Deyi Xiong**[†] **Zhaopeng Tu**[‡*]
Muhua Zhu[‡] **Min Zhang**[†] **Guodong Zhou**[†]
[†]School of Computer Science and Technology,
Soochow University, Suzhou, China
{lijunhui, dyxiong, minzhang, gdzhou}@suda.edu.cn
[‡]Tencent AI Lab, Shenzhen, China
tuzhaopeng@gmail.com, muhuazhu@tencent.com

Abstract

Even though a linguistics-free sequence to sequence model in neural machine translation (NMT) has certain capability of implicitly learning syntactic information of source sentences, this paper shows that source syntax can be explicitly incorporated into NMT effectively to provide further improvements. Specifically, we linearize parse trees of source sentences to obtain structural label sequences. On the basis, we propose three different sorts of encoders to incorporate source syntax into NMT: 1) *Parallel RNN* encoder that learns word and label annotation vectors parallelly; 2) *Hierarchical RNN* encoder that learns word and label annotation vectors in a two-level hierarchy; and 3) *Mixed RNN* encoder that stitchingly learns word and label annotation vectors over sequences where words and labels are mixed. Experimentation on Chinese-to-English translation demonstrates that all the three proposed syntactic encoders are able to improve translation accuracy. It is interesting to note that the simplest RNN encoder, i.e., *Mixed RNN* encoder yields the best performance with an significant improvement of 1.4 BLEU points. Moreover, an in-depth analysis from several perspectives is provided to reveal how source syntax benefits NMT.

1 Introduction

Recently the sequence to sequence model (seq2seq) in neural machine translation (NMT) has achieved certain success over the state-of-the-art of statistical machine translation (SMT) on various language pairs (Bahdanau et al., 2015; Jean et al., 2015; Luong et al., 2015; Luong and Manning, 2015). However, Shi et al. (2016) show that the seq2seq model still fails to capture a lot of deep structural details, even though it is capable of learning certain implicit source syntax from sentence-aligned parallel corpus. Moreover, it requires an additional parsing-task-specific training mechanism to recover the hidden syntax in NMT. As a result, in the absence of explicit linguistic knowledge, the seq2seq model in NMT tends to produce translations that fail to well respect syntax. In this paper, we show that syntax can be well exploited in NMT explicitly by taking advantage of source-side syntax to improve the translation accuracy.

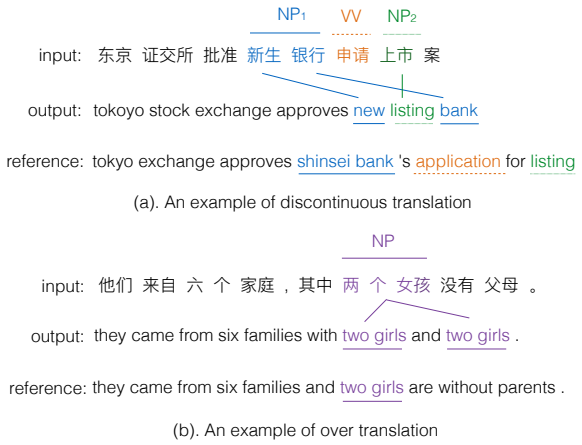


Figure 1: Examples of NMT translation that fail to respect source syntax.

In principle, syntax is a promising avenue for translation modeling. This has been verified by tremendous encouraging studies on syntax-based SMT that substantially improves translation by integrating various kinds of syntactic knowledge (Liu et al., 2006; Marton and Resnik, 2008;

*Work done at Huawei Noah's Ark Lab, HongKong.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 688–697
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1064

Shen et al., 2008; Li et al., 2013). While it is yet to be seen how syntax can benefit NMT effectively, we find that translations of NMT sometimes fail to well respect source syntax. Figure 1 (a) shows a Chinese-to-English translation example of NMT. In this example, the NMT seq2seq model incorrectly translates the Chinese noun phrase (i.e., 新生/*xinsheng* 银行/*yinhang*) into a discontinuous phrase in English (i.e., *new ... bank*) due to the failure of capturing the internal syntactic structure in the input Chinese sentence. Statistics on our development set show that one forth of Chinese noun phrases are translated into discontinuous phrases in English, indicating the substantial disrespect of syntax in NMT translation.[1] Figure 1 (b) shows another example with over translation, where the noun phrase 两/*liang* 个/*ge* 女孩/*nvhai* is translated twice in English. Similar to discontinuous translation, over translation usually happens along with the disrespect of syntax which results in the repeated translation of the same source words in multiple positions of the target sentence.

In this paper we are not aiming at solving any particular issue, either the discontinuous translation or the over translation. Alternatively, we address how to incorporate explicitly the source syntax to improve the NMT translation accuracy with the expectation of alleviating the issues above in general. Specifically, rather than directly assigning each source word with manually designed syntactic labels, as Sennrich and Haddow (2016) do, we linearize a phrase parse tree into a *structural label sequence* and let the model automatically learn useful syntactic information. On the basis, we systematically propose and compare several different approaches to incorporating the label sequence into the seq2seq NMT model. Experimentation on Chinese-to-English translation demonstrates that all proposed approaches are able to improve the translation accuracy.

2 Attention-based NMT

As a background and a baseline, in this section, we briefly describe the NMT model with an attention mechanism by Bahdanau et al. (2015), which mainly consists of an encoder and a decoder, as shown in Figure 2.

Encoder The encoding of a source sentence is formulated using a pair of neural networks, i.e., two recurrent neural networks (denoted *bi-RNN*): one reads an input sequence $x = (x_1, ..., x_m)$ from left to right and outputs a forward sequence of hidden states $(\overrightarrow{h_1}, ..., \overrightarrow{h_m})$, while the other operates from right to left and outputs a backward sequence $(\overleftarrow{h_1}, ..., \overleftarrow{h_m})$. Each source word x_j is represented as h_j (also referred to as word annotation vector): the concatenation of hidden states $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$. Such bi-RNN encodes not only the word itself but also its left and right context, which can provide important evidence for its translation.

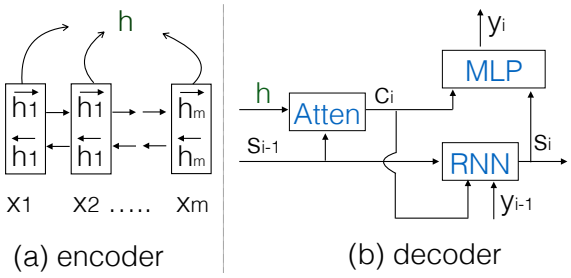


Figure 2: Attention-based NMT model.

Decoder The decoder is also an RNN that predicts a target sequence $y = (y_1, ..., y_n)$. Each target word y_i is predicted via a multi-layer perceptron (MLP) component which is based on a recurrent hidden state s_i, the previous predicted word y_{i-1}, and a source-side context vector c_i. Here, c_i is calculated as a weighted sum over source annotation vectors $(h_1, ..., h_m)$. The weight vector $\alpha_i \in \mathbb{R}^m$ over source annotation vectors is obtained by an attention model, which captures the correspondences between the source and the target languages. The attention weight α_{ij} is computed based on the previous recurrent hidden state s_{i-1} and source annotation vector h_j.

3 NMT with Source Syntax

The conventional NMT models treat a sentence as a sequence of words and ignore external knowledge, failing to effectively capture various kinds of inherent structure of the sentence. To leverage external knowledge, specifically the syntax in the source side, we focus on the parse tree of a sentence and propose three different NMT models that explicitly consider the syntactic structure into encoding. Our purpose is to inform the NMT model the structural context of each word in its corresponding parse tree with the goal that the learned annotation vectors $(h_1, ..., h_m)$ encode not

[1] Manually examining 200 random such discontinuously translated noun phrases, we find that 90% of them should be continuously translated according to the reference translation.

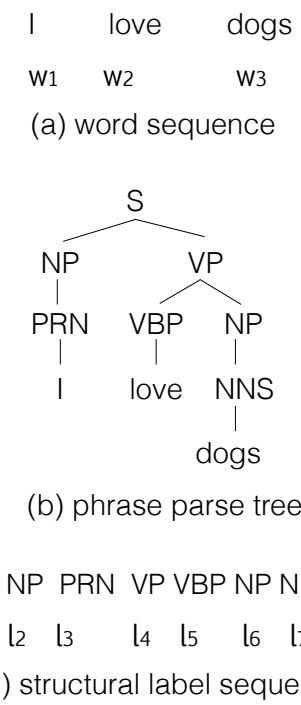


Figure 3: An example of an input sentence (a), its parse tree (b), and the parse tree's sequential form (c).

only the information of words and their surroundings, but also structural context in the parse tree. In the rest of this section, we use English sentences as examples to explain our methods.

3.1 Syntax Representation

To obtain the structural context of a word in its parse tree, ideally the model should not only capture and remember the whole parse tree structure, but also discriminate the contexts of any two different words. However, considering the lack of efficient way to directly model structural information, an alternative way is to linearize the phrase parse tree into a sequence of structural labels and learn the structural context through the sequence. For example, Figure 3(c) shows the structural label sequence of Figure 3(b) in a simple way following a depth-first traversal order. Note that linearizing a parse tree in a depth-first traversal order into a sequence of structural labels has also been widely adopted in recent advances in neural syntactic parsing (Vinyals et al., 2015; Choe and Charniak, 2016), suggesting that the linearized sequence can be viewed as an alternative to its tree structure.[2]

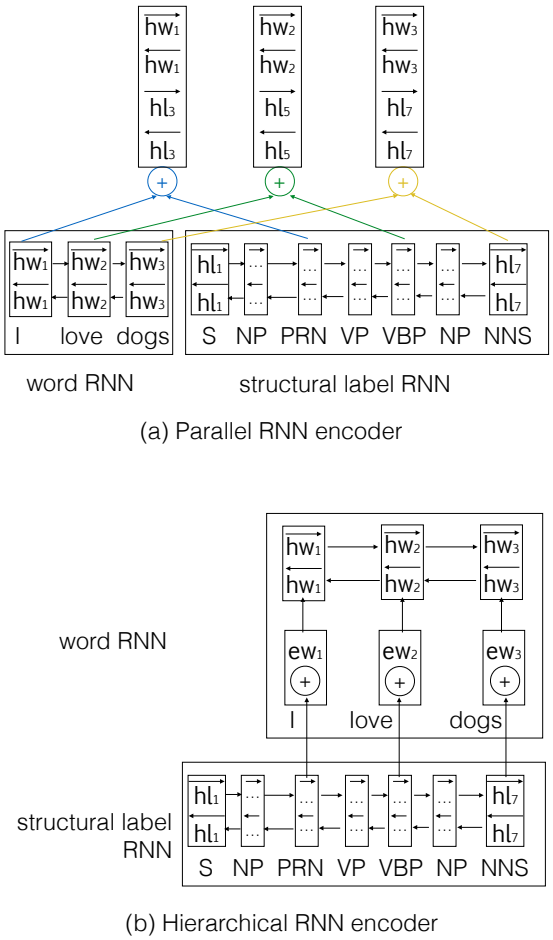


Figure 4: The graphical illustration of the *Parallel RNN* encoder (a) and the *Hierarchical RNN* encoder (b). Here, $\overrightarrow{hw_j}$ and $\overleftarrow{hw_j}$ are the forward and backward hidden states for word w_j, $\overrightarrow{hl_i}$ and $\overleftarrow{hl_i}$ are for structural label l_i, ew_j is the word embedding for word w_j, and $\bigoplus$ is for concatenation operator.

There is no doubt that the structural label sequence is much longer than its word sequence. In order to obtain the structural label annotation vector for w_i in word sequence, we simply look for w_i's part-of-speech (POS) tag in the label sequence and view the tag's annotation vector as w_i's label annotation vector. This is because w_i's POS tag location can also represent w_i's location in the parse tree. For example, in Figure 3, word w_1 in (a) maps to l_3 in (c) since l_3 is the POS tag of w_1. Likewise, w_2 maps to l_5 and w_3 to l_7. That is to say, we use l_3's learned annotation vector as w_1's label annotation vector.

[2]We have also tried to include the ending brackets in the structural label sequence, as what (Vinyals et al., 2015; Choe and Charniak, 2016) do. However, the performance gap is very small by adding the ending brackets or not.

3.2 RNN Encoders with Source Syntax

In the next, we first propose two different encoders to augment word annotation vector with its corresponding label annotation vector, each of which consists of two RNNs [3]: in one encoder, the two RNNs work independently (i.e., *Parallel RNN Encoder*) while in another encoder the two RNNs work in a hierarchical way (i.e., *Hierarchical RNN Encoder*). The difference between the two encoders lies in how the two RNNs interact. Then, we propose the third encoder with a single RNN, which learns word and label annotation vectors stitchingly (i.e., *Mixed RNN Encoder*). Since any of the above three approaches focuses only on the encoder as to generate source annotation vectors along with structural information, we keep the rest part of the NMT models unchanged.

Parallel RNN Encoder Figure 4 (a) illustrates our *Parallel RNN* encoder, which includes two parallel RNNs: i.e., a word RNN and a structural label RNN. On the one hand, the word RNN, as in conventional NMT models, takes a word sequence as input and output a word annotation vector for each word. On the other hand, the structural label RNN takes the structural label sequence of the word sequence as input and obtains a label annotation vector for each label. Besides, we concatenate each word's word annotation vector and its POS tag's label annotation vector as the final annotation vector for the word. For example, the final annotation vector for word *love* in Figure 4 (a) is $[\overrightarrow{hw_2}; \overleftarrow{hw_2}; \overrightarrow{hl_5}; \overleftarrow{hl_5}]$, where the first two subitems $[\overrightarrow{hw_2}; \overleftarrow{hw_2}]$ are the word annotation vector and the rest two subitems $[\overrightarrow{hl_5}; \overleftarrow{hl_5}]$ are its POS tag *VBP*'s label annotation vector.

Hierarchical RNN Encoder Partially inspired by the model architecture of GNMT (Wu et al., 2016) which consists of multiple layers of LSTM RNNs, we propose a two-layer model architecture in which the lower layer is the structural label RNN while the upper layer is the word RNN, as shown in Figure 4 (b). We put the word RNN in the upper layer because each item in the word sequence can map into an item in the structural label sequence, while this does not hold if the order of the two RNNs is reversed. As shown in Figure 4 (b), for example, the POS tag *VBP*'s label annotation vector $[\overrightarrow{hl_5}, \overleftarrow{hl_5}]$ is concatenated with word

[3] Hereafter, we simplify bi-RNN as RNN.

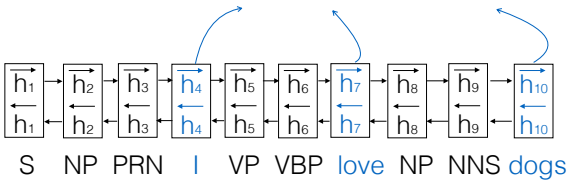


Figure 5: The graphical illustration of the *Mixed RNN* encoder. Here, $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$ are the forward and backward hidden annotation vectors for *j*-th item, which can be either a word or a structural label.

love's word embedding ew_2 to feed as the input to the word RNN.

Mixed RNN Encoder Figure 5 presents our *Mixed RNN* encoder. Similarly, the sequence of input is the linearization of its parse tree (as in Figure 3 (b)) following a depth-first traversal order, but being mixed with both words and structural labels in a stitching way. It shows that the RNN learns annotation vectors for both the words and the structural labels, though only the annotation vectors of words are further fed to decoding (e.g., $([\overrightarrow{h_4}, \overleftarrow{h_4}], [\overrightarrow{h_7}, \overleftarrow{h_7}], [\overrightarrow{h_{10}}, \overleftarrow{h_{10}}])$). Even though the annotation vectors of structural labels are not directly fed forward for decoding, the error signal is back propagated along the word sequence and allows the annotation vectors of structural labels being updated accordingly.

3.3 Comparison of RNN Encoders with Source Syntax

Though all the three encoders model both word sequence and structural label sequence, the differences lie in their respective model architecture with respect to the degree of coupling the two sequences:

- In the *Parallel RNN* encoder, the word RNN and structural label RNN work in a parallel way. That is to say, the error signal back propagated from the word sequence would not affect the structural label RNN, and vice versa. In contrast, in the *Hierarchical RNN* encoder, the error signal back propagated from the word sequence has a direct impact on the structural label annotation vectors, and thus on the structural label embeddings. Finally, the *Mixed RNN* encoder ties the structural label sequence and word sequence together in the closest way. Therefore, the degrees of coupling the word and structural

label sequences in these three encoders are like this: *Mixed RNN* encoder > *Hierarchical RNN* encoder > *Parallel RNN* encoder.

- Figure 4 and Figure 5 suggest that the *Mixed RNN* encoder is the simplest. Moreover, comparing to conventional NMT encoders, the difference lies only in the length of the input sequence. Statistics on our training data reveal that the *Mixed RNN* encoder approximately triples the input sequence length compared to conventional NMT encoders.

4 Experimentation

We have presented our approaches to incorporating the source syntax into NMT encoders. In this section, we evaluate their effectiveness on Chinese-to-English translation.

4.1 Experimental Settings

Our training data for the translation task consists of 1.25M sentence pairs extracted from LDC corpora, with 27.9M Chinese words and 34.5M English words respectively.[4] We choose NIST MT 06 dataset (1664 sentence pairs) as our development set, and NIST MT 02, 03, 04, and 05 datasets (878, 919, 1788 and 1082 sentence pairs, respectively) as our test sets.[5] To get the source syntax for sentences on the source-side, we parse the Chinese sentences with Berkeley Parser [6] (Petrov and Klein, 2007) trained on Chinese TreeBank 7.0 (Xue et al., 2005). We use the case insensitive 4-gram NIST BLEU score (Papineni et al., 2002) for the translation task.

For efficient training of neural networks, we limit the maximum sentence length on both source and target sides to 50. We also limit both the source and target vocabularies to the most frequent 16K words in Chinese and English, covering approximately 95.8% and 98.2% of the two corpora respectively. All the out-of-vocabulary words are mapped to a special token *UNK*. Besides, the word embedding dimension is 620 and the size of a hidden layer is 1000. All the other settings are the same as in Bahdanau et al.(2015).

The inventory of structural labels includes 16 phrase labels and 32 POS tags. In both our *Parallel RNN* encoder and *Hierarchical RNN* encoder, we set the embedding dimension of these labels as 100 and the size of a hidden layer as 100. Besides, the maximum structural label sequence length is set to 100. In our *Mixed RNN* encoder, since we only have one input sequence, we equally treat the structural labels and words (i.e., a structural label is also initialized with 620 dimension embedding). Compared to the baseline NMT model, the only different setting is that we increase the maximum sentence length on source-side from 50 to 150.

We compare our method with two state-of-the-art models of SMT and NMT:

- cdec (Dyer et al., 2010): an open source hierarchical phrase-based SMT system (Chiang, 2007) with default configuration and a 4-gram language model trained on the target portion of the training data.[7]

- RNNSearch: a re-implementation of the attentional NMT system (Bahdanau et al., 2015) with slight changes taken from dl4mt tutorial.[8] For the activation function f of an RNN, RNNSearch uses the gated recurrent unit (GRU) recently proposed by (Cho et al., 2014b). It incorporates dropout (Hinton et al., 2012) on the output layer and improves the attention model by feeding the lastly generated word. We use AdaDelta (Zeiler, 2012) to optimize model parameters in training with the mini-batch size of 80. For translation, a beam search with size 10 is employed.

4.2 Experiment Results

Table 1 shows the translation performances measured in BLEU score. Clearly, all the proposed NMT models with source syntax improve the translation accuracy over all test sets, although there exist considerable differences among different variants.

Parameters The three proposed models introduce new parameters in different ways. As a baseline model, RNNSearch has 60.6M parameters. Due to the infrastructure similarity, the *Parallel RNN* system and the *Hierarchical RNN* system introduce

[4] The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

[5] http://www.itl.nist.gov/iad/mig/tests/mt/

[6] https://github.com/slavpetrov/berkeleyparser

[7] https://github.com/redpony/cdec

[8] https://github.com/nyu-dl/dl4mt-tutorial

#	System	#Params	Time	MT06	MT02	MT03	MT04	MT05	All
1	cdec	-	-	33.4	34.8	33.0	35.7	32.1	34.2
2	RNNSearch	60.6M	153m	34.0	36.9	33.7	37.0	34.1	35.6
3	Parallel RNN	+1.1M	+9m	34.8†	**37.8**‡	34.2	38.3‡	34.6	36.6‡
4	Hierarchical RNN	+1.2M	+9m	35.2‡	37.2	34.7†	38.7‡	34.7†	36.7‡
5	Mixed RNN	+0	+40m	**35.6**‡	37.7†	**34.9**‡	**38.6**‡	**35.5**‡	**37.0**‡

Table 1: Evaluation of the translation performance. † and ‡: significant over RNNSearch at 0.05/0.01, tested by bootstrap resampling (Koehn, 2004). "+" is the additional number of parameters or training time on the top of the baseline system RNNSearch. Column *Time* indicates the training time in minutes per epoch for different NMT models

the similar size of additional parameters, resulting from the RNN model for structural label sequences (about 0.1M parameters) and catering either the augmented annotation vectors (as shown in Figure 4 (a)) or the augmented word embeddings (as shown in Figure 4 (b)) (the remain parameters). It is not surprising that the *Mixed RNN* system does not require any additional parameters since though the input sequence becomes longer, we keep the vocabulary size unchanged, resulting in no additional parameters.

Speed Introducing the source syntax slightly slows down the training speed. When running on a single GPU GeForce GTX 1080, the baseline model speeds 153 minutes per epoch with 14K updates while the proposed structural label RNNs in both *Parallel RNN* and *Hierarchical RNN* systems only increases the training time by about 6% (thanks to the small size of structural label embeddings and annotation vectors), and the *Mixed RNN* system spends 26% more training time to cater the triple sized input sequence.

Comparison with the baseline NMT model (RNNSearch) While all the three proposed NMT models outperform RNNSearch, the *Parallel RNN* system and the *Hierarchical RNN* system achieve similar accuracy (e.g., 36.6 *v.s.* 36.7). Besides, the *Mixed RNN* system achieves the best accuracy overall test sets with the only exception of NIST MT 02. Over all test sets, it outperforms RNNSearch by 1.4 BLEU points and outperforms the other two improved NMT models by 0.3∼0.4 BLEU points, suggesting the benefits of high degree of coupling the word sequence and the structural label sequence. This is very encouraging since the *Mixed RNN* encoder is the simplest, without introducing new parameters and with only slight additional training time.

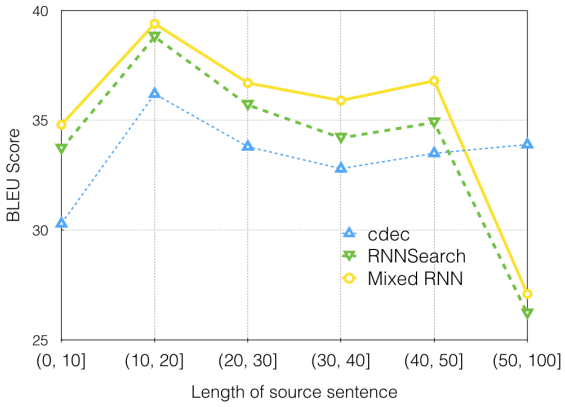


Figure 6: Performance of the generated translations with respect to the lengths of the input sentences.

Comparison with the SMT model (cdec) Table 1 also shows that all NMT systems outperform the SMT system. This is very consistent with other studies on Chinese-to-English translation (Mi et al., 2016; Tu et al., 2017b; Wang et al., 2017).

5 Analysis

As the proposed *Mixed RNN* system achieves the best performance, we further look at the RNNSearch system and the *Mixed RNN* system to explore more on how syntactic information helps in translation.

5.1 Effects on Long Sentences

Following Bahdanau et al. (2015), we group sentences of similar lengths together and compute BLEU scores. Figure 6 presents the BLEU scores over different lengths of input sentences. It shows that *Mixed RNN* system outperforms RNNSearch over sentences with all different lengths. It also shows that the performance drops substantially

System	AER
RNNSearch	50.1
Mixed RNN	47.9

Table 2: Evaluation of alignment quality. The lower the score, the better the alignment quality.

when the length of input sentences increases. This performance trend over the length is consistent with the findings in (Cho et al., 2014a; Tu et al., 2016, 2017a). We also observe that the NMT systems perform surprisingly bad on sentences over 50 in length, especially compared to the performance of SMT system (i.e., cdec). We think that the bad behavior of NMT systems towards long sentences (e.g., length of 50) is due to the following two reasons: (1) the maximum source sentence length limit is set as 50 in training, [9] making the learned models not ready to translate sentences over the maximum length limit; (2) NMT systems tend to stop early for long input sentences.

5.2 Analysis on Word Alignment

Due to the capability of carrying syntactic information in source annotation vectors, we conjecture that our model with source syntax is also beneficial for alignment. To test this hypothesis, we carry out experiments of the word alignment task on the evaluation dataset from Liu and Sun (2015), which contains 900 manually aligned Chinese-English sentence pairs. We force the decoder to output reference translations, as to get automatic alignments between input sentences and their reference translations. To evaluate alignment performance, we report the alignment error rate (AER) (Och and Ney, 2003) in Table 2.

Table 2 shows that source syntax information improves the attention model as expected by maintaining an annotation vector summarizing structural information on each source word.

5.3 Analysis on Phrase Alignment

The above subsection examines the alignment performance at the word level. In this subsection, we turn to phrase alignment analysis by moving from word unit to phrase unit. Given a source phrase *XP*, we use word alignments to examine if the phrase is translated continuously (**Cont.**), or discontinuously (**Dis.**), or if it is not translated at all (**Un.**).

[9] Though the maximum source length limit in *Mixed RNN* system is set to 150, it approximately contains 50 words in maximum.

System	XP	Cont.	Dis.	Un.
RNNSearch	PP	57.3	33.6	9.1
	NP	59.8	25.5	14.7
	CP	47.3	44.6	8.1
	QP	54.0	22.2	23.8
	ALL	58.1	27.1	14.8
Mixed RNN	PP	63.3	27.5	9.2
	NP	63.1	23.1	13.8
	CP	54.5	36.6	8.9
	QP	56.2	19.7	24.1
	ALL	60.4	25.0	14.6

Table 3: Percentages (%) of syntactic phrases in our test sets being translated continuously, discontinuously, or not being translated. Here PP is for prepositional phrase, NP for noun phrase, CP for clause headed by a complementizer, QP for quainter phrase.

There are some phrases, such as noun phrases (NPs), prepositional phrases (PPs) that we usually expect to have a continuous translation. With respect to several such types of phrases, Table 3 shows how these phrases are translated. From the table, we see that translations of RNNSearch system do not respect source syntax very well. For example, in RNNSearch translations, 57.3%, 33.6%, and 9.1% of PPs are translated continuously, discontinuously, and untranslated, respectively. Fortunately, our *Mixed RNN* system is able to have more continuous translation for those phrases. Table 3 also suggests that there is still much room for NMT to show more respect to syntax.

5.4 Analysis on Over Translation

To estimate the over translation generated by NMT, we propose *ratio of over translation (ROT)*:

$$ROT = \frac{\sum_{w_i} t(w_i)}{|w|} \quad (1)$$

where $|w|$ is the number of words in consideration, $t(w_i)$ is the times of over translation for word w_i. Given a word w and its translation $e = e_1 e_2 \dots e_n$, we have:

$$t(w) = |e| - |uniq(e)| \quad (2)$$

where $|e|$ is the number of words in w's translation e, while $|uniq(e)|$ is the number of unique words in e. For example, if a source word 香

System	POS	ROT (%)
RNNSearch	NR	15.7
	CD	7.4
	DT	4.9
	NN	8.0
	ALL	5.5
Mixed RNN	NR	12.3
	CD	5.1
	DT	2.4
	NN	6.8
	ALL	4.5

Table 4: Ratio of over translation (ROT) on test sets. Here NR is for proper noun, CD for cardinal number, DT for determiner, and NN for nouns except proper nouns and temporal nouns.

System	POS	non-UNK	UNK	Un.
RNNSearch	NN	27.2	40.4	32.4
	NR	22.9	58.5	18.6
	VV	34.5	32.9	32.7
	CD	10.7	63.4	25.9
	ALL	27.2	40.4	32.4
Mixed RNN	NN	24.8	41.4	33.8
	NR	17.0	64.5	18.6
	VV	33.6	34.0	32.3
	CD	9.6	68.7	21.7
	ALL	23.9	47.5	28.7

Table 5: Percentages (%) of rare words in our test sets being translated into a *non-UNK* word (**non-UNK**), *UNK* (**UNK**), or if it is not translated at all (**Un.**).

港/xiangkang is translated as *hong kong hong kong*, we say it being over translated *2* times.

Table 4 presents ROT grouped by some typical POS tags. It is not surprising that RNNSearch system has high ROT with respect to POS tags of NR (proper noun) and CD (cardinal number): this is due to the fact that the two POS tags include high percentage of unknown words which tend to be translated multiple times in translation. Words of DT (determiner) are another source of over translation since they are usually translated to multiple *the* in English. It also shows that by introducing source syntax, *Mixed RNN* system alleviates the over translation issue by 18%: ROT drops from 5.5% to 4.5%.

5.5 Analysis on Rare Word Translation

We analyze the translation of source-side rare words that are mapped to a special token *UNK*. Given a rare word *w*, we examine if it is translated into a *non-UNK* word (**non-UNK**), *UNK* (**UNK**), or if it is not translated at all (**Un.**).

Table 5 shows how source-side rare words are translated. The four POS tags listed in the table account for about 90% of all rare words in the test sets. It shows that in *Mixed RNN* system is more likely to translate source-side rare words into *UNK* on the target side. This is reasonable since the source side rare words tends to be translated into rare words in the target side. Moreover, it is hard to obtain its correct non-*UNK* translation when a source-side rare word is replaced as *UNK*.

Note that our approach is compatible with with approaches of open vocabulary. Taking the subword approach (Sennrich et al., 2016) as an example, for a word on the source side which is divided into several subword units, we can synthesize sub-POS nodes that cover these units. For example, if *misunderstand/VB* is divided into units of *mis* and *understand*, we construct substructure *(VB (VB-F mis) (VB-I understand))*.

6 Related Work

While there has been substantial work on linguistically motivated SMT, approaches that leverage syntax for NMT start to shed light very recently. Generally speaking, NMT can provide a flexible mechanism for adding linguistic knowledge, thanks to its strong capability of automatically learning feature representations.

Eriguchi et al. (2016) propose a tree-to-sequence model that learns annotation vectors not only for terminal words, but also for non-terminal nodes. They also allow the attention model to align target words to non-terminal nodes. Our approach is similar to theirs by using source-side phrase parse tree. However, our *Mixed RNN* system, for example, incorporates syntax information by learning annotation vectors of syntactic labels and words stitchingly, but is still a sequence-to-sequence model, with no extra parameters and with less increased training time.

Sennrich and Haddow (2016) define a few linguistically motivated features that are attached to each individual words. Their features include lemmas, subword tags, POS tags, dependency labels, etc. They concatenate feature embeddings with word embeddings and feed the concatenated em-

beddings into the NMT encoder. On the contrast, we do not specify any feature, but let the model implicitly learn useful information from the structural label sequence.

Shi et al. (2016) design a few experiments to investigate if the NMT system without external linguistic input is capable of learning syntactic information on the source-side as a by-product of training. However, their work is not focusing on improving NMT with linguistic input. Moreover, we analyze what syntax is disrespected in translation from several new perspectives.

García-Martínez et al. (2016) generalize NMT outputs as lemmas and morphological factors in order to alleviate the issues of large vocabulary and out-of-vocabulary word translation. The lemmas and corresponding factors are then used to generate final words in target language. Though they use linguistic input on the target side, they are limited to the word level features. Phrase level, or even sentence level linguistic features are harder to obtain for a generation task such as machine translation, since this would require incremental parsing of the hypotheses at test time.

7 Conclusion

In this paper, we have investigated whether and how source syntax can explicitly help NMT to improve its translation accuracy.

To obtain syntactic knowledge, we linearize a parse tree into a structural label sequence and let the model automatically learn useful information through it. Specifically, we have described three different models to capture the syntax knowledge, i.e., *Parallel RNN*, *Hierarchical RNN*, and *Mixed RNN*. Experimentation on Chinese-to-English translation shows that all proposed models yield improvements over a state-of-the-art baseline NMT system. It is also interesting to note that the simplest model (i.e., *Mixed RNN*) achieves the best performance, resulting in obtaining significant improvements of 1.4 BLEU points on NIST MT 02 to 05.

In this paper, we have also analyzed the translation behavior of our improved system against the state-of-the-art NMT baseline system from several perspectives. Our analysis shows that there is still much room for NMT translation to be consistent with source syntax. In our future work, we expect several developments that will shed more light on utilizing source syntax, e.g., designing novel syntactic features (e.g., features showing the syntactic role that a word is playing) for NMT, and employing the source syntax to constrain and guild the attention models.

Acknowledgments

The authors would like to thank three anonymous reviewers for providing helpful comments, and also acknowledge Xing Wang, Xiangyu Duan, Zhengxian Gong for useful discussions. This work was supported by National Natural Science Foundation of China (Grant No. 61525205, 61331011, 61401295).

References

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

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics* 33(2):201–228.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machinetranslation: Encoder-decoder approaches. In *Proceedings of SSST 2014*. pages 103–111.

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

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of EMNLP 2016*. pages 2331–2336.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL 2010 System Demonstrations*. pages 7–12.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of ACL 2016*. pages 823–833.

Mercedes García-Martínez, Loic Barrault, and Fethi Bougares. 2016. Factored neural machine translation. In *arXiv:1609.04621*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by

preventing co-adaptation of feature detectors. In *arXiv:1207.0580*.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for wmt'15. In *Proceedings of WMT 2015*. pages 134–140.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*. pages 388–395.

Junhui Li, Philip Resnik, and Hal Daumé III. 2013. Modeling syntactic and semantic structures in hierarchical phrase-based translation. In *Proceedings of HLT-NAACL 2013*. pages 540–549.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of ACL-COLING 2006*. pages 609–616.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Proceedings of AAAI 2015*. pages 857–868.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of IWSLT 2015*. pages 76–79.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP 2015*. pages 1412–1421.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-HLT 2008*. pages 1003–1011.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of EMNLP 2016*. pages 2283–2288.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*. pages 311–318.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL 2007*. pages 404–411.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*. pages 83–91.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL 2016*. pages 1715–1725.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-HLT 2008*. pages 577–585.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of EMNLP 2016*. pages 1526–1534.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017a. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics* 5:87–99.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017b. Neural machine translation with reconstruction. In *Proceedings of AAAI 2017*. pages 3097–3103.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL 2016*. pages 76–85.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of NIPS 2015*.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017. Neural machine translation advised by statistical machine translation. In *Proceedings of AAAI 2017*. pages 3330–3336.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *arXiv preprint arXiv:1609.08144*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering* 11(2):207–238.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. In *arXiv:1212.5701*.

Sequence-to-Dependency Neural Machine Translation

Shuangzhi Wu[†*], **Dongdong Zhang**[‡], **Nan Yang**[‡], **Mu Li**[‡], **Ming Zhou**[‡]
[†]Harbin Institute of Technology, Harbin, China
[‡]Microsoft Research
{v-shuawu, dozhang, nanya, muli, mingzhou}@microsoft.com

Abstract

Nowadays a typical Neural Machine Translation (NMT) model generates translations from left to right as a linear sequence, during which latent syntactic structures of the target sentences are not explicitly concerned. Inspired by the success of using syntactic knowledge of target language for improving statistical machine translation, in this paper we propose a novel Sequence-to-Dependency Neural Machine Translation (SD-NMT) method, in which the target word sequence and its corresponding dependency structure are jointly constructed and modeled, and this structure is used as context to facilitate word generations. Experimental results show that the proposed method significantly outperforms state-of-the-art baselines on Chinese-English and Japanese-English translation tasks.

1 Introduction

Recently, Neural Machine Translation (NMT) with the attention-based encoder-decoder framework (Bahdanau et al., 2015) has achieved significant improvements in translation quality of many language pairs (Bahdanau et al., 2015; Luong et al., 2015a; Tu et al., 2016; Wu et al., 2016). In a conventional NMT model, an encoder reads in source sentences of various lengths, and transforms them into sequences of intermediate hidden vector representations. After weighted by attention operations, combined hidden vectors are used by the decoder to generate translations. In most of cases, both encoder and decoder are implemented as recurrent neural networks (RNNs).

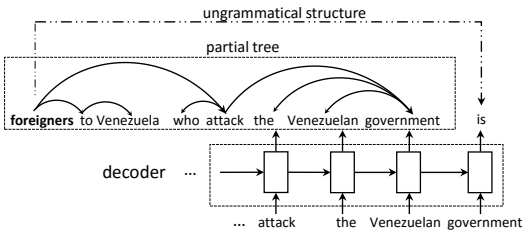
Many methods have been proposed to further improve the sequence-to-sequence NMT model since it was first proposed by Sutskever et al. (2014) and Bahdanau et al. (2015). Previous work ranges from addressing the problem of out-of-vocabulary words (Jean et al., 2015), designing attention mechanism (Luong et al., 2015a), to more efficient parameter learning (Shen et al., 2016), using source-side syntactic trees for better encoding (Eriguchi et al., 2016) and so on. All these NMT models employ a sequential recurrent neural network for target generations. Although in theory RNN is able to remember sufficiently long history, we still observe substantial incorrect translations which violate long-distance syntactic constraints. This suggests that it is still very challenging for a linear RNN to learn models that effectively capture many subtle long-range word dependencies. For example, Figure 1 shows an incorrect translation related to the long-distance dependency. The translation fragment in italic is locally fluent around the word *is*, but from a global view the translation is ungrammatical. Actually, this part of translation should be mostly affected by the distant plural noun *foreigners* rather than words *Venezuelan government* nearby.

Fortunately, such long-distance word correspondence can be well addressed and modeled by syntactic dependency trees. In Figure 1, the head word *foreigners* in the partial dependency tree (top dashed box) can provide correct structural context for the next target word, with this information it is more likely to generate the correct word *will* rather than *is*. This structure has been successfully applied to significantly improve the performance of statistical machine translation (Shen et al., 2008). On the NMT side, introducing target syntactic structures could help solve the problem of ungrammatical output because it can bring two advantages over state-of-the-art NMT models:

[*] Contribution during internship at Microsoft Research.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 698–707
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1065

a) syntactic trees can be used to model the grammatical validity of translation candidates; b) partial syntactic structures can be used as additional context to facilitate future target word prediction.



Source : 他 还 说 , 来 委 外国人 若 攻击 委内瑞拉 政府 会 面临 严重 后果 , 将 被 驱逐出境 .

Ref : He added that foreign visitors to Venezuela who criticize the Venezuelan government will face serious consequences and will be deported .

NMT : He also said that foreigners to Venezuela who attack the Venezuelan government *is facing serious consequences*, will be deported .

Figure 1: Dependency trees help the prediction of the next target word. "NMT" refers to the translation result from a conventional NMT model, which fails to capture the long distance word relation denoted by the dashed arrow.

However, it is not trivial to build and leverage syntactic structures on the target side in current NMT framework. Several practical challenges arise:

(1) How to model syntactic structures such as dependency parse trees with recurrent neural network;

(2) How to efficiently perform both target word generation and syntactic structure construction tasks simultaneously in a single neural network;

(3) How to effectively leverage target syntactic context to help target word generation.

To address these issues, we propose and empirically evaluate a novel Sequence-to-Dependency Neural Machine Translation (SD-NMT) model in our paper. An SD-NMT model encodes source inputs with bi-directional RNNs and associates them with target word prediction via attention mechanism as in most NMT models, but it comes with a new decoder which is able to jointly generate target translations and construct their syntactic dependency trees. The key difference from conventional NMT decoders is that we use two RNNs, one for translation generation and the other for dependency parse tree construction, in which incremental parsing is performed with the arc-standard shift-reduce algorithm proposed by Nivre (2004). We will describe in detail how these two RNNs work interactively in Section 3.

We evaluate our method on publicly available data sets with Chinese-English and Japanese-English translation tasks. Experimental results show that our model significantly improves translation accuracy over the conventional NMT and SMT baseline systems.

2 Background

2.1 Neural Machine Translation

As a new paradigm to machine translation, NMT is an end-to-end framework (Sutskever et al., 2014; Bahdanau et al., 2015) which directly models the conditional probability $P(Y|X)$ of target translation $Y = y_1,y_2,...,y_n$ given source sentence $X = x_1,x_2,...,x_m$. An NMT model consists of two parts: an encoder and a decoder. Both of them utilize recurrent neural networks which can be a Gated Recurrent Unit (GRU) (Cho et al., 2014) or a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) in practice. The encoder bidirectionally encodes a source sentence into a sequence of hidden vectors $H = h_1,h_2,...,h_m$ with a forward RNN and a backward RNN. Then the decoder predicts target words one by one with probability

$$P(Y|X) = \prod_{j=1}^{n} P(y_j|y_{<j}, H) \quad (1)$$

Typically, for the jth target word, the probability $P(y_j|y_{<j}, H)$ is computed as

$$P(y_j|y_{<j}, H) = g(s_j, y_{j-1}, c_j) \quad (2)$$

where g is a nonlinear function that outputs the probability of y_j, and s_j is the RNN hidden state. The context c_j is calculated at each timestamp j based on H by the attention network

$$c_j = \sum_{k=1}^{m} a_{jk} h_k \quad (3)$$

$$a_{jk} = \frac{\exp(e_{jk})}{\sum_{i=1}^{m} \exp(e_{ji})} \quad (4)$$

$$e_{jk} = v_a^T \tanh(W_a s_{j-1} + U_a h_k) \quad (5)$$

where v_a, W_a, U_a are the weight matrices. The attention mechanism is effective to model the correspondences between source and target.

2.2 Dependency Tree Construction

We use a shift-reduce transition-based dependency parser to build the syntactic structure for the target language in our work. Specially, we adopt the arc-standard algorithm (Nivre, 2004) to perform incremental parsing during the translation process. In this algorithm, a stack and a buffer are maintained to store the parsing state over which three kinds of transition actions are applied. Let w_0 and w_1 be two topmost words in the stack, and $\bar{w}$ be the current new word in a sequence of input, three transition actions are described as below.

- Shift(SH) : Push $\bar{w}$ to the stack.
- Left-Reduce(LR(d)) : Link w_0 and w_1 with dependency label d as $w_0 \xrightarrow{d} w_1$, and reduce them to the head w_0.
- Right-Reduce(RR(d)) : Link w_0 and w_1 with dependency label d as $w_0 \xleftarrow{d} w_1$, and reduce them to the head w_1.

During parsing, an specific structure is used to record the dependency relationship between different words of input sentence. The parsing finishes when the stack is empty and all input words are consumed. As each word must be pushed to the stack once and popped off once, the number of actions needed to parse a sentence is always $2n$, where n is the length of the sentence (Nivre, 2004). Because each valid transition action sequence corresponds to a unique dependency tree, a dependency tree can also be equivalently represented by a sequence of transition actions.

3 Sequence-to-Dependency Neural Machine Translation

An SD-NMT model is an extension to the conventional NMT model augmented with syntactic structural information of target translation. Given a source sentence $X = x_1,x_2,..,x_m$, its target translation $Y = y_1,y_2,..,y_n$ and Y's dependency parse tree T, the goal of the extension is to enable us to compute the joint probability $P(Y,T|X)$. As in most structural learning tasks, the full prediction of Y and T is further decomposed into a chain of smaller predictions. For translation Y, it is generated in the left-to-right order as $y_1, y_2, .., y_n$ following the way in a normal sequence-to-sequence model. For Y's parse tree T, instead of directly modeling the tree itself, we predict a parsing action sequence A which can map Y to T. Thus at top level our SD-NMT model can be formulated as

$$\begin{aligned} P(Y,T|X) &= P(Y,A|X) \\ &= P(y_1 y_2..y_n, a_1, a_2..a_l|X) \quad (6) \end{aligned}$$

where $A = a_1,a_2,..,a_j,..,a_l$ [1] with length l ($l = 2n$), $a_j \in \{\text{SH}, \text{RR}(d), \text{LR}(d)\}$[2].

Two recurrent neural networks, Word-RNN and Action-RNN, are used to model generation processes of translation sequence Y and parsing action sequence A respectively. Figure 2 shows an example how translation Y and its parsing actions are predicted step by step.

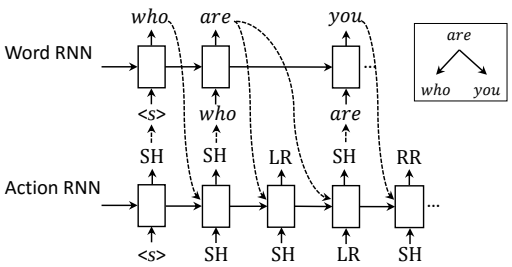


Figure 2: Decoding example of our SD-NMT model for target sentence "who are you" with transition action sequence "SH SH LR SH RR". The ending symbol `EOS` is omitted.

Because the lengths of Word-RNN and Action-RNN are different, they are designed to work in a mutually dependent way: a target word is only allowed to be generated when the SH action is predicted in the action sequence. In this way, we can perform incremental dependency parsing for translation Y and at the same time track the partial parsing status through the translation generation process.

For notational clarity, we introduce a virtual translation sequence $\hat{Y} = \hat{y}_1,\hat{y}_2,..,\hat{y}_j,..,\hat{y}_l$ for Word-RNN which has the same length l with transition action sequence. $\hat{y}_j$ is defined as

$$\hat{y}_j = \begin{cases} y_{v_j} & \delta(\text{SH}, a_j) = 1 \\ y_{v_j - 1} & \delta(\text{SH}, a_j) = 0 \end{cases}$$

where $\delta(\text{SH}, a_j)$ is 1 when $a_j = \text{SH}$, otherwise 0. v_j is the index of Y, computed by $v_j = \sum_{i=1}^{j} \delta(SH, a_i)$. Apparently the mapping from $\hat{Y}$

[1] In the rest of this paper, a_j represents the transition action, rather than the attention weight in Equation 4.

[2] RR(d) refers to a set of RR actions augmented with dependency labels so as to LR(d).

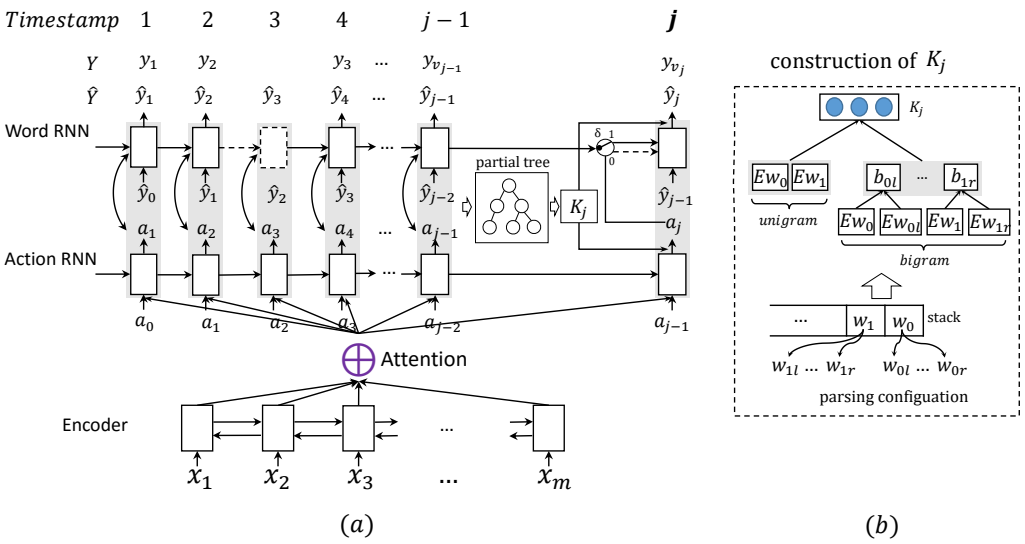


Figure 3: (a) is the overview of SD-NMT model. The dashed arrows mean copying previous recurrent state or word. The two RNNs use the same source context for prediction. $a_j \in \{\text{SH}, \text{RR}(d), \text{LR}(d)\}$. The bidirection arrow refers to the interaction between two RNNs. (b) shows the construction of syntactic context. The gray box means the concatenation of vectors

to Y is deterministic, and Y can be easily derived given $\hat{Y}$ and A.

With the notation of $\hat{Y}$, the sequence probability of Y and A can be written as

$$P(A|X, \hat{Y}_{<l}) = \prod_{j=1}^{l} P(a_j|a_{<j}, X, \hat{Y}_{<j}) \quad (7)$$

$$P(\hat{Y}|X, A_{\leq l}) = \prod_{j=1}^{l} P(\hat{y}_j|\hat{y}_{<j}, X, A_{\leq j})^{\delta(SH, a_j)} \quad (8)$$

where $\hat{Y}_{<j}$ refers to the subsequence $\hat{y}_1, \hat{y}_2, .., \hat{y}_{j-1}$, and $A_{\leq j}$ to $a_1, a_2, .., a_j$. Based on Equation 7 and 8, the overall joint model can be computed as

$$P(Y, T|X) = P(A|X, \hat{Y}_{<l}) \times P(\hat{Y}|X, A_{\leq l}) \quad (9)$$

As we have two RNNs in our model, the termination condition is also different from a conventional NMT model. In decoding, we maintain a stack to track the parsing configuration, and our model terminates once the Word-RNN predicts a special ending symbol `EOS` and all the words in the stack have been reduced.

Figure 3 (a) gives an overview of our SD-NMT model. Due to space limitation, the detailed interconnections between two RNNs are only illustrated at timestamp j. The encoder of our model follows standard bidirectional RNN configuration. At timestamp j during decoding, our model first predicts an action a_j by Action-RNN, then Word-RNN checks the condition gate δ according to a_j. If $a_j = \text{SH}$, the Word-RNN will generate a new state (solid arrow) and predict a new target word y_{v_j}, otherwise it just copies previous state (dashed arrow) to the current state. For example, at timestamp 3, $a_3 \neq \text{SH}$, the state of Word-RNN is copied from its previous one. Meanwhile, $\hat{y}_3 = y_2$ is used as the immediate proceeding word in translation history.

When computing attention scores, we extend Equation 5 by replacing the decoder hidden state with the concatenation of Word-RNN hidden state s and Action-RNN hidden state s' (gray boxes in Figure 3). The new attention score is then updated as

$$e_{jk} = v_a^T \tanh(W_a[s_{j-1}; s'_{j-1}] + U_a h_k) \quad (10)$$

3.1 Syntactic Context for Target Word Prediction

Syntax has been proven useful for sentence generation task (Dyer et al., 2016). We propose to leverage target syntax to help translation generation. In our model, the syntactic context K_j at timestamp j is defined as a vector which is computed by a feed-forward network based on current

parsing configuration of Action-RNN. Denote that w_0 and w_1 are two topmost words in the stack, w_{0l} and w_{1l} are their leftmost modifiers in the partial tree, w_{0r} and w_{1r} their rightmost modifiers respectively. We define two unigram features and four bigram features. The unigram features are w_0 and w_1 which are represented by the word embedding vectors. The bigram features are w_0w_{0l}, w_0w_{0r}, w_1w_{1l} and w_1w_{1r}. Each of them is computed by $b_{hc} = \tanh(W_bEw_h + U_bEw_{hc})$, $h \in \{0,1\}$, $c \in \{l,r\}$. These kinds of feature template have beeb proven effective in dependency parsing task (Zhang and Clark, 2008). Based on these features, the syntactic context vector K_j is computed as

$$K_j = \tanh(W_k[Ew_0; Ew_1] + U_k[b_{0l}; b_{0r}; b_{1l}; b_{1r}]) \quad (11)$$

where W_k, U_k, W_b, U_b are the weight matrices, E stands for the embedding matrix. Figure 2 (b) gives an overview of the construction of K_j. Note that zero vector is used for padding the words which are not available in the partial tree, so that all the K vectors have the same input size in computation.

Adding K_j to Equation 2, the probabilities of transition action and word in Equation 7 and 8 are then updated as

$$P(a_j|a_{<j}, X, \hat{Y}_{<j}) = g(s'_j, a_{j-1}, c_j, K_j) \quad (12)$$
$$P(\hat{y}_j|\hat{y}_{<j}, X, A_{\leq j}) = g(s_j, \hat{y}_{j-1}, c_j, K_j) \quad (13)$$

After each prediction step in Word-RNN and Action-RNN, the syntax context vector K will be updated accordingly. Note that K is not used to calculate the recurrent states s in this work.

3.2 Model Training and Decoding

For SD-NMT model, we use the sum of log-likelihoods of word sequence and action sequence as objective function for training algorithm, so that the joint probability of target translations and their parsing trees can be maximized:

$$J(\theta) = \sum_{(X,Y,A)\in D} \log\ P(A|X, \hat{Y}_{<l}) + \log\ P(\hat{Y}|X, A_{\leq l}) \quad (14)$$

We also use mini-batch for model training. As the target dependency trees are known in the bilingual corpus during training, we pre-compute the partial tree state and syntactic context at each time stamp for each training instance. Thus it is easy for the model to process multiple trees in one batch.

In the decoding process of an SD-NMT model, the score of each search path is the sum of log probabilities of target word sequence and transition action sequence normalized by the sequence length:

$$\text{score} = \frac{1}{l}\sum_{j=1}^{l} \log\ P(a_j|a_{<j}, X, \hat{Y}_{<j}) + \frac{1}{n}\sum_{j=1}^{l} \delta(SH, a_j) \log\ P(\hat{y}_j|\hat{y}_{<j}, X, A_{\leq j}) \quad (15)$$

where n is word sequence length and l is action sequence length.

4 Experiments

The experiments are conducted on the Chinese-English task as well as the Japanese-English translation tasks where the same data set from WAT 2016 ASPEC corpus (Nakazawa et al., 2016) [3] is used for a fair comparison with other work. In addition to evaluate translation performance, we also investigate the quality of dependency parsing as a by-product and the effect of parsing quality against translation quality.

4.1 Setup

In the Chinese-English task, the bilingual training data consists of a set of LDC datasets, [4] which has around 2M sentence pairs. We use NIST2003 as the development set, and the testsets contain NIST2005, NIST2006, NIST2008 and NIST2012. All English words are lowercased.

In the Japanese-English task, we use top 1M sentence pairs from ASPEC Japanese-English corpus. The development data contains 1,790 sentences, and the test data contains 1,812 sentences with single reference per source sentence.

To train SD-NMT model, the target dependency tree references are needed. As there is no golden annotation of parse trees over the target training data, we use pseudo parsing results as the target dependency references, which are got from an in-house developed arc-eager dependency parser based on work in (Zhang and Nivre, 2011).

[3] http://orchid.kuee.kyoto-u.ac.jp/ASPEC/

[4] LDC2003E14, LDC2005T10, LDC2005E83, LDC2006E26, LDC2006E34, LDC2006E85, LDC2006E92, LDC2003E07, LDC2002E18, LDC2005T06, LDC2003E07, LDC2004T07, LDC2004T08, LDC2005T06

Settings	NIST 2005	NIST 2006	NIST 2008	NIST 2012	Average
HPSMT	35.34	33.56	26.06	27.47	30.61
RNNsearch	38.07	38.95	31.61	28.95	34.39
SD-NMT\K	38.83	39.23	31.92	29.72	34.93
SD-NMT	**39.38**	**41.81**	**33.06**	**31.43**	36.42

Table 1: Evaluation results on Chinese-English translation task with BLEU% metric. The "Average" column is the averaged result of all test sets. The numbers in bold indicate statistically significant difference ($p < 0.05$) from baselines.

In the neural network training, the vocabulary size is limited to 30K high frequent words for both source and target languages. All low frequent words are normalized into a special token `unk` and post-processed by following the work in (Luong et al., 2015b). The size of word embedding and transition action embedding is set to 512. The dimensions of the hidden states for all RNNs are set to 1024. All model parameters are initialized randomly with Gaussian distribution (Glorot and Bengio, 2010) and trained on a NVIDIA Tesla K40 GPU. The stochastic gradient descent (SGD) algorithm is used to tune parameters with a learning rate of 1.0. The batch size is set to 96. In the update procedure, Adadelta (Zeiler, 2012) algorithm is used to automatically adapt the learning rate. The beam sizes for both word prediction and transition action prediction are set to 12 in decoding.

The baselines in our experiments are a phrasal system and a neural translation system, denoted by HPSMT and RNNsearch respectively. HPSMT is an in-house implementation of the hierarchical phrase-based model (Chiang, 2005), where a 4-gram language model is trained using the modified Kneser-Ney smoothing (Kneser and Ney, 1995) algorism over the English Gigaword corpus (LDC2009T13) plus the target data from the bilingual corpus. RNNsearch is an in-house implementation of the attention-based neural machine translation model (Bahdanau et al., 2015) using the same parameter settings as our SD-NMT model including word embedding size, hidden vector dimension, beam size, as well as the same mechanism for OOV word processing.

The evaluation results are reported with the case-insensitive IBM BLEU-4 (Papineni et al., 2002). A statistical significance test is performed using the bootstrap resampling method proposed by Koehn (2004) with a 95% confidence level. For Japanese-English task, we use the official evaluation procedure provided by WAT 2016.[5], where both BLEU and RIBES (Isozaki et al., 2010) are used for evaluation.

4.2 Evaluation on Chinese-English Translation

We evaluate our method on the Chinese-English translation task. The evaluation results over all NIST test sets against baselines are listed in Table 1. Generally, RNNsearch outperforms HPSMT by 3.78 BLEU points on average while SD-NMT surpasses RNNsearch 2.03 BLUE point gains on average, which shows that NMT models usually achieve better results than SMT models, and our proposed sequence-to-dependency NMT model performs much better than traditional sequence-to-sequence NMT model.

We also investigate the effect of syntactic knowledge context by excluding its computation in Equation 12 and 13. The alternative model is denoted by SD-NMT\K. According to Table 1, SD-NMT\K outperforms RNNsearch by 0.54 BLEU points but degrades SD-NMT by 1.49 BLEU points on average, which demonstrates that the long distance dependencies captured by the target syntactic knowledge context, such as leftmost/rightmost children together with their dependency relationships, really bring strong positive effects on the prediction of target words.

In addition to translation quality, we compare the perplexity (PPL) changes on the development set in terms of numbers of training mini-batches for RNNsearch and SD-NMT in Figure 4. We can see that the PPL of SD-NMT is initially higher than that of RNNsearch, but decreased to be lower over time. This is mainly because the quality of parse tree is too poor at the beginning which degrades translation quality and leads to higher PPL. After some training iterations, the SD-NMT

[5]http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/index.html

	BLEU	RIBES	System Description
SMT Hiero	18.72	0.6511	Moses' Hierarchical Phrase-based SMT
SMT Phrase	18.45	0.6451	Moses' Phrase-based SMT
SMT S2T	20.36	0.6782	Moses' String-to-Tree Syntax-based SMT
Cromieres (2016)(Single model)	22.86	-	Single-layer NMT model without ensemble
Cromieres (2016)(Self-ensemble)	24.71	0.7508	Self-ensemble of 2-layer NMT model
Cromieres (2016)(4-Ensemble)	26.22	0.7566	Ensemble of 4 single-layer NMT models
RNNsearch	23.50	0.7459	Single-layer NMT model
SD-NMT	25.93	0.7540	Single-layer SD-NMT model

Table 2: Evaluation results on Japanese-English translation task.

model learns reasonable inferences of parse trees which begins to help target word generation and leads to lower PPL.

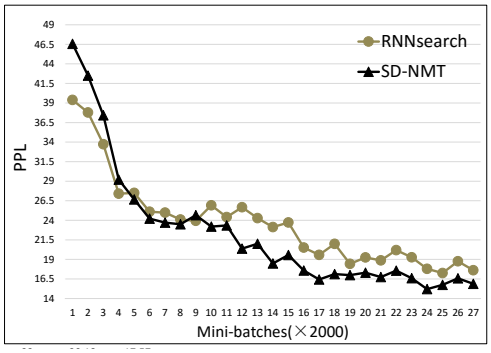


Figure 4: Perplexity (PPL) changes in terms of numbers of training mini-batches.

In our experiments, the time cost of SD-NMT is two times of that for RNNsearch due to a more complicated model structure. But we think it is a worthy trade to pursue high quality translations.

4.3 Evaluation on Japanese-English Translation

In this section, we report results on the Japanese-English translation task. To ensure fair comparisons, we use the same training data and follow the pre-processing steps recommended in WAT 2016[6]. Table 2 shows the comparison results from 8 systems with the evaluation metrics of BLEU and RIBES. The results in the first 3 rows are produced by SMT systems taken from the official WAT 2016. The remaining results are produced by NMT systems, among which the bottom two row results are taken from our in-house NMT systems and others refer to the work in (Cromieres, 2016; Cromieres et al., 2016) that are the competitive NMT results on WAT 2016. According to Table 2, NMT results still outperform SMT results similar to our Chinese-English evaluation results. The SD-NMT model significantly outperforms most other NMT models, which shows that our proposed approach to modeling target dependency tree benefit NMT systems since our RNNsearch baseline achieves comparable performance with the single layer attention-based NMT system in (Cromieres, 2016). Note that our SD-NMT gets comparable results with the 4 single-layer ensemble model in (Cromieres, 2016; Cromieres et al., 2016). We believe SD-NMT can get more improvements with an ensemble of multiple models in future experiments.

[6]http://lotus.kuee.kyoto-u.ac.jp/WAT/baseline/dataPreparationJE.html

4.4 Effect of the Parsing Accuracy upon Translation Quality

The interaction effect between dependency tree conduction and target word generation is investigated in this section. The experiments are conducted on the Chinese-English task over multiple test sets. We evaluate how the quality of dependency trees affect the performance of translation. In the decoding phase of SD-NMT, beam search is applied to the generations of both transition and actions as illustrated in Equation 15. Intuitively, the larger the beam size of action prediction is, the better the dependency tree quality is. We fix the beam size for generating target words to 12, and change the beam size for action prediction to see the difference. Figure 5 shows the evaluation results of all test sets. There is a tendency for BLEU scores to increase with the growth of action prediction beam size. The reason is that the translation quality increases as the quality of dependency tree improves, which shows the construction of dependency trees can boost the generation of target

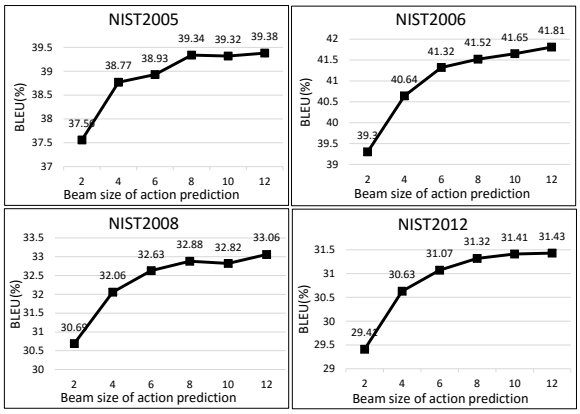


Figure 5: Translation performance against the beam size of action prediction.

words, and vice versa we believe.

4.5 Quality Estimation of Dependency Tree Construction

As a by-product, the quality of dependency trees not only affects the performance of target word generation, but also influences the possible downstream processors or tasks such as text analyses. The direct evaluation of tree quality is not feasible due to the unavailable golden references. So we resort to estimating the consistency between the by-products and the parsing results of our stand-alone dependency parser with state-of-the-art performance. The higher the consistency is, the closer the performance of by-product is to the stand-alone parser. To reduce the influence of ill-formed data as much as possible, we build the evaluation data set by heuristically selecting 360 SD-NMT translation results together with their dependency trees from NIST test sets where both source- and target-side do not contain `unk` and have a length of 20-30. We then take the parsing results of the stand-alone parser for these translations as references to indirectly estimate the quality of by-products. We get a UAS (unlabeled attachment score) of 94.96% and a LAS (labeled attachment score) of 93.92%, which demonstrates that the dependency trees produced by SD-NMT are much similar with the parsing results from the stand-alone parser.

4.6 Translation Example

In this section, we give a case study to explain how our method works. Figure 6 shows a translation example from the NIST testsets. SMT and RNNsearch refer to the translation results from the baselines HPSMT and NMT. For our SD-NMT model, we list both the generated translation and its corresponding dependency tree. We find that the translation of SMT is disfluent and ungrammatical, whereas RNNsearch is better than SMT. Although the translation of RNNsearch is locally fluent around word "have" in the rectangle, both its grammar is incorrect and its meaning is inaccurate from a global view. The word "have" should be in a singular form as its subject is "safety" rather than "workers". For our SD-NMT model, we can see that the translation is much better than baselines and the dependency tree is reasonable. The reason is that after generating the word "workers", the previous subtree in the gray region is transformed to the syntactic context which can guide the generation of the next word as illustrated by the dashed arrow. Thus our model is more likely to generate the correct verb "is" with singular form. In addition, the global structure helps the model correctly identify the inverted sentence pattern of the former translated part and make better choices for the future translation ("only when .. can .." in our translation, "only when .. will .." in the reference), which remains a challenge for conventional NMT model.

5 Related Work

Incorporating linguistic knowledge into machine translation has been extensively studied in Statistic Machine Translation (SMT) (Galley et al., 2006; Shen et al., 2008; Liu et al., 2006). Liu et al. (2006) proposed a tree-to-string alignment template for SMT to leverage source side syntactic information. Shen et al. (2008) proposed a target dependency language model for SMT to employ target-side structured information. These methods show promising improvement for SMT.

Recently, neural machine translation (NMT) has achieved better performance than SMT in many language pairs (Luong et al., 2015a; Zhang et al., 2016; Shen et al., 2016; Wu et al., 2016; Neubig, 2016). In a vanilla NMT model, source and target sentences are treated as sequences where the syntactic knowledge of both sides is neglected. Some effort has been done to incorporate source syntax into NMT. Eriguchi et al. (2016) proposed a tree-to-sequence attentional NMT model where source-side parse tree was used and achieved promising improvement. Intuitively, adding source syntactic information to

[**Source**] 只有 施工 人员 的 安全 得到 了 保证 , 才能 继续 施 工 .
[**Reference**] only when the safety of the workers is guaranteed will they continue with the project .
[**HPSMT**] only safety is assured of construction personnel , to continue construction .
[**RNNsearch**] only when the safety of construction workers *have* been guaranteed *to continue construction* .
[**SD-NMT**] only when the safety of the workers is guaranteed can we continue to work .

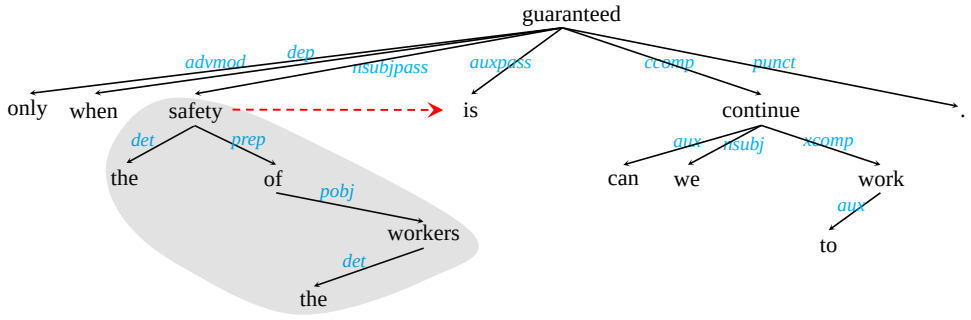


Figure 6: Translation examples of SMT, RNNsearch and our SD-NMT on Chinese-English translation task. The italic words on the arrows are dependency labels. The ending symbol EOS is omitted. RNNsearch fails to capture the long dependency which leads to an ungrammatical result. Whereas with the help of the syntactic tree, our SD-NMT can get a much better translation.

NMT is straightforward, because the source sentence is definitive and easy to attach extra information. However, it is non-trivial to add target syntax as target words are uncertain in decoding process. Up to now, there is few work that attempts to build and leverage target syntactic information for NMT.

There has been work that incorporates syntactic information into NLP tasks with neural networks. Dyer et al. (2016) presented a RNN grammar for parsing and language modeling. They replaced SH with a set of generative actions to generate words under a Stack LSTM framework (Dyer et al., 2015), which achieves promising results for language modeling on the Penn Treebank data. In our work, we propose to involve target syntactic trees into NMT model to jointly learn target translation and dependency parsing where target syntactic context over the parse tree is used to improve the translation quality.

6 Conclusion and Future Work

In this paper, we propose a novel string-to-dependency translation model over NMT. Our model jointly performs target word generation and arc-standard dependency parsing. Experimental results show that our method can boost the two procedures and achieve significant improvements on the translation quality of NMT systems.

In future work, along this research direction, we will try to integrate other prior knowledge, such as semantic information, into NMT systems. In addition, we will apply our method to other sequence-to-sequence tasks, such as text summarization, to verify the effectiveness.

Acknowledgments

We are grateful to the anonymous reviewers for their insightful comments. We also thank Shujie Liu and Zhirui Zhang for the helpful discussions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR 2015* .

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL 2005*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of ENMLP 2014*.

Fabien Cromieres. 2016. Kyoto-nmt: a neural machine translation implementation in chainer. In *Proceedings of COLING 2016*.

Fabien Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto university participation to wat 2016. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*. pages 166–174.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL 2015*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the NAACL 2016*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of ACL 2016*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of ACL 2006*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*. volume 9, pages 249–256.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8).

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of EMNLP*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL 2015*.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*. IEEE, volume 1, pages 181–184.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*. Citeseer, pages 388–395.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of ACL 2006*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP 2015*.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL 2015*.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Portoroz, Slovenia, pages 2204–2208.

Graham Neubig. 2016. Lexicons and minimum risk training for neural machine translation: NAIST-CMU at WAT2016. In *Proceedings of the 3nd Workshop on Asian Translation (WAT2016)*. Osaka, Japan.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*.

Libin Shen, Jinxi Xu, and Ralph M Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *ACL*. pages 577–585.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL 2016*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL 2016*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Biao Zhang, Deyi Xiong, jinsong su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of EMNLP 2016*.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *EMNLP2008*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL 2011*.

Detect Rumors in Microblog Posts Using Propagation Structure via Kernel Learning

Jing Ma[1], **Wei Gao**[2], **Kam-Fai Wong**[1,3]
[1]The Chinese University of Hong Kong, Hong Kong SAR
[2]Qatar Computing Research Institute, Doha, Qatar
[3]MoE Key Laboratory of High Confidence Software Technologies, China
[1]{majing,kfwong}@se.cuhk.edu.hk, [2]wgao@qf.org.qa

Abstract

How fake news goes viral via social media? How does its propagation pattern differ from real stories? In this paper, we attempt to address the problem of identifying rumors, i.e., fake information, out of microblog posts based on their propagation structure. We firstly model microblog posts diffusion with propagation trees, which provide valuable clues on how an original message is transmitted and developed over time. We then propose a kernel-based method called Propagation Tree Kernel, which captures high-order patterns differentiating different types of rumors by evaluating the similarities between their propagation tree structures. Experimental results on two real-world datasets demonstrate that the proposed kernel-based approach can detect rumors more quickly and accurately than state-of-the-art rumor detection models.

1 Introduction

On November 9th, 2016, Eric Tucker, a grassroots user who had just about 40 followers on Twitter, tweeted his unverified observations about paid protesters being bused to attend anti-Trump demonstration in Austin, Texas. The tweet, which was proved false later, was shared over 16 thousand times on Twitter and 350 thousand times on Facebook within a couple of days, fueling a nation-wide conspiracy theory[1]. The diffusion of the story is illustrated as Figure 1 which gives the key spreading points of the story along the time line. We can see that after the initial post, the tweet was shared or promoted by some influential online communities and users (including Trump himself), resulting in its wide spread.

A widely accepted definition of rumor is "unverified and instrumentally relevant information statements in circulation" (DiFonzo and Bordia, 2007). This unverified information may eventually turn out to be true, or partly or entirely false. In today's ever-connected world, rumors can arise and spread at lightening speed thanks to social media platforms, which could not only be wrong, but be misleading and dangerous to the public society. Therefore, it is crucial to track and debunk such rumors in timely manner.

Journalists and fact-checking websites such as snopes.com have made efforts to track and detect rumors. However, such endeavor is manual, thus prone to poor coverage and low speed. Feature-based methods (Castillo et al., 2011; Yang et al., 2012; Ma et al., 2015) achieved certain success by employing large feature sets crafted from message contents, user profiles and holistic statistics of diffusion patterns (e.g., number of retweets, propagation time, etc.). But such an approach was over simplified as they ignored the dynamics of rumor propagation. Existing studies considering propagation characteristics mainly focused on the temporal features (Kwon et al., 2013, 2017) rather than the structure of propagation.

So, can the propagation structure make any difference for differentiating rumors from non-rumors? Recent studies showed that rumor spreaders are persons who want to get attention and popularity (Sunstein, 2014). However, popular users who get more attention on Twitter (e.g., with more followers) are actually less likely to spread rumor in a sense that the high audience size might hinder a user from participating in propagating unverified information (Kwon et al., 2017). Intuitively, for "successful" rumors being propagated as widely

[1]https://www.nytimes.com/2016/11/20/business/media/how-fake-news-spreads.html

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 708–717
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1066

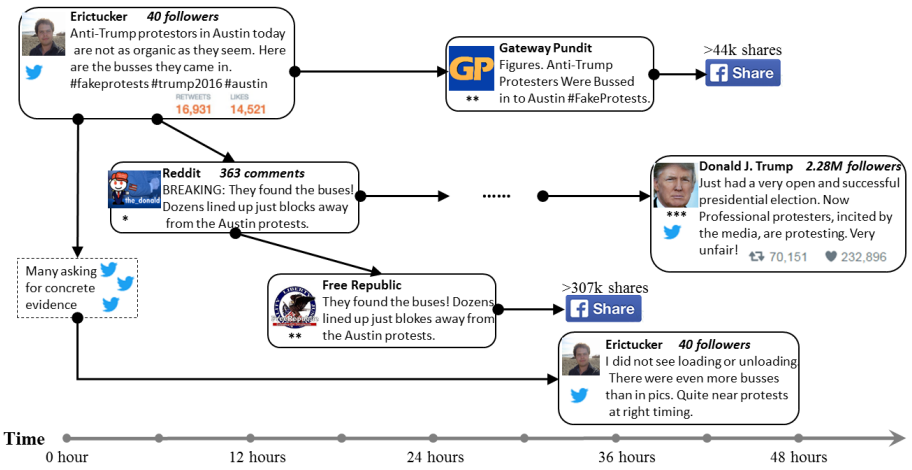
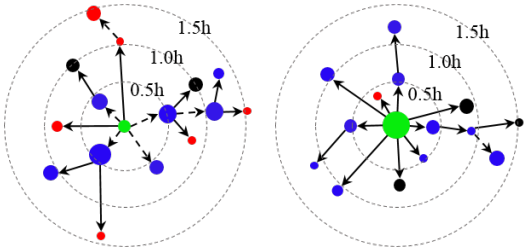


Figure 1: An illustration of how the rumor about "buses used to ship in paid anti-Trump protesters to Austin, Texas" becomes viral, where '*' indicates the level of influence.

as popular real news, initial spreaders (typically lack of popularity) must attract certain amount of broadcasting power, e.g., attention of influential users or communities that have a lot of audiences joining in promoting the propagation. We refer to this as a constrained mode propagation, relative to the open mode propagation of normal messages that everyone is open to share. Such different modes of propagation may imply some distinct propagation structures between rumors and non-rumors and even among different types of rumors.

Due to the complex nature of information diffusion, explicitly defining discriminant features based on propagation structure is difficult and biased. Figure 2 exemplifies the propagation structures of two Twitter posts, a rumor and a non-rumor, initiated by two users shown as the root nodes (in green color). The information flows here illustrate that the rumorous tweet is first posted by a low-impact user, then some popular users joining in who boost the spreading, but the non-rumorous tweet is initially posted by a popular user and directly spread by many general users; content-based signal like various users' stance (Zhao et al., 2015) and edge-based signal such as relative influence (Kwon et al., 2017) can also suggest the different nature of source tweets. Many of such implicit distinctions throughout message propagation are hard to hand craft specifically using flat summary of statistics as previous work did. In addition, unlike representation learning for plain text, learning for representation of structures such as networks is not well studied in general. Therefore, traditional and latest text-based models (Castillo et al., 2011; Ma et al., 2015, 2016) cannot be applied easily on such complex, dynamic structures.



(a) A rumor (b) A non-rumor

Figure 2: Fragments of the propagation for two source tweets. Node size: denotes the popularity of the user who tweet the post (represented by # of followers); Red, black, blue node: content-wise the user express doubt/denial, support, neutrality in the tweet, respectively; Solid (dotted) edge: information flow from a more (less) popular user to a less (more) popular user; Dashed concentric circles: time stamps.

To capture high-order propagation patterns for rumor detection, we firstly represent the propagation of each source tweet with a propagation tree which is formed by harvesting user's interactions to one another triggered by the source tweet. Then, we propose a kernel-based data-driven method called Propagation Tree Kernel (PTK) to generate relevant features (i.e., subtrees) automatically for estimating the similarity between two propagation trees. Unlike traditional tree kernel (Moschitti, 2006; Zhang et al., 2008) for modeling syntactic structure based on parse tree, our propagation tree consists of nodes corresponding to microblog

posts, each represented as a continuous vector, and edges representing the direction of propagation and providing the context to individual posts. The basic idea is to find and capture the salient substructures in the propagation trees indicative of rumors. We also extend PTK into a context-enriched PTK (cPTK) to enhance the model by considering different propagation paths from source tweet to the roots of subtrees, which capture the context of transmission. Extensive experiments on two real-world Twitter datasets show that the proposed methods outperform state-of-the-art rumor detection models with large margin.

Moreover, most existing approaches regard rumor detection as a binary classification problem, which predicts a candidate hypothesis as rumor or not. Since a rumor often begins as unverified and later turns out to be confirmed as true or false, or remains unverified (Zubiaga et al., 2016), here we consider a set of more practical, finer-grained classes: false rumor, true rumor, unverified rumor, and non-rumor, which becomes an even more challenging problem.

2 Related Work

Tracking misinformation or debunking rumors has been a hot research topic in multiple disciplines (DiFonzo and Bordia, 2007; Morris et al., 2012; Rosnow, 1991). Castillo et al. (2011) studied information credibility on Twitter using a wide range of hand-crafted features. Following that, various features corresponding to message contents, user profiles and statistics of propagation patterns were proposed in many studies (Yang et al., 2012; Wu et al., 2015; Sun et al., 2013; Liu et al., 2015). Zhao et al. (2015) focused on early rumor detection by using regular expressions for finding questing and denying tweets as the key for debunking rumor. All such approaches are over simplistic because they ignore the dynamic propagation patterns given the rich structures of social media data.

Some studies focus on finding temporal patterns for understanding rumor diffusion. Kown et al. (2013; 2017) introduced a time-series fitting model based on the temporal properties of tweet volume. Ma et al. (2015) extended the model using time series to capture the variation of features over time. Friggeri et al. (2014) and Hannak et al. (2014) studied the structure of misinformation cascades by analyzing comments linking to rumor debunking websites. More recently, Ma et al. (2016) used recurrent neural networks to learn the representations of rumor signals from tweet text at different times. Our work will consider temporal, structural and linguistic signals in a unified framework based on propagation tree kernel.

Most previous work formulated the task as classification at event level where an event is comprised of a number of source tweets, each being associated with a group of retweets and replies. Here we focus on classifying a given source tweet regarding a claim which is a finer-grained task. Similar setting was also considered in (Wu et al., 2015; Qazvinian et al., 2011).

Kernel methods are designed to evaluate the similarity between two objects, and tree kernel specifically addresses structured data which has been successfully applied for modeling syntactic information in many natural language tasks such as syntactic parsing (Collins and Duffy, 2001), question-answering (Moschitti, 2006), semantic analysis (Moschitti, 2004), relation extraction (Zhang et al., 2008) and machine translation (Sun et al., 2010). These kernels are not suitable for modeling the social media propagation structures because the nodes are not given as discrete values like part-of-speech tags, but are represented as high dimensional real-valued vectors. Our proposed method is a substantial extension of tree kernel for modeling such structures.

3 Representation of Tweets Propagation

On microblogging platforms, the follower/friend relationship embeds shared interests among the users. Once a user has posted a tweet, all his followers will receive the tweet. Furthermore, Twitter allows a user to retweet or comment another user's post, so that the information could reach beyond the network of the original creator.

We model the propagation of each source tweet as a tree structure $T(r) = \langle V, E \rangle$, where r is the source tweet as well as the root of the tree, V refers to a set of nodes each representing a responsive post (i.e., retweet or reply) of a user at a certain time to the source tweet r which initiates the circulation, and E is a set of directed edges corresponding to the response relation among the nodes in V. If there exists a directed edge from v_i to v_j, it means v_j is a direct response to v_i.

More specifically, each node $v \in V$ is represented as a tuple $v = (u_v, c_v, t_v)$, which provides

the following information: u_v is the creator of the post, c_v represents the text content of the post, and t_v is the time lag between the source tweet r and v. In our case, u_v contains attributes of the user such as # of followers/friends, verification status, # of history posts, etc., c_v is a vector of binary features based on uni-grams and/or bi-grams representing the post's content.

4 Propagation Tree Kernel Modeling

In this section, we describe our rumor detection model based on propagation trees using kernel method called Propagation Tree Kernel (PTK). Our task is, given a propagation tree $T(r)$ of a source tweet r, to predict the label of r.

4.1 Background of Tree Kernel

Before presenting our proposed algorithm, we briefly present the traditional tree kernel, which our PTK model is based on.

Tree kernel was designed to compute the syntactic and semantic similarity between two natural language sentences by implicitly counting the number of common subtrees between their corresponding parse trees. Given a syntactic parse tree, each node with its children is associated with a grammar production rule. Figure 3 illustrates the syntactic parse tree of "cut a tree" and its subtrees. A subtree is defined as any subgraph which has more than one nodes, with the restriction that entire (not partial) rule productions must be included. For example, the fragment [NP [D a]] is excluded because it contains only part of the production NP $\rightarrow$ D N (Collins and Duffy, 2001).

Following Collins and Duffy (2001), given two parse trees T_1 and T_2, the kernel function $K(T_1, T_2)$ is defined as:

$$\sum_{v_i \in V_1} \sum_{v_j \in V_2} \Delta(v_i, v_j) \quad (1)$$

where V_1 and V_2 are the sets of all nodes respectively in T_1 and T_2, and each node is associated with a production rule, and $\Delta(v_i, v_j)$ evaluates the common subtrees rooted at v_i and v_j. $\Delta(v_i, v_j)$ can be computed using the following recursive procedure (Collins and Duffy, 2001):

1) **if** the production rules at v_i and v_j are different, **then** $\Delta(v_i, v_j) = 0$;

2) **else if** the production rules at v_i and v_j are same, and v_i and v_j have only leaf children (i.e., they are pre-terminal symbols), **then** $\Delta(v_i, v_j) = \lambda$;

3) **else** $\Delta(v_i, v_j) = \lambda \prod_{k=1}^{min(nc(v_i), nc(v_j))} (1 + \Delta(ch(v_i, k), ch(v_j, k)))$.

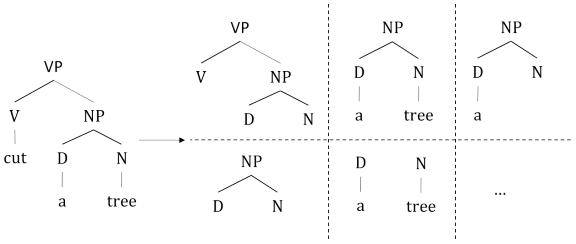


Figure 3: A syntactic parse tree and subtrees.

where $nc(v)$ is the number of children of node v, $ch(v, k)$ is the k-th child of node v, and λ ($0 < \lambda \leq 1$) is a decay factor. $\lambda = 1$ yields the number of common subtrees; $\lambda < 1$ down weighs the contribution of larger subtrees to make the kernel value less variable with respect to subtree size.

4.2 Our PTK Model

To classify propagation trees, we can calculate the similarity between the trees, which is supposed to reflect the distinction of different types of rumors and non-rumors based on structural, linguistic and temporal properties. However, existing tree kernels cannot be readily applied on propagation trees because 1) unlike parse tree where the node is represented by enumerable nominal value (e.g., part-of-speech tag), the propagation tree node is given as a vector of continuous numerical values representing the basic properties of the node; 2) the similarity of two parse trees is based on the count of common subtrees, for which the commonality of subtrees is evaluated by checking if the same production rules and the same children are associated with the nodes in two subtrees being compared, whereas in our context the similarity function should be defined softly since hardly two nodes from different propagation trees are same.

With the representation of propagation tree, we first define a function f to evaluate the similarity between two nodes v_i and v_j (we simplify the node representation for instance $v_i = (u_i, c_i, t_i)$) as the following:

$$f(v_i, v_j) = e^{-t} \left(\alpha \mathcal{E}(u_i, u_j) + (1 - \alpha) \mathcal{J}(c_i, c_j)\right)$$

where $t = |t_i - t_j|$ is the absolute difference between the time lags of v_i and v_j, $\mathcal{E}$ and $\mathcal{J}$ are

user-based similarity and content-based similarity, respectively, and α is the trade-off parameter. The intuition of using exponential function of t to scale down the similarity is to capture the discriminant signals or patterns at the different stages of propagation. For example, a questioning message posted very early may signal a false rumor while the same posted far later from initial post may indicate the rumor is still unverified, despite that the two messages are semantically similar.

The user-based similarity is defined as an Euclidean distance $\mathcal{E}(u_i, u_j) = ||u_i - u_j||_2$, where u_i and u_j are the user vectors of node v_i and v_j and $||\bullet||_2$ is the 2-norm of a vector. Here $\mathcal{E}$ is used to capture the characteristics of users participating in spreading rumors as discriminant signals, throughout the entire stage of propagation.

Contentwise, we use Jaccard coefficient to measure the similarity of post content:

$$\mathcal{J}(c_i, c_j) = \frac{|Ngram(c_i) \cap Ngram(c_j)|}{|Ngram(c_i) \cup Ngram(c_j)|}$$

where c_i and c_j are the sets of content words in two nodes. For n-grams here, we adopt both uni-grams and bi-grams. It can capture cue terms e.g., 'false', 'debunk', 'not true', etc. commonly occurring in rumors but not in non-rumors.

Given two propagation trees $T_1 = \langle V_1, E_1 \rangle$ and $T_2 = \langle V_2, E_2 \rangle$, PTK aims to compute the similarity between T_1 and T_2 iteratively based on enumerating all pairs of most similar subtrees. First, for each node $v_i \in V_1$, we obtain $v_i' \in V_2$, the most similar node of v_i from V_2:

$$v_i' = \arg\max_{v_j \in V_2} f(v_i, v_j)$$

Similarly, for each $v_j \in V_2$, we obtain $v_j' \in V_1$:

$$v_j' = \arg\max_{v_i \in V_1} f(v_i, v_j)$$

Then, the propagation tree kernel $K_P(T_1, T_2)$ is defined as:

$$\sum_{v_i \in V_1} \Lambda(v_i, v_i') + \sum_{v_j \in V_2} \Lambda(v_j', v_j) \qquad (2)$$

where $\Lambda(v, v')$ evaluates the similarity of two subtrees rooted at v and v', which is computed recursively as follows:

1) **if** v or v' are leaf nodes, **then** $\Lambda(v, v') = f(v, v')$;

2) **else** $\Lambda(v, v') = f(v, v') \prod_{k=1}^{min(nc(v), nc(v'))} (1 + \Lambda(ch(v, k), ch(v', k)))$

Note that unlike traditional tree kernel, in PTK the node similarity $f \in [0, 1]$ is used for softly counting similar subtrees instead of common subtrees. Also, λ in tree kernel is not needed as subtree size is not an issue here thanks to node similarity f.

PTK aims to capture discriminant patterns from propagation trees inclusive of user, content and temporal traits, which is inspired by prior analyses on rumors spreading, e.g., user information can be a strong clue in the initial broadcast, content features are important throughout entire propagation periods, and structural and temporal patterns help for longitudinal diffusion (Zubiaga et al., 2016; Kwon et al., 2017).

4.3 Context-Sensitive Extension of PTK

One defect of PTK is that it ignores the clues outside the subtrees, e.g., how the information propagates from source post to the current subtree. Intuitively, propagation paths provide further clue for determining the truthfulness of information since they embed the route and context of how the propagation happens. Therefore, we propose context-sensitive PTK (cPTK) by considering the propagation paths from the root of the tree to the roots of subtrees, which shares similar intuition with the context-sensitive tree kernel (Zhou et al., 2007).

For a propagation tree node $v \in T(r)$, let L_v^r be the length (i.e., # of nodes) of the propagation path from root r to v, and $v[x]$ be the x-th ancestor of v on the path starting from v ($0 \le x < L_v^r, v[0] = v, v[L_v^r - 1] = r$). cPTK evaluates the similarity between two trees $T_1(r_1)$ and $T_2(r_2)$ as follows:

$$\sum_{v_i \in V_1} \sum_{x=0}^{L_{v_i}^{r_1}-1} \Lambda_x(v_i, v_i') + \sum_{v_j \in V_2} \sum_{x=0}^{L_{v_j}^{r_2}-1} \Lambda_x(v_j', v_j) \qquad (3)$$

where $\Lambda_x(v, v')$ measures the similarity of subtrees rooted at $v[x]$ and $v'[x]$ for context-sensitive evaluation, which is computed as follows:

1) **if** $x > 0$, $\Lambda_x(v, v') = f(v[x], v'[x])$, where $v[x]$ and $v'[x]$ are the x-th ancestor nodes of v and v' on the respective propagation path.

2) **else** $\Lambda_x(v, v') = \Lambda(v, v')$, namely PTK.

Clearly, PTK is a special case of cPTK when $x = 0$ (see equation 3). cPTK evaluates the oc-

currence of both context-free (without considering ancestors on propagation paths) and context-sensitive cases.

4.4 Rumor Detection via Kernel Learning

The advantage of kernel-based method is that we can avoid painstakingly engineering the features. This is possible because the kernel function can explore an implicit feature space when calculating the similarity between two objects (Culotta and Sorensen, 2004).

We incorporate the proposed tree kernel functions, i.e., PTK (equation 2) or cPTK (equation 3), into a supervised learning framework, for which we utilize a kernel-based SVM classifier. We treat each tree as an instance, and its similarity values with all training instances as feature space. Therefore, the kernel matrix of training set is $m \times m$ and that of test set is $n \times m$ where m and n are the sizes of training and test sets, respectively.

For our multi-class task, we perform a one-vs-all classification for each label and then assign the one with the highest likelihood among the four, i.e., non-rumor, false rumor, true rumor or unverified rumor. We choose this method due to interpretability of results, similar to recent work on occupational class classification (Preotiuc-Pietro et al., 2015; Lukasik et al., 2015).

5 Experiments and Results

5.1 Data Sets

To our knowledge, there is no public large dataset available for classifying propagation trees, where we need a good number of source tweets, more accurately, the tree roots together with the corresponding propagation structure, to be appropriately annotated with ground truth.

We constructed our datasets based on a couple of reference datasets, namely Twitter15 (Liu et al., 2015) and Twitter16 (Ma et al., 2016). The original datasets were released and used for binary classification of rumor and non-rumor with respect to given events that contain their relevant tweets.

First, we extracted the popular source tweets[2] that are highly retweeted or replied. We then collected all the propagation threads (i.e., retweets and replies) for these source tweets. Because Twitter API cannot retrieve the retweets or replies, we gathered the retweet users for a given tweet from Twrench[3] and crawled the replies through Twitter's web interface.

Table 1: Statistics of the datasets

Statistic	Twitter15	Twitter16
# of users	276,663	173,487
# of source tweets	1,490	818
# of threads	331,612	204,820
# of non-rumors	374	205
# of false rumors	370	205
# of true rumors	372	205
# of unverified rumors	374	203
Avg. time length / tree	1,337 Hours	848 Hours
Avg. # of posts / tree	223	251
Max # of posts / tree	1,768	2,765
Min # of posts / tree	55	81

Finally, we annotated the source tweets by referring to the labels of the events they are from. We first turned the label of each event in Twitter15 and Twitter16 from binary to quaternary according to the veracity tag of the article in rumor debunking websites (e.g., snopes.com, Emergent.info, etc). Then we labeled the source tweets by following these rules: 1) Source tweets from unverified rumor events or non-rumor events are labeled the same as the corresponding event's label; 2) For a source tweet in false rumor event, we flip over the label and assign true to the source tweet if it expresses denial type of stance; otherwise, the label is assigned as false; 3) The analogous flip-over/no-change rule applies to the source tweets from true rumor events.

We make the datasets produced publicly accessible[4]. Table 1 gives statistics on the resulting datasets.

5.2 Experimental Setup

We compare our kernel-based method against the following baselines:

SVM-TS: A linear SVM classification model that uses time-series to model the variation of a set of hand-crafted features (Ma et al., 2015).

DTR: A Decision-Tree-based Ranking method to identify trending rumors (Zhao et al., 2015), which searches for enquiry phrases and clusters disputed factual claims, and ranked the clustered results based on statistical features.

DTC and **SVM-RBF:** The Twitter information credibility model using Decision Tree Classifier (Castillo et al., 2011) and the SVM-based

[2] Though unpopular tweets could be false, we ignore them as they do not draw much attention and are hardly impactful

[3] https://twren.ch

[4] https://www.dropbox.com/s/7ewzdrbelpmrnxu/rumdetect2017.zip?dl=0

model with RBF kernel (Yang et al., 2012), respectively, both using hand-crafted features based on the overall statistics of the posts.

RFC: The Random Forest Classifier proposed by Kwon et al. (2017) using three parameters to fit the temporal properties and an extensive set of hand-crafted features related to the user, linguistic and structure characteristics.

GRU: The RNN-based rumor detection model proposed by Ma et al. (2016) with gated recurrent unit for representation learning of high-level features from relevant posts over time.

BOW: A naive baseline we worked by representing the text in each tree using bag-of-words and building the rumor classifier with linear SVM.

Our models: **PTK** and **cPTK** are our full PTK and cPTK models, respectively; **PTK-** and **cPTK-** are the setting of only using content while ignoring user properties.

We implemented DTC and RFC with Weka[5], SVM models with LibSVM[6] and GRU with Theano[7]. We held out 10% of the trees in each dataset for model tuning, and for the rest of the trees, we performed 3-fold cross-validation. We used accuracy, F_1 measure as evaluation metrics.

5.3 Experimental Results

Table 2 shows that our proposed methods outperform all the baselines on both datasets.

Among all baselines, **GRU** performs the best, which learns the low-dimensional representation of responsive tweets by capturing the textual and temporal information. This indicates the effectiveness of complex signals indicative of rumors beyond cue words or phrases (e.g., "what?", "really?", "not sure", etc.). This also justifies the good performance of **BOW** even though it only uses uni-grams for representation. Although **DTR** uses a set of regular expressions, we found only 19.59% and 22.21% tweets in our datasets containing these expressions. That is why the results of **DTR** are not satisfactory.

SVM-TS and **RFC** are comparable because both of them utilize an extensive set of features especially focusing on temporal traits. But none of the models can directly incorporate structured propagation patterns for deep similarity comparison between propagation trees. **SVM-RBF**, although using a non-linear kernel, is based on traditional hand-crafted features instead of the structural kernel like ours. So, they performed obviously worse than our approach.

Table 2: Rumor detection results (NR: Non-Rumor; FR: False Rumor; TR: True Rumor; UR: Unverified Rumor)

(a) Twitter15 Dataset

Method	Acc.	NR F_1	FR F_1	TR F_1	UR F_1
DTR	0.409	0.501	0.311	0.364	0.473
SVM-RBF	0.318	0.455	0.037	0.218	0.225
DTC	0.454	0.733	0.355	0.317	0.415
SVM-TS	0.544	0.796	0.472	0.404	0.483
RFC	0.565	0.810	0.422	0.401	0.543
GRU	0.646	0.792	0.574	0.608	0.592
BOW	0.548	0.564	0.524	0.582	0.512
PTK-	0.657	0.734	0.624	0.673	0.612
cPTK-	0.697	0.760	0.645	0.696	0.689
PTK	0.710	**0.825**	0.685	0.688	0.647
cPTK	**0.750**	0.804	**0.698**	**0.765**	**0.733**

(b) Twitter16 Dataset

Method	Acc.	NR F_1	FR F_1	TR F_1	UR F_1
DTR	0.414	0.394	0.273	0.630	0.344
SVM-RBF	0.321	0.423	0.085	0.419	0.037
DTC	0.465	0.643	0.393	0.419	0.403
SVM-TS	0.574	0.755	0.420	0.571	0.526
RFC	0.585	0.752	0.415	0.547	0.563
GRU	0.633	0.772	0.489	0.686	0.593
BOW	0.585	0.553	0.556	0.655	0.578
PTK-	0.653	0.673	0.640	0.722	0.567
cPTK-	0.702	0.711	0.664	0.816	0.608
PTK	0.722	**0.784**	0.690	0.786	0.644
cPTK	**0.732**	0.740	**0.709**	**0.836**	**0.686**

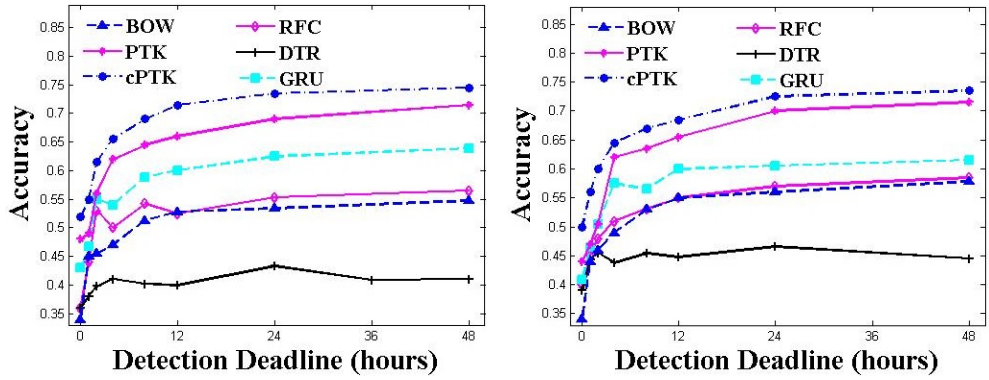
Representation learning methods like **GRU** cannot easily utilize complex structural information for learning important features from our networked data. In contrast, our models can capture complex propagation patterns from structured data rich of linguistic, user and temporal signals. Therefore, the superiority of our models is clear: **PTK-** which only uses text is already better than **GRU**, demonstrating the importance of propagation structures. **PTK** that combines text and user yields better results on both datasets, implying that both properties are complementary and **PTK** integrating flat and structured information is obviously more effective.

It is also observed that **cPTK** outperforms **PTK** except for non-rumor class. This suggests the context-sensitive modeling based on **PTK** is effective for different types of rumors, but for non-

[5] http://www.cs.waikato.ac.nz/ml/weka/

[6] https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[7] http://deeplearning.net/software/theano/



(a) Twitter15 Dataset (b) Twitter16 Dataset

Figure 4: Results of rumor early detection

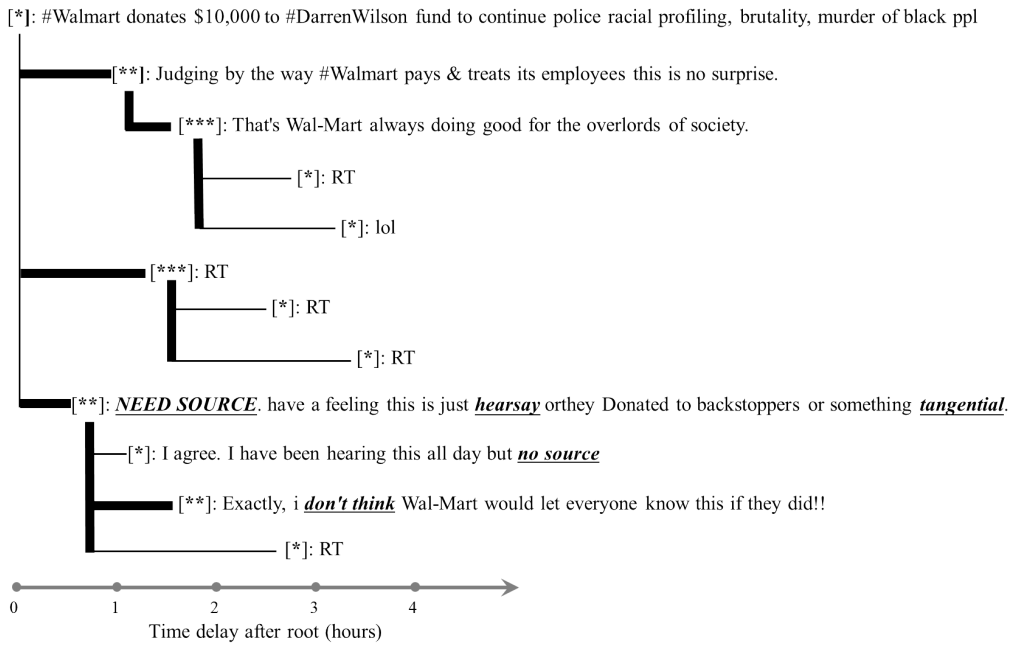


Figure 5: The example subtree of a rumor captured by the algorithm at early stage of propagation

rumors, it seems that considering context of propagation path is not always helpful. This might be due to the generally weak signals originated from node properties on the paths during non-rumor's diffusion since user distribution patterns in non-rumors do not seem as obvious as in rumors. This is not an issue in **cPTK-** since user information is not considered at all. Over all classes, **cPTK** achieves the highest accuracies on both datasets.

Furthermore, we observe that all the baseline methods perform much better on non-rumors than on rumors. This is because the features of existing methods were defined for a binary (rumor vs. non-rumor) classification problem. So, they do not perform well for finer-grained classes. Our approach can differentiate various classes much better by deep, detailed comparison of different patterns based on propagation structure.

5.4 Early Detection Performance

Detecting rumors at an early stage of propagation is very important so that preventive measures could be taken as quickly as possible. In early detection task, all the posts after a detection deadline are invisible during test. The earlier the deadline, the less propagation information can be available.

Figure 4 shows the performances of our **PTK** and **cPTK** models versus **RFC** (the best system based on feature engineering), **GRU** (the best system based on RNN) and **DTR** (an early-detection-

specific algorithm) against various deadlines. In the first few hours, our approach demonstrates superior early detection performance than other models. Particularly, **cPTK** achieve 75% accuracy on Twitter15 and 73% on Twitter16 within 24 hours, that is much faster than other models.

Our analysis shows that rumors typically demonstrate more complex propagation substructures especially at early stage. Figure 5 shows a detected subtree of a false rumor spread in its first few hours, where influential users are somehow captured to boost its propagation and the information flows among the users with an obvious unpopular-to-popular-to-unpopular trend in terms of user popularity, but such pattern was not witnessed in non-rumors in early stage. Many textual signals (underlined) can also be observed in that early period. Our method can learn such structures and patterns naturally, but it is difficult to know and hand-craft them in feature engineering.

6 Conclusion and Future Work

We propose a novel approach for detecting rumors in microblog posts based on kernel learning method using propagation trees. A propagation tree encodes the spread of a hypothesis (i.e., a source tweet) with complex structured patterns and flat information regarding content, user and time associated with the tree nodes. Enlightened by tree kernel techniques, our kernel method learns discriminant clues for identifying rumors of finer-grained levels by directly measuring the similarity among propagation trees via kernel functions. Experiments on two Twitter datasets show that our approach outperforms state-of-the-art baselines with large margin for both general and early rumor detection tasks.

Since kernel-based approach covers more structural information than feature-based methods, it allows kernel to further incorporate information from a high dimensional space for possibly better discrimination. In the future, we will focus on improving the rumor detection task by exploring network representation learning framework. Moreover, we plan to investigate unsupervised models considering massive unlabeled rumorous data from social media.

Acknowledgment

This work is partly supported by General Research Fund of Hong Kong (14232816). We would like to thank anonymous reviewers for the insightful comments.

References

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of WWW*.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Advances in neural information processing systems*. pages 625–632.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd annual meeting on association for computational linguistics*. Association for Computational Linguistics, page 423.

Nicholas DiFonzo and Prashant Bordia. 2007. Rumor, gossip and urban legends. *Diogenes* 54(1):19–35.

Adrien Friggeri, Lada A Adamic, Dean Eckles, and Justin Cheng. 2014. Rumor cascades. In *Proceedings of ICWSM*.

Aniko Hannak, Drew Margolin, Brian Keegan, and Ingmar Weber. 2014. Get back! you don't know me like that: The social mediation of fact checking interventions in twitter conversations. In *ICWSM*.

Sejeong Kwon, Meeyoung Cha, and Kyomin Jung. 2017. Rumor detection over varying time windows. *PLOS ONE* 12(1):e0168344.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *Proceedings of ICDM*.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *Proceedings of CIKM*.

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Classifying tweet level judgements of rumours in social media. *arXiv preprint arXiv:1506.00468* .

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of IJCAI*.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of CIKM*.

Meredith Ringel Morris, Scott Counts, Asta Roseway, Aaron Hoff, and Julia Schwarz. 2012. Tweeting is believing?: understanding microblog credibility perceptions. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, pages 441–450.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 335.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning*. Springer, pages 318–329.

Daniel Preotiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015. An analysis of the user occupational class through twitter content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*. Association for Computational Linguistics, pages 1754–1764.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1589–1599.

Ralph L Rosnow. 1991. Inside rumor: A personal journey. *American Psychologist* 46(5):484.

Jun Sun, Min Zhang, and Chew Lim Tan. 2010. Exploring syntactic structural features for sub-tree alignment using bilingual tree kernels. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 306–315.

Shengyun Sun, Hongyan Liu, Jun He, and Xiaoyong Du. 2013. Detecting event rumors on sina weibo automatically. In *Web Technologies and Applications*, Springer, pages 120–131.

Cass R Sunstein. 2014. *On rumors: How falsehoods spread, why we believe them, and what can be done*. Princeton University Press.

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st International Conference on Data Engineering (ICDE)*. IEEE, pages 651–662.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*.

Min Zhang, GuoDong Zhou, and Aiti Aw. 2008. Exploring syntactic structured features over parse trees for relation extraction using kernel methods. *Information processing & management* 44(2):687–701.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of WWW*.

GuoDong Zhou, Min Zhang, Dong Hong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. *EMNLP-CoNLL 2007* page 728.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLOS ONE* 11(3):e0150989.

EmoNet: Fine-Grained Emotion Detection with Gated Recurrent Neural Networks

Muhammad Abdul-Mageed
School of Library, Archival & Information Studies
University of British Columbia
muhammad.mageed@ubc.ca

Lyle Ungar
Computer and Information Science
University of Pennsylvania
ungar@cis.upenn.edu

Abstract

Accurate detection of emotion from natural language has applications ranging from building emotional chatbots to better understanding individuals and their lives. However, progress on emotion detection has been hampered by the absence of large labeled datasets. In this work, we build a very large dataset for fine-grained emotions and develop deep learning models on it. We achieve a new state-of-the-art on 24 fine-grained types of emotions (with an average accuracy of 87.58%). We also extend the task beyond emotion types to model Robert Plutchik's 8 primary emotion dimensions, acquiring a superior accuracy of 95.68%.

1 Introduction

According to the *Oxford English Dictionary*, emotion is defined as "[a] strong feeling deriving from one's circumstances, mood, or relationships with others." [1] This "standard" definition identifies emotions as constructs involving something innate that is often invoked in social interactions and that aids in communicating with others(Hwang and Matsumoto, 2016). It is no exaggeration that humans are emotional beings: Emotions are an integral part of human life, and affect our decision making as well as our mental and physical health. As such, developing emotion detection models is important; they have a wide array of applications, ranging from building nuanced virtual assistants that cater for the emotions of their users to detecting the emotions of social media users in order to understand their mental and/or physical health. However, emotion detection has remained a challenging task, partly due to the limited availability of labeled data and partly due the controversial nature of what emotions themselves are (Aaron C. Weidman and Tracy, 2017).

Recent advances in machine learning for natural language processing (NLP) suggest that, given enough labeled data, there should be an opportunity to build better emotion detection models. Manual labeling of data, however, is costly and so it is desirable to develop labeled emotion data without annotators. While the proliferation of social media has made it possible for us to acquire large datasets with implicit labels in the form of hashtags (Mohammad and Kiritchenko, 2015), such labels are noisy and reliable.

In this work, we seek to enable deep learning by creating a large dataset of fine-grained emotions using Twitter data. More specifically, we harness cues in Twitter data in the form of emotion hashtags as a way to build a labeled emotion dataset that we then exploit using *distant supervision* (Mintz et al., 2009) (the use of hashtags as a surrogate for annotator-generated emotion labels) to build emotion models grounded in psychology. We construct such a dataset and exploit it using powerful deep learning methods to build accurate, high coverage models for emotion prediction. Overall, we make the following contributions: 1) Grounded in psychological theory of emotions, we build a large-scale, high quality dataset of tweets labeled with emotions. Key to this are methods to ensure data quality, 2) we validate the data collection method using human annotations, 3) we develop powerful deep learning models using a gated recurrent network to exploit the data, yielding new state-of-the-art on 24 fine-grained types of emotions, and 4) we extend the task beyond these emotion types to model Plutick's 8 primary emotion dimensions.

[1] https://en.oxforddictionaries.com/definition/emotion.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 718–728
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1067

Our emotion modeling relies on *distant supervision* (Read, 2005; Mintz et al., 2009), the approach of using cues in data (e.g., hashtags or emoticons) as a proxy for "ground truth" labels as we explained above. Distant supervision has been investigated by a number of researchers for emotion detection (Tanaka et al., 2005; Mohammad, 2012; Purver and Battersby, 2012; Wang et al., 2012; Pak and Paroubek, 2010; Yang et al., 2007) and for other semantic tasks such as sentiment analysis (Read, 2005; Go et al., 2009) and sarcasm detection (González-Ibánez et al., 2011). In these works, authors successfully use emoticons and/or hashtags as marks to label data after performing varying degrees of data quality assurance. We take a similar approach, using a larger collection of tweets, richer emotion definitions, and stronger filtering for tweet quality.

The remainder of the paper is organized as follows: We first overview related literature in Section 2, describe our data collection in Section 3.1, and the annotation study we performed to validate our distant supervision method in Section 4. We then describe our methods in Section 5, provide results in Section 6, and conclude in Section 8.

2 Related Work

2.1 Computational Treatment of Emotion

The SemEval-2007 Affective Text task (Strapparava and Mihalcea, 2007) [SEM07] focused on classification of emotion and valence (i.e., positive and negative texts) in news headlines. A total of 1,250 headlines were manually labeled with the 6 basic emotions of Ekman (Ekman, 1972) and made available to participants. Similarly, (Aman and Szpakowicz, 2007) describe an emotion annotation task of identifying emotion category, emotion intensity and the words/phrases that indicate emotion in blog post data of 4,090 sentences and a system exploiting the data. Our work differs from both that of SEM07 (Strapparava and Mihalcea, 2007) and (Aman and Szpakowicz, 2007) in that we focus on a different genre (i.e., Twitter) and investigate distant supervision as a way to acquire a significantly larger labeled dataset.

Our work is similar to (Mohammad, 2012; Mohammad and Kiritchenko, 2015), (Wang et al., 2012), and (Volkova and Bachrach, 2016) who use distant supervision to acquire Twitter data with emotion hashtags and report analyses and experiments to validate the utility of this approach. For example, (Mohammad, 2012) shows that by using a simple domain adaptation method to train a classifier on their data they are able to improve both precision and recall on the SemEval-2007 (Strapparava and Mihalcea, 2007) dataset. As the author points out, this is another premise that the self-labeled hashtags acquired from Twitter are consistent, to some degree, with the emotion labels given by the trained human judges who labeled the SemEval-2007 data. As pointed out earlier, (Wang et al., 2012) randomly sample a set of 400 tweets from their data and human-label as relevant/irrelevant, as a way to verify the distant supervision approach with the quality assurance heuristics they employ. The authors found that the precision on a test set is 93.16%, thus confirming the utility of the heuristics. (Wang et al., 2012) provide a number of important observations, as conclusions based on their work. These include that since they are provided by the tweets' writers, the emotion hashtags are more natural and reliable than the emotion labels traditionally assigned by annotators to data by a few annotators. This is the case since in the lab-condition method annotators need to infer the writers emotions from text, which may not be accurate. Additionally, (Volkova and Bachrach, 2016) follow the same distant supervision approach and find correlations of users' emotional tone and the perceived demographics of these users' social networks exploiting the emotion hashtag-labeled data. Our dataset is more than an order of magnitude larger than (Mohammad, 2012) and (Volkova and Bachrach, 2016) and the range of emotions we target is much more fine grained than (Mohammad, 2012; Wang et al., 2012; Volkova and Bachrach, 2016) since we model 24 emotion types, rather than focus on ≤ 7 basic emotions.

(Yan et al., 2016; Yan and Turtle, 2016a,b) develop a dataset of 15,553 tweets labeled with 28 emotion types and so target a fine-grained range as we do. The authors instruct human annotators under lab conditions to assign any emotion they feel is expressed in the data, allowing them to assign more than one emotion to a given tweet. A set of 28 chosen emotions was then decided upon and further annotations were performed using Amazon Mechanical Turk (AMT). The authors cite an agreement of 0.50 Krippendorff's alpha (α) between the lab/expert annotators, and an (α) of 0.28 between experts and AMT workers. EmoTweet-

28 is a useful resource. However, the agreement between annotators is not high and the set of assigned labels do not adhere to a specific theory of emotion. We use a much larger dataset and report an accuracy of the hashtag approach at 90% based on human judgement as reported in Section 4.

2.2 Mood

A number of studies have also been performed to analyze and/or model mood in social media data. (De Choudhury et al., 2012) identify more than 200 moods frequent on Twitter as extracted from psychological literature and filtered by AMT workers. They then collect tweets which have one of the moods in their mood lexicon in the form of a hashtag. To verify the quality of the mood data, the authors run AMT studies where they ask workers whether a tweet displayed the respective mood hashtag or not and find that in 83% of the cases hashtagged moods at the end of posts did capture users' moods, whereas for posts with mood hashtags anywhere in the tweet, only 58% of the cases capture the mood of users. Although they did not build models for mood detection, the annotation studies (De Choudhury et al., 2012) perform further support our specific use of hashtags to label emotions. (Mishne and De Rijke, 2006) collect user-labeled mood from blog post text on LiveJournal and exploit them for predicting the intensity of moods over a time span rather than at the post level. Similarly, (Nguyen, 2010) builds models to infer patterns of moods in a large collection of LiveJournal posts. Some of the moods in these LiveJournal studies (e.g., *hungry*, *cold*), as (De Choudhury et al., 2012) explain, would not fit any psychological theory. Our work is different in that it is situated in psychological theory of emotion.

2.3 Deep Learning for NLP

In spite of the effectiveness of feature engineering for NLP, it is a labor intensive task that also needs domain expertise. More importantly, feature engineering falls short of extracting and organizing all the discriminative information from data (LeCun et al., 2015; Goodfellow et al., 2016). Neural networks (Goodfellow et al., 2016) have emerged as a successful class of methods that has the power of automatically discovering the representations needed for detection or classification and has been successfully applied to multiple NLP tasks. A line of studies in the literature (e.g., (Labutov and Lipson, 2013; Maas et al., 2011; Tang et al., 2014b,a) aim to learn sentiment-specific word embeddings (Bengio et al., 2003; Mikolov et al., 2013) from neighboring text. Another thread of research focuses on learning semantic composition (Mitchell and Lapata, 2010), including extensions to phrases and sentences with recursive neural networks (a class of syntax-tree models) (Socher et al., 2013; Irsoy and Cardie, 2014; Li et al., 2015) and to documents with distributed representations of sentences and paragraphs (Le and Mikolov, 2014; Tang et al., 2015) for modeling sentiment.

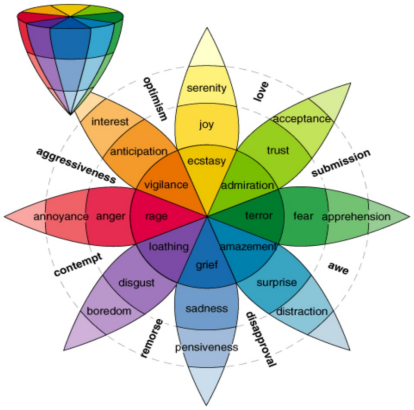
Long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Neural Nets (GRNNs) (Cho et al., 2014; Chung et al., 2015), variations of recurrent neural networks (RNNs), a type of networks suitable for handling time-series data like speech (Graves et al., 2013) or handwriting recognition (Graves, 2012; Graves and Schmidhuber, 2009), have also been used successfully for sentiment analysis (Ren et al., 2016; Liu et al., 2015; Tai et al., 2015; Tang et al., 2015; Zhang et al., 2016). Convolutional neural networks (CNNs) have also been quite successful in NLP, and have been applied to a range of sentence classification tasks, including sentiment analysis (Blunsom et al., 2014; Kim, 2014; Zhang et al., 2015). Other architectures have also been recently proposed (e.g., (Bradbury et al., 2016)). A review of neural network methods for NLP can be found in (Goldberg, 2016).

3 Data

3.1 Collection of a Large-Scale Dataset

To be able to use deep learning for modeling emotion, we needed a large dataset of labeled tweets. Since there is no such human-labeled dataset publicly available, we follow (Mohammad, 2012; Mintz et al., 2009; Purver and Battersby, 2012; González-Ibánez et al., 2011; Wang et al., 2012) in adopting *distant supervision*: We collect tweets with emotion-carrying hashtags as a surrogate for emotion labels. To be able to collect enough tweets to serve our need, we developed a list of hashtags representing each of the 24 emotions proposed by Robert Plutchick (Plutchik, 1980, 1985, 1994). Plutchik (Plutchik, 2001) organizes emotions in a three-dimensional circumplex model analogous to the colors on a color wheel. The cone's vertical dimension represents intensity, and the 3 circle represent degrees of similarity

Figure 1: Plutchik's wheel of emotion.



among the various emotion types. The eight sectors are meant to capture that there are eight primary emotion dimensions arranged as four pairs of opposites. Emotions in the blank spaces are the primary emotion dyads (i.e., emotions that are mixtures of two of the primary emotions). For this work, we exclude the dyads in the exploded model from our treatment. For simplicity, we refer to the circles as `plutchik-1`: with the emotions {*admiration, amazement, ecstasy, grief, loathing, rage, terror, vigilance*}, `plutchik-2`: with the emotions {*joy, trust, fear, surprise, sadness, disgust, anger, anticipation*}, and `plutchik-3`: with the emotions {*acceptance, annoyance, apprehension, boredom, distraction, interest, pensiveness, serenity*}. The wheel is shown in Figure 1.

For each emotion type, we prepared a seed set of hashtags representing the emotion. We used Google synonyms and other online dictionaries and thesauri (e.g., `www.thesaurus.com`) to expand the initial seed set of each emotion. We acquire a total of 665 emotion hashtags across the 24 emotion types. For example, for the *joy* emotion, a subset of the seeds in our expanded set is {*"happy", "happiness", "joy", "joyful", "joyfully", "delighted", "feelingsunny", "blithe", "beatific", "exhilarated", "blissful", "walkingonair", "jubilant"*}. We then used the expanded set to extract tweets with hashtags from the set from a number of massive-scale in-house Twitter datasets. We also used Twitter API to crawl Twitter with hashtags from the expanded set. Using this method, we were able to acquire a dataset of about $1/4$ billion tweets covering an extended time span from July 2009 till January 2017.

3.2 Preprocessing and Quality Assurance

Twitter data are very noisy, not only because of use of non-standard typography (which is less of a problem here) but due to the many duplicate tweets and the fact that tweets often have multiple emotion hashtags. Since these reduce our ability to build accurate models, we need to clean the data and remove duplicates. Starting with $>$ 1/4 billion tweets, we employ a rigorous and strict pipeline. This results in a vastly smaller set of about 1.6 million dependable labeled tweets.

Since our goal is to create non-overlapping categories at the level of a tweet, we first removed all tweets with hashtags belonging to more than one emotion of the 24 emotion categories. Since it was observed (e.g., (Mohammad, 2012; Wang et al., 2012)) and also confirmed by our annotation study as described in Section 4, that hashtags in tweets with URLs are less likely to correlate with a true emotion label, we remove all tweets with URLs from our data. We filter out duplicates using a two-step procedure: 1) we remove all retweets (based on existence of the token "RT" regardless of case) and 2) we use the Python library *pandas* `http://pandas.pydata.org/` "drop_duplicates" method to compare the tweet texts of all the tweets after normalizing character repetitions [all consecutive characters of $>$ 2 to 2] and user mentions (as detected by a string starting with an "@" sign). We then performed a manual inspection of a random sample of 1,000 tweets from the data and found no evidence of any remaining tweet duplicates.

Next, even though the emotion hashtags themselves are exclusively in English, we observe the data do have tweets in languages other than English. This is due to code-switching, but also to the fact that our data dates back to 2009 and Twitter did not allow use of hashtags for several non-English languages until 2012. To filter out non-English, we use the langid (Lui and Baldwin, 2012) (`https://github.com/saffsd/langid.py`) library to assign language tags to the tweets. Since the common wisdom in the literature (e.g., (Mohammad, 2012; Wang et al., 2012)) is to restrict data to hashtags occurring in final position of a tweet, we investigate correlations between a tweet's relevance and emotion hashtag location in Section 4 and test models exclusively on data with hashtags occurring in final position. We also only use tweets con-

taining at least 5 words.

Table 2 shows statistics of the data after applying our cleaning, filtering, language identification, and deduplication pipeline. Since our focus is on English, we only show statistics for tweets tagged with an "en" (for "English") label by langid. Table 2 provides three types of relevant statistics: 1) counts of all tweets, 2) counts of tweets with at least 5 words and the emotion hashtags occurring in the *last quarter* of the tweet text (based on character count), and 3) counts of tweets with at least 5 words and the emotion hashtags occurring as the *final* word in the tweet text. As the last column in Table 2 shows, employing our most strict criterion where an emotion hashtag must occur finally in a tweet of a minimal length 5 words, we acquire a total of 1,608,233 tweets: 205,125 tweets for `plutchik-1`, 790,059 for `plutchik-2`, and 613,049 for `plutchik-3`. [2]

Emotion	ct	ct@lq	ct@end
admiration	292,153	150,509	112,694
amazement	568,255	358,472	34,826
ecstasy	54,174	34,307	23,856
grief	102,980	33,141	12,568
loathing	90,465	41,787	456
rage	30,994	11,777	4,749
terror	84,827	25,908	15,268
vigilance	6,171	1,028	708
plutchik-1	**1,230,019**	**656,929**	**205,125**
anger	131,082	82,447	56,472
anticipation	67,175	36,846	26,655
disgust	212,770	145,052	52,067
fear	302,989	153,513	98,657
joy	974,226	522,689	330,738
sadness	1,252,192	762,901	142,300
surprise	143,755	78,570	53,915
trust	198,619	103,332	29,255
plutchik-2	**3,282,808**	**1,885,350**	**790,059**
acceptance	138,899	54,706	16,522
annoyance	954,027	791,869	364,135
apprehension	29,174	11,650	7,828
boredom	872,246	583,994	152,105
distraction	122,009	52,633	617
interest	113,555	67,216	56,659
pensiveness	11,751	5,012	3,513
serenity	97,467	36,817	11,670
plutchik-3	**2,339,128**	**1,603,897**	**613,049**
ALL	**6,851,955**	**4,146,176**	**1,608,233**

Table 2: Data statistics.

4 Annotation Study

In their work, (Wang et al., 2012) manually label a random sample of 400 tweets extracted with hashtags in a similar way as we acquire our data and find that human annotators agree 93% of the time with the hashtag emotion type if the hashtag occurs as the last word in the tweet. We wanted to validate our use of hashtags in a similar fashion and on a bigger random sample. We had human annotators label a random sample of 5,600 tweets that satisfy our preprocessing pipeline. Manual inspection during annotation resulted in further removing a negligible 16 tweets that were found to have problems. For each of the remaining 5,584 tweets, the annotators assign a binary tag from the set {*relevant, irrelevant*} to indicate whether a tweet carries an emotion category as assigned using our distant supervision method or not. Annotators assigned 61.37% ($n = 3,427$) "relevant" tags and 38.63% ($n = 2,157$) "irrelevant" tags. Our analysis of this manually labeled dataset also supports the findings of (Wang et al., 2012): When we limit position of the emotion hashtag to the end of a tweet, we acquire 90.57% relevant data. We also find that if we relax the constraint on the hashtag position such that we allow the hashtag to occur in the last quarter of a tweet (based on a total tweet character count), we acquire 85.43% relevant tweets. We also find that only 23.20% ($n = 795$ out of $3,427$) of the emotion carrying tweets have the emotion hashtags occurring in final position, whereas 31.75% ($n = 1,088$ out of $3,427$) of the tweets have the emotion hashtags in the last quarter of the tweet string. This shows how enforcing a final hashtag location results in loss of a considerable number of emotion tweets. As shown in Table 2, only $1,608,233$ tweets out of a total of $6,851,955$ tweets ($\% = 23,47$) in our bigger dataset have emotion hashtags occurring in final position. Overall, we agree with (Mohammad, 2012; Wang et al., 2012) that the accuracy acquired by enforcing a strict pipeline and limiting to emotion hashtags to final position is a reasonable measure for warranting good-quality data for training supervised systems, an assumption we have also validated with our empirical findings here.

One advantage of using distant supervision under these conditions for labeling emotion data, as (Wang et al., 2012) also notes, is that the label is assigned by the writer of the tweet himself/herself rather than an annotator who could wrongly decide what category a tweet is. After all, emotion is a fuzzy concept and $> 90\%$ agreement as we

[2] The data can be acquired by emailing the first author. The distribution is in the form of tweet ids and labels, to adhere to Twitter conditions.

report here is higher than the human agreement usually acquired on many NLP tasks. Another advantage of this method is obviously that it enables us to acquire a sufficiently large training set to use deep learning. We now turn to describing our deep learning methods.

5 Methods

For our core modeling, we use *Gated Recurrent Neural Networks (GRNNs)*, a modern variation of *recurrent neural networks (RNNs)*, which we now turn to introduce. For notation, we denote scalars with italic lowercase (e.g., x), vectors with bold lowercase (e.g.,$\mathbf{x}$), and matrices with bold uppercase (e.g.,$\mathbf{W}$).

Recurrent Neural Network A recurrent neural network (RNN) is one type of neural network architecture that is particularly suited for modeling sequential information. At each time step t, an RNN takes an input vector $\mathbf{x}_t \in \mathbb{R}^n$ and a hidden state vector $\mathbf{h}_{t-1} \in \mathbb{R}^m$ and produces the next hidden state $\mathbf{h}_t$ by applying the recursive operation:

$$\mathbf{h}_t = f\left(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}\right) \quad (1)$$

Where the input to hidden matrix $\mathbf{W} \in \mathbb{R}^{mxn}$, the hidden to hidden matrix $\mathbf{U} \in \mathbb{R}^{mxm}$, and the bias vector $\mathbf{b} \in \mathbb{R}^m$ are parameters of an affine transformation and f is an element-wise nonlinearity. While an RNN can in theory summarize all historical information up to time step $\mathbf{h}_t$, in practice it runs into the problem of vanishing/exploding gradients (Bengio et al., 1994; Pascanu et al., 2013) while attempting to learn long-range dependencies.

LSTM Long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) addresses this exact problem of learning long-term dependencies by augmenting an RNN with a memory cell $\mathbf{c}_t \in \mathbb{R}^n$ at each time step. As such, in addition to the input vector $\mathbf{x}_t$, the hiddent vector $\mathbf{h}_{t-1}$, an LSTM takes a cell state vector $\mathbf{c}_{t-1}$ and produces $\mathbf{h}_t$ and $\mathbf{c}_t$ via the following calculations:

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}^i x_t + \mathbf{U}^i \mathbf{h}_{t-1} + \mathbf{b}^i\right) \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}^f x_t + \mathbf{U}^f \mathbf{h}_{t-1} + \mathbf{b}^f\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}^o x_t + \mathbf{U}^o \mathbf{h}_{t-1} + \mathbf{b}^o\right) \\
\mathbf{g}_t &= \tanh\left(\mathbf{W}^g x_t + \mathbf{U}^g \mathbf{h}_{t-1} + \mathbf{b}^g\right) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \quad (2)$$

Where $\sigma(\cdot)$ and $\tanh(\cdot)$ are the element-wise sigmoid and hyperbolic tangent functions, $\odot$ the element-wise multiplication operator, and $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the *input*, forget, and *output* gates. The $\mathbf{g}_t$ is a new memory cell vector with candidates that could be added to the state. The LSTM parameters $\mathbf{W}_j$, $\mathbf{U}_j$, and $\mathbf{b}_j$ are for $j \in \{i, f, o, g\}$.

GRNNs (Cho et al., 2014; Chung et al., 2015) propose a variation of LSTM with a *reset gate* $\mathbf{r}_t$, an update state $\mathbf{z}_t$, and a new simpler hidden unit $\mathbf{h}_t$, as follows:

$$\begin{aligned}
\mathbf{r}_t &= \sigma\left(\mathbf{W}^r x_t + \mathbf{U}^r \mathbf{h}_{t-1} + \mathbf{b}^r\right) \\
\mathbf{z}_t &= \sigma\left(\mathbf{W}^z x_t + \mathbf{U}^z \mathbf{h}_{t-1} + \mathbf{b}^z\right) \\
\tilde{\mathbf{h}}_t &= \tanh\left(\mathbf{W} x_t + \mathbf{r}_t * \mathbf{U}^{\tilde{h}} \mathbf{h}_{t-1} + \mathbf{b}^{\tilde{h}}\right) \\
\mathbf{h}_t &= \mathbf{z}_t * \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) * \tilde{\mathbf{h}}_t
\end{aligned} \quad (3)$$

The GRNN parameters $\mathbf{W}_j$, $\mathbf{U}_j$, and $\mathbf{b}_j$ are for $j \in \{r, z, \tilde{h}\}$. In this set up, the hidden state is forced to ignore a previous hidden state when the reset gate is close to 0, thus enabling the network to forget or drop irrelevant information. Additionally, the update gate controls how much information carries over from a previous hidden state to the current hidden state (similar to an LSTM memory cell). We use GRNNs as they are simpler and faster than LSTM. For GRNNs, we use Theano (Theano Development Team, 2016).

Online Classifiers We compare the performance of the GRNNs to four online classifiers that are capable of handling the data size: Stochastic Gradient Descent (SGD), Multinomial Naive Bayes (MNB), Perceptron, and the Passive Agressive Classifier (PAC). These classifiers learn online from mini-batches of data. We use mini-batches of 10,000 instances with all the four classifiers. We use the *scikit-learn* implementation of these classifiers (`http://scikit-learn.org`).

Settings We aim to model Plutchik's 24 fine-grained emotions as well as his 8 primary emotion dimensions where each 3 related types of emotion (perceived as varying in intensity) are combined in one dimension. We now turn to describing our experiments experiments.

6 Experiments

6.1 Predicting Fine-Grained Emotions

As explained earlier, Plutchik organizes the 24 emotion types in the 3 main circles that we will refer to as *plutchik-1*, *plutchik-2*, and *plutchik-3*.

Emotion	Qadir (2013)		Roberts (2012)		MD (2015)		Wang (2012)		Volkova (2016)		This work	
anger	400	0.44	583	0.64	1,555	0.28	457,972	0.72	4,963	0.80	56,472	0.75
anticip	-	-	-	-	-	-	-	-	-	-	26,655	0.70
disgust	-	-	922	0.67	761	0.19	-	-	12,948	0.92	52,067	0.82
fear	592	0.54	222	0.74	2,816	0.51	11,156	0.44	9,097	0.77	98,657	0.74
joy	1,005	0.59	716	0.68	8,240	0.62	567,487	0.72	15,559	0.79	330,738	0.91
sadness	560	0.46	493	0.69	3,830	0.39	489,831	0.65	4,232	0.62	142,300	0.73
surprise	-	-	324	0.61	3849	0.45	1,991	0.14	8,244	0.64	53,915	0.86
trust	-	-	-	-	-	-	-	-	-	-	29,255	0.82
ALL	**4,500**	**0.53**	**3,777**	**0.67**	**21,051**	**0.49**	**1,991,184**	**-**	**52,925**	**0.78**	**790,059**	**0.83**

Table 6: Comparison (in F-score) of our results with GRNNs to published literature. MD = Mohammad (2015). *Note:* For space restrictions, we take the liberty of using the last name of only the first author of each work.

Emotion	SGD	MNB	PRCPTN	PAC
baseline	60.00	60.00	60.00	60.00
admiration	78.30	78.01	74.24	79.86
amazement	37.57	35.71	42.51	46.69
ecstasy	51.53	51.89	47.37	53.53
grief	38.64	36.94	37.33	48.10
loathing	0.00	0.00	2.09	2.99
rage	3.47	4.49	14.02	17.04
terror	33.23	44.12	40.48	47.00
vigilance	2.53	2.56	5.52	8.42
plutchik-1	**60.26**	**60.54**	**59.11**	**64.86**
anger	19.41	13.84	24.54	29.26
anticipation	7.46	12.63	17.29	26.70
disgust	29.51	29.87	31.83	36.60
fear	21.45	25.49	30.41	33.59
joy	72.83	72.96	72.32	75.50
sadness	50.04	51.72	39.58	49.21
surprise	8.46	4.75	17.34	19.54
trust	42.09	38.52	44.48	47.51
plutchik-2	**48.05**	**48.33**	**48.60**	**53.30**
acceptance	0.12	2.74	13.98	13.04
annoyance	80.28	80.71	78.80	81.47
apprehension	0.80	0.00	9.72	10.66
boredom	49.53	51.27	52.02	57.84
distraction	0.00	2.99	3.42	0.00
interest	21.69	30.45	34.85	44.14
pensiveness	2.61	8.08	11.22	12.27
serenity	8.87	19.57	27.23	38.59
plutchik-3	**62.20**	**64.00**	**64.04**	**68.14**
ALL	**56.84**	**57.62**	**57.25**	**62.10**

Table 3: Results in F-score with traditional online classifiers.

We model the set of emotions belonging to each of the 3 circles independently, thus casting each as an 8-way classification task. Inspired by observations from the literature and our own annotation study, we limit our data to tweets of at least 5 words with an emotional hashtag occurring at the end. We then split the data representing each of the 3 circles into 80% training (TRAIN), 10% development (DEV), and 10% testing (TEST). As mentioned above, we run experiments with a range of online, out-of-core classifiers as well as the GRNNs. To train the GRNNs, we optimize the hyper-parameters of the network on a development set as we describe below, choosing a vocabulary size of 80K words (a vocabulary size we also use for the out-of-core classifiers), a word embedding vector of size 300 dimensions learnt directly from the training data, an input maximum length of 30 words, 7 epochs, and the Adam (Kingma and Ba, 2014) optimizer with a learning rate of 0.001. We use 3 dense layers each with $1,000$ units. We use dropout (Hinton et al., 2012) for regularization, with a dropout rate of 0.5. For our loss function, we use categorical cross-entropy. We use a mini-batch (Cotter et al., 2011) size of 128. We found this architecture to work best with almost all the settings and so we fix it across the board for all experiments with GRNNs.

Results with Traditional Classifiers Results with the online classifiers are presented in terms of F-score in Table 3. As the table shows, among this group of classifiers, the Passive Agressive classifier (PAC) acquires the best performance. PAC achieves an overall F-score of 64.86% on `plutchik-1`, 53.30% on `plutchik-2`, and 68.14% on `plutchik-3`, two of which are higher than an arbitrary baseline[3] of 60%.

Results with GRNNs Table 4 presents results with GRNNs, compared with the best results using the traditional classifiers as acquired with PAC. As the table shows, the GRNN models are very successful across all the 3 classification tasks. With GRNNs, we acquire an overall F-scores of: 91.21% on `plutchik-1`, 82.32% on `plutchik-2`, and 87.47% on `plutchik-3`. These results are 26.35%, 29.02%, and 25.37% higher than PAC, respectively.

Negative Results We experiment with aug-

[3]The arbitrary baseline is higher than the majority class in the training data in any of the 3 cases.

	PAC	GRNNs		
Emotion	**f-score**	**prec**	**rec**	**f-score**
admiration	79.86	94.53	95.28	94.91
amazement	46.69	90.44	89.02	89.73
ecstasy	53.53	83.49	90.01	86.62
grief	48.10	85.07	81.13	83.05
loathing	2.99	83.87	54.17	65.82
rage	17.04	80.00	75.11	77.48
terror	47.00	91.15	84.01	87.44
vigilance	8.42	71.93	70.69	71.30
plutchik-1	**64.86**	**91.26**	**91.24**	**91.21**
anger	29.26	74.95	69.20	71.96
anticipation	26.70	70.05	69.00	69.52
disgust	36.60	82.18	68.84	74.92
fear	33.59	73.74	72.51	73.12
joy	75.50	90.96	93.88	92.40
sadness	49.21	73.20	82.04	77.37
surprise	19.54	85.60	67.40	75.42
trust	47.51	82.43	76.83	79.53
plutchik-2	**53.30**	**82.53**	**82.46**	**82.32**
acceptance	13.04	77.10	71.76	74.33
annoyance	81.47	91.46	95.01	93.20
apprehension	10.66	80.40	61.07	69.41
boredom	57.84	85.95	84.40	85.16
distraction	0.00	87.50	25.00	38.89
interest	44.14	86.79	78.38	82.37
pensiveness	12.27	91.87	43.24	58.80
serenity	38.59	82.15	78.16	80.11
plutchik-3	**68.14**	**88.94**	**89.08**	**88.89**
ALL	**62.10**	**87.58**	**87.59**	**87.47**

Table 4: Results with GRNNs across Plutchik's 24 emotion categories. We compare to best-performing traditional classifier (i.e. Passive Aggressive).

menting training data reported here in two ways: 1) For each emotion type, we concatenate the training data with training data of tweets that are more (or less) intense from the same sector/dimension in the wheel, and 2) for each emotion type, we add tweets where emotion hashtags occur in the last quarter of a tweet (which were originally filtered out from TRAIN). However, we gain no improvements based on either of these methods, thus reflecting the importance of using high-quality training data and the utility of our strict pipeline.

6.2 Predicting 8 Primary Dimensions

We now investigate the task of predicting each of the 8 primary emotion dimensions represented by the sectors of the wheel (where the three degrees of intensity of a given emotion are reduced to a single emotion dimension [e.g., {*ecstasy, joy, serenity*} are reduced to the *joy* dimension]). We concatenate the 80% training data (TRAIN) from each of the 3 circles' data into a single training set (TRAIN-ALL), the 10% DEV to form DEV-ALL, and the 10% TEST to form TEST-ALL. We test a number of hyper-parameters on DEV and find the ones we have identified on the fine-grained prediction to work best and so we adopt them as is with the exception of limiting to only 2 epochs. We believe that with a wider exploration of hyper-parameters, improvements could be possible. As Table 5 shows, we are able to model the 8 dimensions with an overall superior accuracy of 95.68%. As far as we know, this is the first work on modeling these dimensions.

Dimension	**prec**	**rec**	**f-score**
anger	97.40	97.72	97.56
anticipation	91.18	89.95	90.56
disgust	96.20	93.94	95.06
fear	94.97	94.38	94.68
joy	94.61	96.40	95.50
sadness	95.52	95.25	95.39
surprise	94.99	91.62	93.27
trust	96.36	97.58	96.96
All	**95.68**	**95.68**	**95.68**

Table 5: GRNNs results across 8 emotion dimensions. Each dimension represents three different emotions. For example, the *joy* dimension represents *serenity, joy* and *ecstasy*.

Emotion	**Volkova (2016) model**	**This work**
anger	12.38	74.95
disgust	5.71	82.18
fear	11.18	73.74
joy	44.57	90.96
sadness	18.04	73.20
surprise	5.33	85.60
ALL	**26.95**	**80.12**

Table 7: Comparison (in acc) to (Volkova and Bachrach, 2016)'s model.

7 Comparisons to Other Systems

We compare our results on the 8 basic emotions to the published literature. As Table 6 shows, on this subset of emotions, our system is 4.53% (acc) higher than the best published results (Volkova and Bachrach, 2016), facilitated by the fact that we have an order of magnitude more training data. As shown in Table 7, we also apply (Volkova and Bachrach, 2016)'s pre-trained model on our test set of the 6 emotions they predict (which belong to `plutchik-2`), and acquire an overall accuracy of 26.95%, which is significantly lower than our accuracy.

8 Conclusion

In this paper, we built a large, automatically curated dataset for emotion detection using distant supervision and then used GRNNs to model fine-grained emotion, achieving a new state-of-the-art performance. We also extended the classification to 8 primary emotion dimensions situated in psychological theory of emotion.

References

Conor M. Steckler Aaron C. Weidman and Jessica L. Tracy. 2017. The jingle and jangle of emotion assessment: Imprecise measurement, casual scale usage, and conceptual fuzziness in emotion research. *Emotion* .

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *Text, Speech and Dialogue*. Springer, pages 196–205.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5(2):157–166.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576* .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Junyoung Chung, Caglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *ICML*. pages 2067–2075.

Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. 2011. Better mini-batch algorithms via accelerated gradient methods. In *Advances in neural information processing systems*. pages 1647–1655.

Munmun De Choudhury, Scott Counts, and Michael Gamon. 2012. Not all moods are created equal! exploring human emotional states in social media.

P. Ekman. 1972. Universal and cultural differences in facial expression of emotion. *Nebraska Symposium on Motivation* pages 207–283.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford* 1(12).

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* 57:345–420.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2*. Association for Computational Linguistics, pages 581–586.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT Press.

Alex Graves. 2012. Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer, pages 5–13.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, pages 6645–6649.

Alex Graves and Jürgen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*. pages 545–552.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* .

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

Hyisung C Hwang and David Matsumoto. 2016. Emotional expression. *The Expression of Emotion: Philosophical, Psychological and Legal Perspectives* page 137.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*. pages 2096–2104.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *ACL (2)*. pages 489–493.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. volume 14, pages 1188–1196.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521(7553):436–444.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185* .

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *EMNLP*. Citeseer, pages 2326–2335.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*. Association for Computational Linguistics, pages 25–30.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 142–150.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Hlt-naacl*. volume 13, pages 746–751.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, pages 1003–1011.

Gilad Mishne and Maarten De Rijke. 2006. Capturing global mood levels using blog posts. In *AAAI spring symposium: computational approaches to analyzing weblogs*. pages 145–152.

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

Saif M Mohammad. 2012. #emotional tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 246–255.

Saif M Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence* 31(2):301–326.

Thin Nguyen. 2010. Mood patterns and affective lexicon access in weblogs. In *Proceedings of the ACL 2010 Student Research Workshop*. Association for Computational Linguistics, pages 43–48.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*. volume 10.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28:1310–1318.

Robert Plutchik. 1980. *Emotion: A psychoevolutionary synthesis*. Harpercollins College Division.

Robert Plutchik. 1985. On emotion: The chicken-and-egg problem revisited. *Motivation and Emotion* 9(2):197–200.

Robert Plutchik. 1994. *The psychology and biology of emotion.*. HarperCollins College Publishers.

Robert Plutchik. 2001. The nature of emotions human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American scientist* 89(4):344–350.

Matthew Purver and Stuart Battersby. 2012. Experimenting with distant supervision for emotion classification. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 482–491.

Jonathon Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL student research workshop*. Association for Computational Linguistics, pages 43–48.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Context-sensitive twitter sentiment classification using neural network. In *AAAI*. pages 215–221.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Citeseer, volume 1631, page 1642.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, pages 70–74.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Yuki Tanaka, Hiroya Takamura, and Manabu Okumura. 2005. Extraction and classification of facemarks. In *Proceedings of the 10th international conference on Intelligent user interfaces*. ACM, pages 28–34.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1422–1432.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014a. Building large-scale twitter-specific sentiment lexicon: A representation learning approach. In *COLING*. pages 172–182.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014b. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*. pages 1555–1565.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.

Svitlana Volkova and Yoram Bachrach. 2016. Inferring perceived demographics from user emotional tone and user-environment emotional contrast. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2012. Harnessing twitter" big data" for automatic emotion identification. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*. IEEE, pages 587–592.

Jasy Liew Suet Yan and Howard R Turtle. 2016a. Exploring fine-grained emotion detection in tweets. In *Proceedings of NAACL-HLT*. pages 73–80.

Jasy Liew Suet Yan and Howard R Turtle. 2016b. Exposing a set of fine-grained emotion categories from tweets. In *25th International Joint Conference on Artificial Intelligence*. page 8.

Jasy Liew Suet Yan, Howard R Turtle, and Elizabeth D Liddy. 2016. Emotweet-28: A fine-grained emotion corpus for sentiment analysis .

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007. Emotion classification using web blog corpora. In *Web Intelligence, IEEE/WIC/ACM International Conference on*. IEEE, pages 275–278.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *AAAI*. pages 3087–3093.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. pages 649–657.

Beyond Binary Labels: Political Ideology Prediction of Twitter Users

Daniel Preoţiuc-Pietro
Positive Psychology Center
University of Pennsylvania
danielpr@sas.upenn.edu

Ye Liu*
School of Computing
National University of Singapore
liuye@comp.nus.edu.sg

Daniel J. Hopkins
Political Science Department
University of Pennsylvania
danhop@sas.upenn.edu

Lyle Ungar
Computing & Information Science
University of Pennsylvania
ungar@cis.upenn.edu

Abstract

Automatic political preference prediction from social media posts has to date proven successful only in distinguishing between publicly declared liberals and conservatives in the US. This study examines users' political ideology using a seven-point scale which enables us to identify politically moderate and neutral users – groups which are of particular interest to political scientists and pollsters. Using a novel data set with political ideology labels self-reported through surveys, our goal is two-fold: a) to characterize the political groups of users through language use on Twitter; b) to build a fine-grained model that predicts political ideology of unseen users. Our results identify differences in both political leaning and engagement and the extent to which each group tweets using political keywords. Finally, we demonstrate how to improve ideology prediction accuracy by exploiting the relationships between the user groups.

1 Introduction

Social media is used by people to share their opinions and views. Unsurprisingly, an important part of the population shares opinions and news related to politics or causes they support, thus offering strong cues about their political preferences and ideologies. In addition, political membership is also predictable purely from one's interests or demographics — it is much more likely for a religious person to be conservative or for a younger person to lean liberal (Ellis and Stimson, 2012).

User trait prediction from text is based on the assumption that language use reflects a user's demographics, psychological states or preferences. Applications include prediction of age (Rao et al., 2010; Flekova et al., 2016b), gender (Burger et al., 2011; Sap et al., 2014), personality (Schwartz et al., 2013; Preoţiuc-Pietro et al., 2016), socioeconomic status (Preoţiuc-Pietro et al., 2015a,b; Liu et al., 2016c), popularity (Lampos et al., 2014) or location (Cheng et al., 2010).

Research on predicting political orientation has focused on methodological improvements (Pennacchiotti and Popescu, 2011) and used data sets with publicly stated dichotomous political orientation labels due to their easy accessibility (Sylwester and Purver, 2015). However, these data sets are not representative samples of the entire population (Cohen and Ruths, 2013) and do not accurately reflect the variety of political attitudes and engagement (Kam et al., 2007).

For example, we expect users who state their political affiliation in their profile description, tweet with partisan hashtags or appear in public party lists to use social media as a means of popularizing and supporting their political beliefs (BarberASa, 2015). Many users may choose not to publicly post about their political preference for various social goals or perhaps this preference may not be strong or representative enough to be disclosed online. Dichotomous political preference also ignores users who do not have a political ideology. All of these types of users are very important for researchers aiming to understand group preferences, traits or moral values (Lewis and Reiley, 2014; Hersh, 2015).

The most common political ideology spectrum in the US is the conservative – liberal (Ellis and Stimson, 2012). We collect a novel data set of Twitter users mapped to this seven-point spectrum which allows us to:

* Work carried out during a research visit at the University of Pennsylvania

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 729–740
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1068

1. Uncover the differences in language use between ideological groups;
2. Develop a user-level political ideology prediction algorithm that classifies all levels of engagement and leverages the structure in the political ideology spectrum.

First, using a broad range of language features including unigrams, word clusters and emotions, we study the linguistic differences between the two ideologically extreme groups, the two ideologically moderate groups and between both extremes and moderates in order to provide insight into the content they post on Twitter. In addition, we examine the extent to which the ideological groups in our data set post about politics and compare it to a data set obtained similarly to previous work.

In prediction experiments, we show how accurately we can distinguish between opposing ideological groups in various scenarios and that previous binary political orientation prediction has been oversimplified. Then, we measure the extent to which we can predict the two dimensions of political leaning and engagement. Finally, we build an ideology classifier in a multi-task learning setup that leverages the relationships between groups.[1]

2 Related Work

Automatically inferring user traits from their online footprints is a prolific topic of research, enabled by the increasing availability of user generated data and advances in machine learning. Beyond its research oriented goals, user profiling has important industry applications in online marketing, personalization or large-scale audience profiling. To this end, researchers have used a wide range of types of online footprints, including video (Subramanian et al., 2013), audio (Alam and Riccardi, 2014), text (Preoţiuc-Pietro et al., 2015a), profile images (Liu et al., 2016a), social data (Van Der Heide et al., 2012; Hall et al., 2014), social networks (Perozzi and Skiena, 2015; Rout et al., 2013), payment data (Wang et al., 2016) and endorsements (Kosinski et al., 2013).

Political orientation prediction has been studied in two related, albeit crucially different scenarios, as also identified in (Zafar et al., 2016). First, researchers aimed to identify and quantify orientation of words (Monroe et al., 2008), hashtags (Weber et al., 2013) or documents (Iyyer et al., 2014), or to detect bias (Yano et al., 2010) or impartiality (Zafar et al., 2016) at a document level.

Our study belongs to the second category, where political orientation is inferred at a user-level. All previous studies study labeling US conservatives vs. liberals using either text (Rao et al., 2010), social network connections (Zamal et al., 2012), platform-specific features (Conover et al., 2011) or a combination of these (Pennacchiotti and Popescu, 2011; Volkova et al., 2014), with very high reported accuracies of up to 94.9% (Conover et al., 2011).

However, all previous work on predicting user-level political preferences are limited to a binary prediction between liberal/democrat and conservative/republican, disregarding any nuances in political ideology. In addition, as the focus of the studies is more on the methodological or interpretation aspects of the problem, another downside is that the user labels were obtained in simple, albeit biased ways. These include users who explicitly state their political orientation on user lists of party supporters (Zamal et al., 2012; Pennacchiotti and Popescu, 2011), supporting partisan causes (Rao et al., 2010), by following political figures (Volkova et al., 2014) or party accounts (Sylwester and Purver, 2015) or that retweet partisan hashtags (Conover et al., 2011). As also identified in (Cohen and Ruths, 2013) and further confirmed later in this study, these data sets are biased: most people do not clearly state their political preference online – fewer than 5% according to Priante et al. (2016) – and those that state their preference are very likely to be political activists. Cohen and Ruths (2013) demonstrated that predictive accuracy of classifiers is significantly lower when confronted with users that do not explicitly mention their political orientation. Despite this, their study is limited because in their hardest classification task, they use crowdsourced political orientation labels, which may not correspond to reality and suffer from biases (Flekova et al., 2016a; Carpenter et al., 2016). Further, they still only look at predicting binary political orientation. To date, no other research on this topic has taken into account these findings.

3 Data Set

The main data set used in this study consists of 3,938 users recruited through the Qualtrics platform ($\mathcal{D}_1$). Each participant was compensated

[1] Data is available at http://www.preotiuc.ro

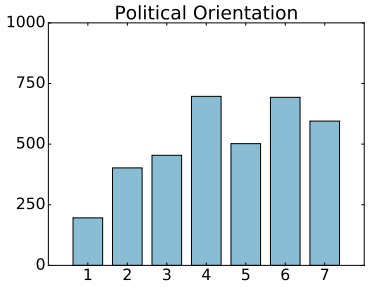


Figure 1: Distribution of political ideology in our data set, from 1 – Very Conservative through 7 – Very Liberal.

with 3 USD for 15 minutes of their time. All participants first answered the same demographic questions (including political ideology), then were directed to one of four sets of psychological questionnaires unrelated to the political ideology question. They were asked to self-report their political ideology on a seven point scale: *Very conservative* (1), *Conservative* (2), *Moderately conservative* (3), *Moderate* (4), *Moderately liberal* (5), *Liberal* (6), *Very liberal* (7). In addition, participants had the option of choosing *Apathetic* and *Other*, which have ambiguous fits on the conservative – liberal spectrum and were removed from our analysis (399 users). We also asked participants to self-report their gender (2322 female, 1205 male, 12 other) and age. Participants were all from the US in order to limit the impact of cultural and political factors. The political ideology distribution in our sample is presented in Figure 1.

We asked users their Twitter handle and downloaded their most recent 3,200 tweets, leading to a total of 4,833,133 tweets. Before adding users to our 3,938 user data set, we performed the following checks to ensure that the Twitter handle was the user's own: 1) *after* compensation, users were if they were truthful in reporting their handle and if not, we removed their data from analysis; 2) we manually examined all handles marked as verified by Twitter or that had over 2000 followers and eliminated them if they were celebrities or corporate/news accounts, as these were unlikely the users who participated in the survey. This study received approval from the Institutional Review Board (IRB) of the University of Pennsylvania.

In addition, to facilitate comparison to previous work, we also use a data set of 13,651 users with overt political orientation ($\mathcal{D}_2$). We selected popular political figures unambiguously associated with US liberal politics (@SenSanders, @JoeBiden, @CoryBooker, @JohnKerry) or US conservative politics (@marcorubio, @tedcruz, @RandPaul, @RealBenCarson). Liberals in our set ($N_l = 7417$) had to follow on Twitter all of the liberal political figures and none of the conservative figures. Likewise, conservative users ($N_c = 6234$) had to follow all of the conservative figures and no liberal figures. We downloaded up to 3,200 of each user's most recent tweets, leading to a total of 25,493,407 tweets. All tweets were downloaded around 10 August 2016.

4 Features

In our analysis, we use a broad range of linguistic features described below.

Unigrams We use the bag-of-words representation to reduce each user's posting history to a normalised frequency distribution over the vocabulary consisting of all words used by at least 10% of the users (6,060 words).

LIWC Traditional psychological studies use a dictionary-based approach to representing text. The most popular method is based on Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2001), and automatically counts word frequencies for 64 different categories manually constructed based on psychological theory. These include different parts-of-speech, topical categories and emotions. Each user is thereby represented as a frequency distribution over these categories.

Word2Vec Topics An alternative to LIWC is to use automatically generated word clusters i.e., groups of words that are semantically and/or syntactically similar. The clusters help reducing the feature space and provides additional interpretability.

To create these groups of words, we use an automatic method that leverages word co-occurrence patterns in large corpora by making use of the distributional hypothesis: similar words tend to co-occur in similar contexts (Harris, 1954). Based on co-occurrence statistics, each word is represented as a low dimensional vector of numbers with words closer in this space being more similar (Deerwester et al., 1990). We use the method from (Preoţiuc-Pietro et al., 2015a) to compute topics using word2vec similarity (Mikolov et al., 2013a,b) and spectral clustering (Shi and Malik, 2000; von Luxburg, 2007) of different sizes (from 30 to 2000). We have tried other alternatives to building clusters: using other word similarities to

generate clusters – such as NPMI (Lampos et al., 2014) or GloVe (Pennington et al., 2014) as proposed in (Preoţiuc-Pietro et al., 2015a) – or using standard topic modelling approached to create soft clusters of words e.g., Latent Dirichlet Allocation (Blei et al., 2003). For brevity, we present experiments with the best performing feature set containing 500 Word2Vec clusters. We aggregate all the words posted in a users' tweets and represent each user as a distribution of the fraction of words belonging to each cluster.

Sentiment & Emotions We hypothesise that different political ideologies differ in the type and amount of emotions the users express through their posts. The most studied model of discrete emotions is the Ekman model (Ekman, 1992; Strapparava and Mihalcea, 2008; Strapparava et al., 2004) which posits the existence of six basic emotions: anger, disgust, fear, joy, sadness and surprise. We automatically quantify these emotions from our Twitter data set using a publicly available crowd-sourcing derived lexicon of words associated with any of the six emotions, as well as general positive and negative sentiment (Mohammad and Turney, 2010, 2013). Using these lexicons, we assign a predicted emotion to each message and then average across all users' posts to obtain user level emotion expression scores.

Political Terms In order to select unigrams pertaining to politics, we assigned the most frequent 12,000 unigrams in our data set to three categories:

- **Political words**: mentions of political terms (234);
- **Political NEs**: mentions of politician proper names out of the political terms (39);
- **Media NEs**: mentions of political media sources and pundits out of the political terms (20).

This coding was initially performed by a research assistant studying political science with good knowledge of US politics and were further filtered and checked by one of the authors.

5 Analysis

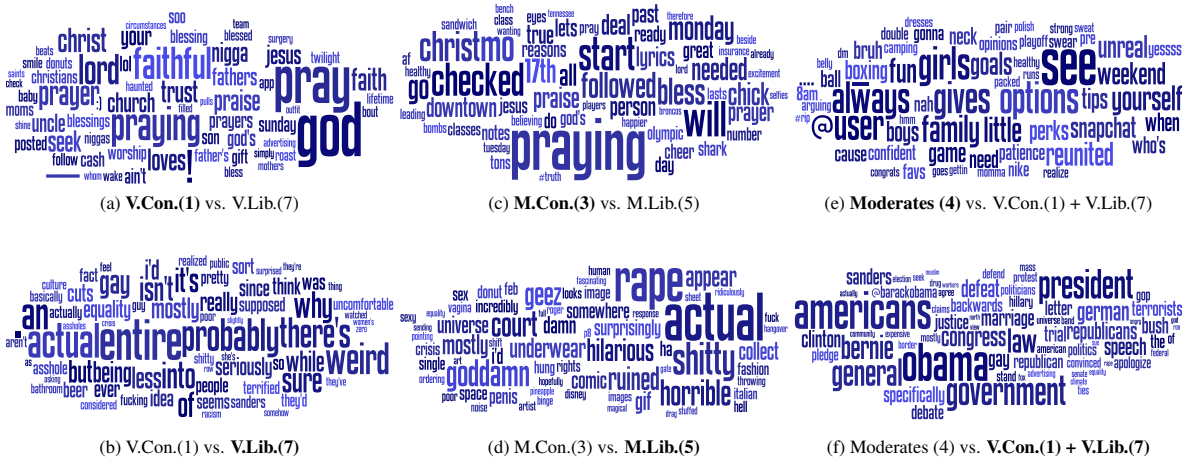
First, we explore the relationships between language use and political ideological groups within each feature set and pairs of opposing user groups. To illustrate differences between ideological groups we compare the two political extremes (Very Conservative – Very Liberal) and the political moderates (Moderate Conservative – Moderate Liberal). We further compare outright moderates with a group combining the two political extremes to study if we can uncover differences in political engagement and extremity, regardless of the conservative–liberal leaning.

We use univariate partial linear correlations with age and gender as co-variates to factor out the influence of basic demographics. For example, in $\mathcal{D}_1$, users who reported themselves as very conservative are older and more likely males ($\mu_{age} = 35.1, \text{pct}_{male} = 44\%$) than the data average ($\mu_{age} = 31.2, \text{pct}_{male} = 35\%$). Additionally, prior to combining the two ideologically extreme groups, we sub-sampled the larger class (Very Liberal) to match the smaller class (Very Conservative) in age and gender. In the later prediction experiments, we do not perform matching, as this represents useful signal for classification (Ellis and Stimson, 2012). Results with unigrams are presented in Figure 2 and with the other features in Table 1. These are selected using standard statistical significance tests.

5.1 Very Conservatives vs. Very Liberals

The comparison between the extreme categories reveals the largest number of significant differences. The unigrams and Word2Vec clusters specific to conservatives are dominated by religion specific terms ('praying', 'god', W2V-485, W2V-018, W2V-099, L-RELIG), confirming a well-documented relationship (Gelman, 2009) and words describing family relationships ('uncle', 'son', L-FAMILY), another conservative value (Lakoff, 1997). The emphasis on religious terms among conservatives is consistent with the claim that many Americans associate 'conservative' with 'religious' (Ellis and Stimson, 2012). Extreme liberals show a tendency to use more adjectives (W2V-075, W2V-110), adverbs (L-ADVERB), conjunctions (L-CONJ) and comparisons (L-COMPARE) which indicate more nuanced and complex posts. Extreme conservatives post tweets higher in all positive emotions than liberals (L-POSEMO, Emot-Joy, Emot-Positive), confirming a previously hypothesised relationship (Napier and Jost, 2008). However, extreme liberals are not associated with posting negative emotions either, only using words that reflect more anxiety (L-ANX), which is related to neuroticism in which the liberals are higher (Gerber et al., 2010).

Political term analysis reveals the partisan terms



(a) **V.Con.(1)** vs. V.Lib.(7)

(c) **M.Con.(3)** vs. M.Lib.(5)

(e) **Moderates (4)** vs. V.Con.(1) + V.Lib.(7)

(b) V.Con.(1) vs. **V.Lib.(7)**

(d) M.Con.(3) vs. **M.Lib.(5)**

(f) Moderates (4) vs. **V.Con.(1) + V.Lib.(7)**

Figure 2: Unigrams with the highest 80 Pearson correlations shown as word clouds in three vertical panels with a binary variable representing the two ideological groups compared. The size of the unigram is scaled by its correlation with the ideological group in bold. The color indexes relative frequency, from light blue (rarely used) to dark blue (frequently used). All correlations are significant at $p < .05$ and controlled for age and gender.

r	Category	Words	r	Category	Words
V.Con.(1) vs. V.Lib.(7)			V.Con.(1) vs. **V.Lib.(7)**		
.249	W2V–485	god, peace, thankful, pray, bless, blessed, prayers, praying	.236	W2V–075	bad, kind, weird, kinda, horrible, creepy, strange, extremely
.180	W2V–018	jesus, lord, christ, sin, grace, god's, praise, gods, glory, thou	.195	W2V–238	an, excuse, actual, idiot, asshole, example, absolute
.156	W2V–099	church, bible, serve, worship, preach, christians, pastor	.192	W2V–487	into, through, must, myself, decided, completely, upon
.140	W2V–491	soooo, soo, sooooo, soooooo, tooo, soooooooo, toooo	.191	W2V–110	quite, awful, exciting, brilliant, perfectly, usual
.119	W2V–027	kno, yu, abt, tht, dnt, wut, tru, somethin, ion, wen	.186	W2V–448	off, almost, whole, literally, entire, basically, ridiculous
.204	L–RELIG	god, hell, holy, soul, pray, angel, praying, christ, sin, amen	.175	L–ANX	awkward, worry, scared, fear, afraid, horrible, scary, upset
.145	L–POSEMO	love, good, lol, :), great, happy, best, thanks, win, free	.164	L–ADVERB	just, so, when, about, now, how, too, why, back, really
.127	L–FAMILY	baby, family, mom, dad, son, bro, mother, babies, fam, folks	.161	L–CONJ	and, so, but, if, when, how, as, or, because, then
.118	L–NETSPEAK	rt, u, lol, :), twitter, gonna, yo, ur, omg, ya	.147	L–COMPARE	like, more, as, best, than, better, after, most, before, same
.101	L–YOU	you, your, u, you're, ur, ya, yourself, youre, you'll, you've	.138	L–DIFFER	not, but, if, or, really, can't, than, other, didn't, actually
.152	Emot–Joy	love, good, happy, hope, god, birthday, fun, favorite, pretty			
.086	Emot–Positive	love, good, happy, hope, god, birthday, real, fun, favorite			
.107	Emot–Surprise	good, hope, birthday, excited, money, finally, chance, guess			
.132 ↓ .068	Political Terms	#pjnet, #tcot, @foxnews, polls, @realdonaldtrump, @tedcruz, @yahoonews	.161 ↓ .090	Political Terms	gay, sanders, racism, racist, rape, @barackobama, democracy, feminist, democratic, protesting, protest, bernie, feminism, protesters, transgender
M.Con.(3) vs. M.Lib.(5)			M.Con.(3) vs. **M.Lib.(5)**		
.108	W2V–485	god, peace, thankful, pray, bless, blessed, prayers, praying	.116	W2V–458	hilarious, celeb, capaldi, corrie, chatty, corden, barrowman
.088	W2V–018	jesus, lord, christ, sin, grace, god's, praise, gods, glory, thou	.106	W2V–373	photo, art, pictures, photos, instagram, photoset, image
.085	W2V–214	frank, savage, brad, ken, kane, pitt, watson, leonardo	.106	W2V–316	hot, sex, naked, adult, teen, porn, lesbian, tube, tits
.085	W2V–436	luck, lucky, boss, sir, c'mon, mate, bravo, ace, pal, keeper	.087	W2V–024	turn, accidentally, barely, constantly, onto, bug, suddenly
			.086	W2V–389	ha, ooo, uh, ohhh, ohhhh, ma'am, gotcha, gee, ohhhhh
.096	L–RELIG	god, hell, holy, soul, pray, angel, praying, christ, sin, amen	.104	L–SEXUAL	fuck, gay, sex, sexy, dick, naked, fucks, cock, aids, cum
.093	L–DRIVES	love, good, lol, :), great, happy, best, thanks, win, free	.088	L–ANGER	hate, fuck, hell, stupid, mad, sucks, suck, war, dumb, ugly
.093	L–WE	we, our, us, let's, we're, lets, we'll, we've, ourselves, we'd			
.087	L–AFFILIATION	love, we, our, use, help, twitter, friends, family, join, friend			
.086	Emot–Joy	love, good, happy, hope, god, birthday, fun, favorite, pretty	.097	Emot–Disgust	bad, hate, shit, finally, damn, feeling, hell, bitch, boy, sick
.096	Political Terms	islamic	.136	Political Terms	rape
			.086		rights
Moderates (4) vs. V.Con.(1)+V.Lib.(7)			Moderates (4) vs. **V.Con.(1)+V.Lib.(7)**		
.084	W2V–098	girls, boys, em, ladies, bitches, hoes, grown, dudes, dem	.191	W2V–309	obama, president, scott, hillary, romney, clinton, ed, sarah
			.188	W2V–237	freedom, violence, revolution, muslim, muslims, terrorists
			.184	W2V–269	bill, rights, congress, gop, republicans, republican, passes
			.174	W2V–296	justice, rule, crusade, civil, pope, plot, humanity, terror
			.160	W2V–288	law, general, legal, safety, officer, emergency, agent
			.120	L–POWER	up, best, over, win, down, help, god, big, high, top
			.103	L–RELIG	god, hell, holy, soul, pray, angel, praying, christ, sin, amen
			.100	L–ARTICLE	the, a, an
			.089	L–DEATH	dead, die, died, war, alive, dying, wars, dies, buried, bury
			.083	L–RISK	bad, stop, wrong, worst, lose, trust, safe, worse, losing
			.118	Emot–Fear	watch, bad, god, hate, change, feeling, hell, crazy, bitch, die
			.094	Emot–Disgust	bad, hate, shit, finally, damn, feeling, hell, bitch, boy, sick
			.086	Emot–Negative	wait, bad, hate, shit, black, damn, ass, wrong, vote, feeling
			.084	Emot–Sadness	bad, hate, music, black, vote, feeling, hell, crazy, lost, bitch
			.181 ↓ .103	Political Terms	obama, liberal, president, government, senators, bernie, law, #demdebate, same-sex, feminist, congress, republicans, clinton, gay, #p2, iran, activists, bush, sanders, obamacare, terrorists, justice, debate, republican, #obamacare, @moveon, @barackobama, #tcot, democrats, politics, ...

Table 1: Pearson correlations between political ideology groups and text features, split into Word2Vec clusters (W2V), LIWC categories (L), emotions (Emot) and political terms (maximum 5 categories per group). All correlations are significant at $p < .01$, two-tailed t-test and are controlled for age and gender. Words in a category are sorted by frequency in our data set.

employed by both sides. For example, conservatives retweet or mention politicians such as Donald Trump or Ted Cruz, while liberals mention Barack Obama. Extreme conservatives also reference known partisan conservative media sources (@foxnews, @yahoonews) and hashtags (#pjnet,

#tcot), while extreme liberals focus on issues ('gay', 'racism', 'feminism', 'transgender'). This perhaps reflects the desire for conservatives on Twitter to identify like-minded individuals, as extreme conservatives are a minority on the platform. Liberals, by contrast, use the platform to discuss and popularize their causes.

5.2 Moderate Conservatives vs. Moderate Liberals

Comparing the two sides of moderate users reveals a slightly more nuanced view of the two ideologies. While moderate conservatives still make heavy use of religious terms and express positive emotions (Emot-Joy, L-DRIVES), they also use affiliative language (L-AFFILIATION) and plural pronouns (L-WE). Moderate liberals are identified by very different features compared to their more extreme counterparts. Most striking is the use of swear and sex words (L-SEXUAL, L-ANGER, W2V-316), also highlighted by Sylwester and Purver (2015). Two word clusters relating to British culture (W2V-458) and art (W2V-373) reflect that liberals are more inclined towards arts (Dollinger, 2007). Statistically significant political terms are very few compared to the previous comparison, probably due to their lower overall usage, which we further investigate later.

5.3 Moderates vs. Extremists

Our final comparison looks at outright moderates compared to the two extreme groups combined, as we hypothesise the existence of a difference in overall political engagement. Moderates are not characterized by many features besides a topic of casual words (W2V-098), indicating the heterogeneity of this group of users. However, regardless of their orientation, the ideological extremists stand out from moderates. They use words and word clusters related to political actors (W2V-309), issues (W2V-237) and laws (W2V-296, W2V-288). LIWC analysis uncovers differences in article use (L-ARTICLE) or power words (L-POWER) specific of political tweets. The overall sentiment of these users is negative (Emot-Fear, Emot-Disgust, Emot-Sadness, L-DEATH) compared to moderates. This reveals – combined with the finding from the first comparison – that while extreme conservatives are overall more positive than liberals, both groups share negative expression. Political terms are almost all significantly correlated with the extreme ideological groups, confirming the existence of a difference in political engagement which we study in detail next.

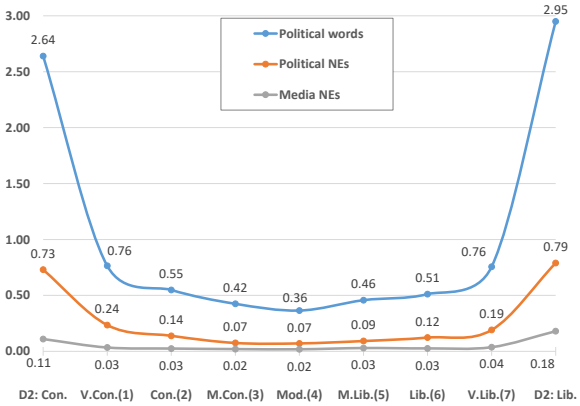


Figure 3: Distribution of political word and entity usage across political categories in % from the total words used. Users from data set $\mathcal{D}_2$ who are following the accounts of the four political figures are prefixed with *D2*. The rest of the categories are from data set $\mathcal{D}_1$.

5.4 Political Terms

Figure 3 presents the use of the three types of political terms across the 7 ideological groups in $\mathcal{D}_1$ and the two political groups from $\mathcal{D}_2$. We notice the following:

- $\mathcal{D}_2$ has a huge skew towards political words, with an average of more than three times more political terms across all three categories than our extreme classes from $\mathcal{D}_1$;
- Within the groups in $\mathcal{D}_1$, we observe an almost perfectly symmetrical U-shape across all three types of political terms, confirming our hypothesis about political engagement;
- The difference between 1–2/6–7 is larger than 2–3/5–6. The extreme liberals and conservatives are disproportionately political, and have the potential to give Twitter's political discussions an unrepresentative, extremist hue (Fiorina, 1999). It is also possible, however, that characterizing one as an extreme liberal or conservative indicates as much about her level of political engagement as it does about her placement on a left-right scale (Converse, 1964; Broockman, 2016).

6 Prediction

In this section we build predictive models of political ideology and compare them to data sets obtained using previous work.

6.1 Cross-Group Prediction

First, we experiment with classifying between conservatives and liberals across various levels of political engagement in $\mathcal{D}_1$ and between the two polarized groups in $\mathcal{D}_2$. We use logistic regression classification to compare three setups in Table 2 with results measured with ROC AUC as the classes are slightly inbalanced:

- 10-fold cross-validation where training is performed on the same task as the testing (principal diagonal);
- A train–test setup where training is performed on one task (presented in rows) and testing is performed on another (presented in columns);
- A domain adaptation setup (results in brackets) where on each of the 10 folds, the 9 training folds (presented in rows) are supplemented with all the data from a different task (presented in columns) using the EasyAdapt algorithm (Daumé III, 2007) as a proof on concept on the effects of using additional distantly supervised data. Data pooling lead to worse results than EasyAdapt.

Each of the three tasks from $\mathcal{D}_1$ have a similar number of training samples, hence we do not expect that data set size has any effects in comparing the results across tasks.

The results with both sets of features show that:

- Prediction performance is much higher for $\mathcal{D}_2$ than for $\mathcal{D}_1$, with the more extreme groups in $\mathcal{D}_1$ being easier to predict than the moderate groups. This confirms that the very high accuracies reported by previous research are an artifact of user label collection and that on regular users, the expected accuracy is much lower (Cohen and Ruths, 2013). We further show that, as the level of political engagement decreases, the classification problem becomes even harder;
- The model trained on $\mathcal{D}_2$ and Word2Vec word clusters performs significantly worse on $\mathcal{D}_1$ tasks even if the training data is over 10 times larger. When using political words, the $\mathcal{D}_2$ trained classifier performs relatively well on all tasks from $\mathcal{D}_1$;
- Overall, using political words as features performs better than Word2Vec clusters in the binary classification tasks;
- Domain adaptation helps in the majority of cases, leading to improvements of up to .03 in AUC (predicting 2v6 supplemented with 3v5 data).

Train	Test			
	1v7	**2v6**	**3v5**	**D2**
1v7	**.785**	.639 (*.681*)	.575 (*.598*)	.705 (.887)
2v6	.729 (*.789*)	**.662**	.574 (.586)	.663 (.889)
3v5	.618 (.778)	.617 (*.690*)	**.581**	.684 (.887)
D2	.708 (.764)	.627 (.644)	.571 (.574)	**.891**

(a) Word2Vec 500

Train	Test			
	1v7	**2v6**	**3v5**	**D2**
1v7	**.785**	.657 (.679)	.589 (.616)	.928 (*.976*)
2v6	.739 (.773)	**.679**	.593 (.612)	.920 (*.976*)
3v5	.727 (.766)	.636 (.670)	.590	.891 (*.976*)
D2	.766 (*.789*)	.677 (*.683*)	**.625** (.613)	**.972**

(b) Political Terms

Table 2: Prediction results of the logistic regression classification in ROC AUC when discriminating between two political groups across different levels of engagement and both data sets. The binary classifier from data set $\mathcal{D}_2$ is represented by *D2*, the rest of the categories are from data set $\mathcal{D}_1$. Results on the principal diagonal represent 10-fold cross-validation results (training in-domain). Results off-diagonal represent training the classifier from the column and testing on the problem indicated in the row (training out-of-domain). Numbers in brackets indicate performance when the training data was added in the 10-fold cross-validation setup using the EasyAdapt algorithm (domain adaptation). Best results without domain adaptation are in bold, while the best results with domain adaptation are in italics.

6.2 Political Leaning and Engagement Prediction

Political leaning (Conservative – Liberal, excluding the Moderate group) can be considered an ordinal variable and the prediction problem framed as one of regression. In addition to the political leaning prediction, based on analysis and previous prediction results, we hypothesize the existence of a separate dimension of political engagement regardless of the partisan side. Thus, we merge users from classes 3–5, 2–6, 1–7 and create a variable with four values, where the lowest value is represented by moderate users (4) and the highest value is represented by either very conservative (1) or very liberal (7) users.

We use a linear regression algorithm with an Elastic Net regularizer (Zou and Hastie, 2005) as implemented in ScikitLearn (Pedregosa et al., 2011). To evaluate our results, we split our data into 10 stratified folds and performed cross-validation on one held-out fold at a time. For all our methods we tune the parameters of our models on a separate validation fold. The overall performance is assessed using Pearson correlation between the set of predicted values and the user-reported score. Results are presented in Table 3.

The same patterns hold when evaluating the results with Root Mean Squared Error (RMSE).

Features	# Feat.	Political Leaning	Political Engagement
Unigrams	6060	.294	.165
LIWC	73	.286	.149
Word2Vec Clusters	500	*.300*	*.169*
Emotions	8	.145	.079
Political Terms	234	.256	*.169*
All (Ensemble)	5	**.369**	**.196**

Table 3: Pearson correlations between the predictions and self-reported ideologies using linear regression with each feature category and a linear combination of their predictions in a 10-fold cross-validation setup. Political leaning is represented on the 1–7 scale removing the moderates (4). Political engagement is a scale ranging from 4 through 3–5 and 2–6 to 1–7.

The results show that both dimensions can be predicted well above chance, with political leaning being easier to predict than engagement. Word2Vec clusters obtain the highest predictive accuracy for political leaning, even though they did not perform as well in the previous classification tasks. For political engagement, political terms and Word2Vec clusters obtain similar predictive accuracy. This result is expected based on the results from Figure 3, which showed how political term usage varies across groups, and how it is especially dependent on political engagement. While political terms are very effective at distinguishing between two opposing political groups, they can not discriminate as well between levels of engagement within the same ideological orientation. Combining all classifiers' predictions in a linear ensemble obtains best results when compared to each individual category.

6.3 Encoding Class Structure

In our previous experiments, we uncovered that certain relationships exist between the seven groups. For example, extreme conservatives and liberals both demonstrate strong political engagement. Therefore, this class structure can be exploited to improve classification performance. To this end, we deploy the sparse graph regularized approach (Argyriou et al., 2007; Zhou et al., 2011) to encode the structure of the seven classes as a graph regularizer in a logistic regression framework.

In particular, we employed a multi-task learning paradigm, where each task is a one-vs-all classification. Multi-task learning (MTL) is a learning paradigm that jointly learns multiple related tasks and can achieve better generalization performance than learning each task individually, especially when presented with insufficient training samples (Liu et al., 2015, 2016b,d). The group structure is encoded into a matrix $\mathbf{R}$ which codes the groups which are considered similar. The objective of the sparse graph regularized multi-task learning problem is:

$$\min_{\mathbf{W},\mathbf{c}} \sum_{t=1}^{\tau} \sum_{i=1}^{N} log(1 + exp(-\mathbf{Y}_{t,i}(\mathbf{W}_{i,t}^T \mathbf{X}_{t,i} + c_t))) + \gamma \|\mathbf{WR}\|_F^2 + \lambda \|\mathbf{W}\|_1,$$

where τ is the number of tasks, $|N|$ the number of samples, $\mathbf{X}$ the feature matrix, $\mathbf{Y}$ the outcome matrix, $\mathbf{W}_{i,t}$ and c_t is the model for task t and $\mathbf{R}$ is the structure matrix.

We define three R matrices: (1) codes that groups with similar political engagement are similar (i.e. 1–7, 2–6, 3–5); (2) codes that groups from each ideological side are similar (i.e. 1–2, 1–3, 2–3, 5–6, 5–7, 6–7); (3) learnt from the data. Results are presented in Table 4. Regular logistic regression performs slightly better than the majority class baseline, which demonstrates that the 7-class classification is a very hard problem although most miss-classifications are within one ideology point. The graph regularization (GR) improves the classification performance over logistic regression (LR) in all cases, with political leaning based matrix (GR–Leaning) obtaining 2% in accuracy higher than the political engagement one (GR–Engagement) and the learnt matrix (GR–Learnt) obtaining best results.

Method	Accuracy
Baseline	19.6%
LR	22.2%
GR–Engagement	24.2%
GR–Leaning	26.2%
GR–Learnt	**27.6%**

Table 4: Experimental results for seven-way classification using multi-task learning (GR–Engagement, GR–Leaning, GR-Learnt) and 500 Word2Vec clusters as features.

7 Conclusions

This study analyzed user-level political ideology through Twitter posts. In contrast to previous work, we made use of a novel data set where fine-grained user political ideology labels are obtained through surveys as opposed to binary self-reports. We showed that users in our data set are far less

likely to post about politics and real-world fine-grained political ideology prediction is harder and more nuanced than previously reported. We analyzed language differences between the ideological groups and uncovered a dimension of political engagement separate from political leaning.

Our work has implications for pollsters or marketers, who are most interested to identify and persuade moderate users. With respect to political conclusions, researchers commonly conceptualize ideology as a single, left-right dimension similar to what we observe in the U.S. Congress (Ansolabehere et al., 2008; Bafumi and Herron, 2010). Our results suggest a different direction: self-reported political extremity is more an indication of political engagement than of ideological self-placement (Abramowitz, 2010). In fact, only self-reported extremists appear to devote much of their Twitter activity to politics at all.

While our study focused solely on text posted by the user, follow-up work can use other modalities such as images or social network analysis to improve prediction performance. In addition, our work on user-level modeling can be integrated with work on message-level political bias to study how this is revealed across users with various levels of engagement. Another direction of future study will look at political ideology prediction in other countries and cultures, where ideology has different or multiple dimensions.

Acknowledgments

The authors acknowledge the support of the Templeton Religion Trust, grant TRT-0048. We wish to thank Prof. David S. Rosenblum for supporting the research visit of Ye Liu.

References

Alan I Abramowitz. 2010. *The Disappearing Center: Engaged Citizens, Polarization, and American Democracy*. Yale University Press.

Firoj Alam and Giuseppe Riccardi. 2014. Predicting Personality Traits using Multimodal Information. In *Workshop on Computational Personality Recognition (WCPR)*. MM, pages 15–18.

Stephen Ansolabehere, Jonathan Rodden, and James M Snyder. 2008. The strength of issues: Using multiple measures to gauge preference stability, ideological constraint, and issue voting. *American Political Science Review* 102(02):215–232.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2007. Multi-task Feature Learning. In *Advances in Neural Information Processing Systems*. NIPS, pages 41–49.

Joseph Bafumi and Michael C Herron. 2010. Leapfrog Representation and Extremism: A Study of American Voters and their Members in Congress. *American Political Science Review* 104(03):519–542.

Pablo BarberASa. 2015. Birds of the Same Feather Tweet Together: Bayesian Ideal Point Estimation using Twitter Data. *Political Analysis* 23(1):76–91.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3:993–1022.

David E Broockman. 2016. Approaches to Studying Policy Representation. *Legislative Studies Quarterly* 41(1):181–215.

D. John Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating Gender on Twitter. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 1301–1309.

Jordan Carpenter, Daniel Preoţiuc-Pietro, Lucie Flekova, Salvatore Giorgi, Courtney Hagan, Margaret Kern, Anneke Buffone, Lyle Ungar, and Martin Seligman. 2016. Real Men don't say 'Cute': Using Automatic Language Analysis to Isolate Inaccurate Aspects of Stereotypes. *Social Psychological and Personality Science* .

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you Tweet: A Content-Based Approach to Geo-Locating Twitter Users. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*. CIKM, pages 759–768.

Raviv Cohen and Derek Ruths. 2013. Classifying Political Orientation on Twitter: It's Not Easy! In *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*. ICWSM, pages 91–99.

Michael D Conover, Bruno Gonçalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. 2011. Predicting the Political Alignment of Twitter Users. In *IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and the IEEE Third Inernational Conference on Social Computing (SocialCom)*. pages 192–199.

Philip E Converse. 1964. The Nature of Belief Systems in Mass Publics. In David Apter, editor, *Ideology and Discontent*, Free Press, New York.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. ACL, pages 256–263.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* 41(6):391–407.

Stephen J Dollinger. 2007. Creativity and Conservatism. *Personality and Individual Differences* 43(5):1025–1035.

Paul Ekman. 1992. An Argument for Basic Emotions. *Cognition & Emotion* 6(3-4):169–200.

Christopher Ellis and James A Stimson. 2012. *Ideology in America*. Cambridge University Press.

Morris P Fiorina. 1999. Extreme Voices: A Dark Side of Civic Engagement. In Morris P. Fiorina and Theda Skocpol, editors, *Civic engagement in American democracy*, Washington, DC: Brookings Institution Press, pages 405–413.

Lucie Flekova, Jordan Carpenter, Salvatore Giorgi, Lyle Ungar, and Daniel Preoţiuc-Pietro. 2016a. Analyzing Biases in Human Perception of User Age and Gender from Text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. ACL, pages 843–854.

Lucie Flekova, Lyle Ungar, and Daniel Preoctiuc-Pietro. 2016b. Exploring Stylistic Variation with Age and Income on Twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. ACL, pages 313–319.

Andrew Gelman. 2009. *Red State, Blue State, Rich State, Poor State: Why Americans Vote the Way they Do*. Princeton University Press.

Alan S Gerber, Gregory A Huber, David Doherty, Conor M Dowling, and Shang E Ha. 2010. Personality and Political Attitudes: Relationships across Issue Domains and Political Contexts. *American Political Science Review* 104(01):111–133.

Jeffrey A Hall, Natalie Pennington, and Allyn Lueders. 2014. Impression Management and Formation on Facebook: A Lens Model Approach. *New Media & Society* 16(6):958–982.

Z. Harris. 1954. Distributional Structure. *Word* 10(23):146 – 162.

Eitan D Hersh. 2015. *Hacking the Electorate: How Campaigns Perceive Voters*. Cambridge University Press.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political Ideology Detection using Recursive Neural Networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. ACL, pages 1113–1122.

Cindy D Kam, Jennifer R Wilking, and Elizabeth J Zechmeister. 2007. Beyond the Narrow Data base: Another Convenience Sample for Experimental Research. *Political Behavior* 29(4):415–440.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private Traits and Attributes are Predictable from Digital Records of Human Behavior. *PNAS* 110(15):5802–5805.

George Lakoff. 1997. *Moral Politics: What Conservatives Know that Liberals Don't*. University of Chicago Press.

Vasileios Lampos, Nikolaos Aletras, Daniel Preoţiuc-Pietro, and Trevor Cohn. 2014. Predicting and Characterising User Impact on Twitter. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. EACL, pages 405–413.

Randall A Lewis and David H Reiley. 2014. Online Ads and Offline Sales: Measuring the Effect of Retail Advertising via a Controlled Experiment on Yahoo! *Quantitative Marketing and Economics* 12(3):235–266.

Leqi Liu, Daniel Preoţiuc-Pietro, Zahra Riahi Samani, Mohsen E. Moghaddam, and Lyle Ungar. 2016a. Analyzing Personality through Social Media Profile Picture Choice. In *Proceedings of the Tenth International AAAI Conference on Weblogs and Social Media*. ICWSM, pages 211–220.

Ye Liu, Liqiang Nie, Lei Han, Luming Zhang, and David S Rosenblum. 2015. Action2Activity: Recognizing Complex Activities from Sensor Data. In *Proceedings of the International Joint Conference on Artificial Intelligence*. IJCAI, pages 1617–1623.

Ye Liu, Liqiang Nie, Li Liu, and David S Rosenblum. 2016b. From Action to Activity: Sensor-based Activity Recognition. *Neurocomputing* 181:108–115.

Ye Liu, Luming Zhang, Liqiang Nie, Yan Yan, and David S Rosenblum. 2016c. Fortune Teller: Predicting your Career Path. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI, pages 201–207.

Ye Liu, Yu Zheng, Yuxuan Liang, Shuming Liu, and David S. Rosenblum. 2016d. Urban Water Quality Prediction Based on Multi-task Multi-view Learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*. IJCAI, pages 2576–2582.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*. NIPS, pages 3111–3119.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013b. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2010 annual Conference of the North American Chapter of the Association for Computational Linguistics*. NAACL, pages 746–751.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon. In *Proceedings of the Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*. NAACL, pages 26–34.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence* 29(3):436–465.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin' Words: Lexical Feature Selection and Evaluation for Identifying the Content of Political Conflict. *Political Analysis* 16(4):372–403.

Jaime L Napier and John T Jost. 2008. Why are Conservatives Happier than Liberals? *Psychological Science* 19(6):565–572.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine Learning in Python. *JMLR* 12.

Marco Pennacchiotti and Ana-Maria Popescu. 2011. A Machine Learning Approach to Twitter User Classification. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*. ICWSM, pages 281–288.

James W. Pennebaker, Martha E. Francis, and Roger J. Booth. 2001. *Linguistic Inquiry and Word Count*. Mahway: Lawrence Erlbaum Associates.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 1532–1543.

Bryan Perozzi and Steven Skiena. 2015. Exact Age Prediction in Social Networks. In *Proceedings of the 24th International Conference on World Wide Web*. WWW, pages 91–92.

Daniel Preoţiuc-Pietro, Jordan Carpenter, Salvatore Giorgi, and Lyle Ungar. 2016. Studying the Dark Triad of Personality using Twitter Behavior. In *Proceedings of the 25th ACM Conference on Information and Knowledge Management*. CIKM, pages 761–770.

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015a. An Analysis of the User Occupational Class through Twitter Content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. ACL, pages 1754–1764.

Daniel Preoţiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015b. Studying User Income through Language, Behaviour and Affect in Social Media. *PLoS ONE* .

Anna Priante, Djoerd Hiemstra, Tijs van den Broek, Aaqib Saeed, Michel Ehrenhard, and Ariana Need. 2016. #WhoAmI in 160 Characters? Classifying Social Identities Based on Twitter. In *Proceedings of the Workshop on Natural Language Processing and Computational Social Science*. EMNLP, pages 55–65.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying Latent User Attributes in Twitter. In *Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents*. SMUC, pages 37–44.

Dominic Rout, Daniel Preoţiuc-Pietro, Bontcheva Kalina, and Trevor Cohn. 2013. Where's @wally: A Classification Approach to Geolocating Users based on their Social Ties. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*. HT, pages 11–20.

Maarten Sap, Gregory Park, Johannes C. Eichstaedt, Margaret L. Kern, David J. Stillwell, Michal Kosinski, Lyle H. Ungar, and Hansen Andrew Schwartz. 2014. Developing Age and Gender Predictive Lexica over Social Media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 1146–1151.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, and Martin EP Seligman. 2013. Personality, Gender, and Age in the Language of Social Media: The Open-vocabulary Approach. *PloS ONE* 8(9).

Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *Transactions on Pattern Analysis and Machine Intelligence* 22(8):888–905.

Carlo Strapparava and Rada Mihalcea. 2008. Learning to Identify Emotions in Text. In *Proceedings of the 2008 ACM Symposium on Applied Computing*. pages 1556–1560.

Carlo Strapparava, Alessandro Valitutti, et al. 2004. WordNet Affect: an Affective Extension of WordNet. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*. volume 4 of *LREC*, pages 1083–1086.

Ramanathan Subramanian, Yan Yan, Jacopo Staiano, Oswald Lanz, and Nicu Sebe. 2013. On the Relationship between Head Pose, Social Attention and Personality Prediction for Unstructured and Dynamic Group Interactions. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*. ICMI, pages 3–10.

Karolina Sylwester and Matthew Purver. 2015. Twitter Language Use Reflects Psychological Differences between Democrats and Republicans. *PLoS ONE* 10(9).

Brandon Van Der Heide, Jonathan D D'Angelo, and Erin M Schumaker. 2012. The Effects of Verbal versus Photographic Self-presentation on Impression Formation in Facebook. *Journal of Communication* 62(1):98–116.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring User Political Preferences from Streaming Communications. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. ACL, pages 186–196.

Ulrike von Luxburg. 2007. A Tutorial on Spectral Clustering. *Statistics and Computing* 17(4):395–416.

Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Your Cart tells You: Inferring Demographic Attributes from Purchase Data. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. WSDM, pages 173–182.

Ingmar Weber, Venkata Rama Kiran Garimella, and Asmelash Teka. 2013. Political Hashtag Trends. In *European Conference on Information Retrieval*. ECIR, pages 857–860.

Tae Yano, Philip Resnik, and Noah A Smith. 2010. Shedding (a Thousand Points of) Light on Biased Language. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. NAACL, pages 152–158.

Muhammad Bilal Zafar, Krishna P Gummadi, and Cristian Danescu-Niculescu-Mizil. 2016. Message Impartiality in Social Media Discussions. In *Proceedings of the Tenth International AAAI Conference on Weblogs and Social Media*. ICWSM, pages 466–475.

Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. 2012. Homophily and Latent Attribute Inference: Inferring Latent Attributes of Twitter Users from Neighbors. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media*. ICWSM, pages 387–390.

Jiayu Zhou, Jianhui Chen, and Jieping Ye. 2011. MALSAR: Multi-Task Learning via Structural Regularization. *Arizona State University* .

Hui Zou and Trevor Hastie. 2005. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society, Series B* .

Leveraging Behavioral and Social Information for Weakly Supervised Collective Classification of Political Discourse on Twitter

Kristen Johnson, Di Jin, Dan Goldwasser
Department of Computer Science
Purdue University, West Lafayette, IN 47907
{john1187, jind, dgoldwas}@purdue.edu

Abstract

Framing is a political strategy in which politicians carefully word their statements in order to control public perception of issues. Previous works exploring political framing typically analyze frame usage in longer texts, such as congressional speeches. We present a collection of weakly supervised models which harness collective classification to predict the frames used in political discourse on the microblogging platform, Twitter. Our global probabilistic models show that by combining both lexical features of tweets and network-based behavioral features of Twitter, we are able to increase the average, unsupervised F_1 score by 21.52 points over a lexical baseline alone.

1 Introduction

The importance of understanding political discourse on social media platforms is becoming increasingly clear. In recent U.S. presidential elections, Twitter was widely used by all candidates to promote their agenda, interact with supporters, and attack their opponents. Social interactions on such platforms allow politicians to quickly react to current events and gauge interest in and support for their actions. These dynamic settings emphasize the importance of constructing automated tools for analyzing this content. However, these same dynamics make constructing such tools difficult, as the language used to discuss new events and political agendas continuously changes. Consequently, the rich social interactions on Twitter can be leveraged to help support such analysis by providing alternatives to direct supervision.

In this paper we focus on political framing, a very nuanced political discourse analysis task, on a variety of issues frequently discussed on Twitter. Framing (Entman, 1993; Chong and Druckman, 2007) is employed by politicians to bias the discussion towards their stance by emphasizing specific aspects of the issue. For example, the debate around increasing the minimum wage can be framed as a *quality of life* issue or as an *economic* issue. While the first frame supports increasing minimum wage because it improves workers' lives, the second frame, by conversely emphasizing the costs involved, opposes the increase. Using framing to analyze political discourse has gathered significant interest over the last few years (Tsur et al., 2015; Card et al., 2015; Baumer et al., 2015) as a way to automatically analyze political discourse in congressional speeches and political news articles. Different from previous works which focus on these longer texts or single issues, our dataset includes tweets authored by all members of the U.S. Congress from both parties, dealing with several policy issues (e.g., immigration, ACA, etc.). These tweets were annotated by adapting the annotation guidelines developed by Boydstun et al. (2014) for Twitter.

Twitter issue framing is a challenging multilabel prediction task. Each tweet can be labeled as using one or more frames, out of 17 possibilities, while only providing 140 characters as input to the classifier. The main contribution of this work is to evaluate whether the *social and behavioral information* available on Twitter is sufficient for constructing a reliable classifier for this task. We approach this framing prediction task using a weakly supervised collective classification approach which leverages the dependencies between tweet frame predictions based on the interactions between their authors.

These dependencies are modeled by connecting Twitter users who have social connections or behavioral similarities. Social connections are di-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 741–752
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1069

rected dependencies that represent the followers of each user as well as retweeting behavior (i.e., user A retweets user B's content). Interestingly, such social connections capture the flow of influence within political parties; however, the number of connections that cross party lines is extremely low. Instead, we rely on capturing behavioral similarity between users to provide this information. For example, users whose Twitter activity peaks at similar times tend to discuss issues in similar ways, providing indicators of their frame usage for those issues. In addition to using social and behavioral information, our approach also incorporates each politician's party affiliation and the frequent phrases (e.g., bigrams and trigrams) used by politicians on Twitter.

These lexical, social, and behavioral features are extracted from tweets via weakly supervised models and then declaratively compiled into a graphical model using Probabilistic Soft Logic (PSL), a recently introduced probabilistic modeling framework.[1] As described in Section 4, PSL specifies high level rules over a relational representation of these features. These rules are then compiled into a graphical model called a hinge-loss Markov random field (Bach et al., 2013), which is used to make the frame prediction. Instead of direct supervision we take a bootstrapping approach by providing a small seed set of keywords adapted from Boydstun et al. (2014), for each frame.

Our experiments show that modeling social and behavioral connections improves F_1 prediction scores in both supervised and unsupervised settings, with double the increase in the latter. We apply our unsupervised model to our entire dataset of tweets to analyze framing patterns over time by both party and individual politicians. Our analysis provides insight into the usage of framing for identification of *aisle-crossing politicians*, i.e., those politicians who vote against their party.

2 Related Work

Issue framing is related to the broader challenges of biased language analysis (Recasens et al., 2013; Choi et al., 2012; Greene and Resnik, 2009) and subjectivity (Wiebe et al., 2004). Several previous works have explored framing in public statements, congressional speeches, and news articles (Fulgoni et al., 2016; Tsur et al., 2015; Card et al., 2015; Baumer et al., 2015). Our approach builds upon the previous work on frame analysis of Boydstun et al. (2014), by adapting and applying their annotation guidelines for Twitter.

In recent years there has been growing interest in analyzing political discourse. Most previous work focuses on opinion mining and stance prediction (Sridhar et al., 2015; Hasan and Ng, 2014; Abu-Jbara et al., 2013; Walker et al., 2012; Abbott et al., 2011; Somasundaran and Wiebe, 2010, 2009). Analyzing political tweets has also attracted considerable interest: a recent SemEval task looked into stance prediction,[2] and more related to our work, Tan et al. (2014) have shown how wording choices can affect message propagation on Twitter. Two recent works look into predicting stance (at user and tweet levels respectively) on Twitter using PSL (Johnson and Goldwasser, 2016; Ebrahimi et al., 2016). Frame classification, however, has a finer granularity than stance classification and describes how someone expresses their view on an issue, not whether they support the issue. Other works focus on identifying and measuring political ideologies (Iyyer et al., 2014; Bamman and Smith, 2015; Sim et al., 2013), policies (Nguyen et al., 2015), and voting patterns (Gerrish and Blei, 2012).

Exploiting social interactions and group structure for prediction has also been explored (Sridhar et al., 2015; Abu-Jbara et al., 2013; West et al., 2014). Works focusing on inferring signed social networks (West et al., 2014), stance classification (Sridhar et al., 2015), social group modeling (Huang et al., 2012), and collective classification using PSL (Bach et al., 2015) are closest to our approach. Unsupervised and weakly supervised models of Twitter data for several various tasks have been suggested, including: profile (Li et al., 2014b) and life event extraction (Li et al., 2014a), conversation modeling (Ritter et al., 2010), and methods for dealing with the unique language used in microblogs (Eisenstein, 2013).

Predicting political affiliation and other characteristics of Twitter users has been explored (Volkova et al., 2015, 2014; Yano et al., 2013; Conover et al., 2011). Others have focused on sentiment analysis (Pla and Hurtado, 2014; Bakliwal et al., 2013), predicting ideology (Djemili et al., 2014), automatic polls

[1] http://psl.cs.umd.edu

[2] http://alt.qcri.org/semeval2016/task6/

based on Twitter sentiment and political forecasting using Twitter (Bermingham and Smeaton, 2011; O'Connor et al., 2010; Tumasjan et al., 2010), as well as distant supervision applications (Marchetti-Bowick and Chambers, 2012).

Several works from political and social science research have studied the role of Twitter and framing in shaping public opinion of certain events, e.g. the Vancouver riots (Burch et al., 2015) and the Egyptian protests (Harlow and Johnson, 2011; Meraz and Papacharissi, 2013). Others have covered framing and sentiment analysis of opponents (Groshek and Al-Rawi, 2013) and network agenda modeling (Vargo et al., 2014) in the 2012 U.S. presidential election. Jang and Hart (2015) studied frames used by the general population specific to global warming. In contrast to these works, we predict the *issue-independent* general frames of tweets, by U.S. politicians, which discuss six different policy issues.

3 Data Collection and Annotation

Data Collection and Preprocessing: We collected 184,914 of the most recent tweets of members of the U.S. Congress (both the House of Representatives and Senate). Using an average of ten keywords per issue, we filtered out tweets not related to the following six issues of interest: (1) limiting or gaining access to abortion, (2) debates concerning the Affordable Care Act (i.e., ACA or Obamacare), (3) the issue of gun rights versus gun control, (4) effects of immigration policies, (5) acts of terrorism, and (6) issues concerning the LGBTQ community. Forty politicians (10 Republicans and 10 Democrats, from both the House and Senate), were chosen randomly for annotation. Table 1 presents the statistics of our congressional tweets dataset, which is available for the community.[3] Appendix A contains more details of our dataset and preprocessing steps.

Data Annotation: Two graduate students were trained in the use of the Policy Frames Codebook developed by Boydstun et al. (2014) for annotating each tweet with a frame. The general aspects of each frame are shown in Table 2. Frames are designed to generalize across issues and overlap of multiple frames is possible. Additionally, the Codebook is typically applied to newspaper articles where discussion of policy can encompass other frames in the text. Consequently, annotators using the Codebook are advised to be careful when assigning Frame 13 to a text.

Based on this guidance and the difficulty of labeling tweets (as discussed in Card et al. (2015)), annotators were instructed to use the following procedure: (1) attempt to assign a primary frame to the tweet if possible, (2) if not possible, assign two frames to the tweet where the first frame is chosen as the more accurate of the two frames, (3) when assigning frames 12 through 17, double check that the tweet cannot be assigned to any other frames. Annotators spent one month labeling the randomly chosen tweets. For all tweets with more than one frame, annotators met to come to a consensus on whether the tweet should have one frame or both. The labeled dataset has an inter-annotator agreement, calculated using Cohen's Kappa statistic, of 73.4%.

Extensions of the Codebook for Twitter Use: The first 14 frames outlined in Table 2 are directly applicable to the tweets of U.S. politicians. In our labeled set, Frame 15 (Other) was never used. Therefore, we drop its analysis from this paper. From our observations, we propose the addition of the 3 frames at the bottom of Table 2 for Twitter analysis: Factual, (Self) Promotion, and Personal Sympathy and Support. Tweets that present a fact, with no detectable political spin or twists, are labeled as having the Factual frame (15). Tweets that discuss a politician's appearances, speeches, statements, or refer to political friends are considered to have the (Self) Promotion frame. Finally, tweets where a politician offers their "thoughts and prayers", condolences, or stands in support of others, are considered to have the Personal frame.

We find that for many tweets, one frame is not enough. This is caused by the compound nature of many tweets, e.g., some tweets are two separate sentences, with each sentence having a different frame or tweets begin with one frame and end with another. A final problem, that may also be relevant to longer text articles, is that of subframes within a larger frame. For example, the tweet *"We must bolster the security of our borders and craft an immigration policy that grows our economy."* has two frames: Security & Defense and Economic. However, both frames could fall under Frame 13 (Policy), if this tweet as a whole was a rebuttal point about an immigration policy. The lack of

[3]The dataset and PSL scripts are available at: http://purduenlp.cs.purdue.edu/projects/twitterframing.

Tweets	By Party		By Issue					
	Rep	Dem	Abortion	ACA	Guns	Immigration	Terrorism	LGBTQ
Entire Dataset	48504	43953	6467	35854	15532	13442	15205	6046
Labeled Subset	894	1156	170	564	543	233	446	183

Table 1: Statistics of Collected Tweets. Rep stands for Republican and Dem for Democrats.

Frame Number, Frame Name, and Brief Description of Frame
1. Economic: *Pertains to the economic impacts of a policy*
2. Capacity & Resources: *Pertains to lack of or availability of resources*
3. Morality & Ethics: *Motivated by religious doctrine, righteousness, sense of responsibility*
4. Fairness & Equality: *Of how laws, punishments, resources, etc. are distributed among groups*
5. Legality, Constitutionality, & Jurisdiction: *Including court cases, restriction and expressions of rights*
6. Crime & Punishment: *Policy violation and consequences*
7. Security & Defense: *Threats or defenses/preemptive actions to protect against threats*
8. Health & Safety: *Includes care access and effectiveness*
9. Quality of Life: *Effects on individual and community life*
10. Cultural Identity: *Culture's norms, trends, customs*
11. Public Sentiment: *Pertains to opinions, polling, and demographics*
12. Political Factors & Implications: *Efforts, stances, filibusters, lobbying, references to other politicians*
13. Policy Description, Prescription, & Evaluation: *Discusses effectiveness of current or proposed policies*
14. External Regulation and Reputation: *Interstate and international relationships of the U.S.*
15. Factual: *Expresses a pure fact, with no detectable political spin*
16. (Self) Promotion: *Promotes another person or the author in some way, e.g. television appearances*
17. Personal Sympathy & Support: *Expresses sympathy, emotional response, or solidarity with others*

Table 2: Frames and Descriptions. The first 14 are Boydstun's frames and the last 3 are our proposed Twitter-specific frames. Boydstun's original Frame 15 (Other) is omitted from this study.

available context for short tweets can make it difficult to determine if a tweet should have one primary frame or is more accurately represented by multiple frames.

4 Global Models of Twitter Language and Activity

Due to the dynamic nature of political discourse on Twitter, our approach is designed to require as little supervision as possible. We implement 6 weakly supervised models which are data-dependent and used to extract and format information from tweets into input for PSL predicates. These predicates are then combined into the probabilistic rules of each model as shown in Table 3. The only sources of supervision these models require includes: unigrams related to the issues, unigrams adapted from the Boydstun et al. (2014) Codebook for frames, and political party of the author of the tweets.

4.1 Global Modeling Using PSL

PSL is a declarative modeling language which can be used to specify weighted, first-order logic rules. These rules are compiled into a hinge-loss Markov random field which defines a probability distribution over possible continuous value assignments to the random variables of the model (Bach et al., 2015).[4] This probability density function is represented as:

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z} \exp\left(-\sum_{r=1}^{M} \lambda_r \phi_r(\mathbf{Y}, \mathbf{X}) \right)$$

where Z is a normalization constant, λ is the weight vector, and

$$\phi_r(\mathbf{Y}, \mathbf{X}) = (\max\{l_r(\mathbf{Y}, \mathbf{X}), 0\})^{\rho_r}$$

is the hinge-loss potential specified by a linear function l_r. The exponent $\rho_r \in 1, 2$ is optional. Each potential represents the instantiation of a rule, which takes the following form:

$$\lambda_1 : P_1(x) \wedge P_2(x, y) \rightarrow P_3(y)$$
$$\lambda_2 : P_1(x) \wedge P_4(x, y) \rightarrow \neg P_3(y)$$

P_1, P_2, P_3, and P_4 are predicates (e.g., political party, issue, frame, and presence of n-grams) and x, y are variables. Each rule has a weight λ which reflects that rule's importance and is learned using the Expectation-Maximization algorithm in our unsupervised experiments. Using concrete constants *a, b* (e.g., tweets and words) which instantiate the variables x, y, model atoms are mapped

[4] Unlike other probabilistic logical models, e.g. MLNs, in which the model's random variables are strictly true or false.

to continuous [0,1] assignments. More important rules (i.e., those with larger weights) are given preference by the model.

4.2 Language Based Models

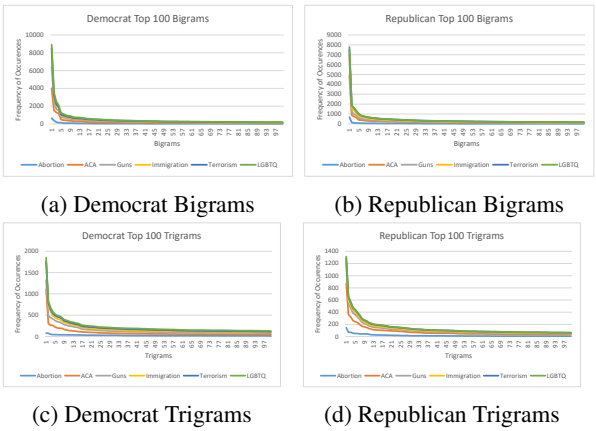
Unigrams: Using the guidelines provided in the Policy Frames Codebook (Boydstun et al., 2014), we adapted a list of expected unigrams for each frame. For example, unigrams that should be related to Frame 12 (Political Factors & Implications) include: filibuster, lobby, Democrats, Republicans. We expect that if a tweet and frame contain a matching unigram, then that frame is likely present in that tweet. The information that tweet T has expected unigram U of frame F is represented with the PSL predicate: UNIGRAM$_F$(T, U). This knowledge is then used as input to PSL Model 1 via the rule: UNIGRAM$_F$(T, U) $\rightarrow$FRAME(T, F) (shown in line 1 of Table 3).

However, not every tweet will have a unigram that matches those in this list. Under the intuition that at least one unigram in a tweet should be *similar* to a unigram in the list, we designed the following *MaxSim* metric to compute the maximum similarity between a word in a tweet and a word from the list of unigrams.

$$\text{MAXSIM(T, F)} = \underset{\text{u}\in F,\text{w}\in T}{\arg\max}\ \text{SIMILARITY(W,U)} \quad (1)$$

T is a tweet, W is each word in T, and U is each unigram in the list of expected unigrams (per frame). SIMILARITY is the computed `word2vec` similarity (using pretrained embeddings) of each word in the tweet with every unigram in the list of unigrams for each frame. The frame F of the maximum scoring unigram is input to the PSL predicate: MAXSIM$_F$(T, F), which indicates that tweet T has the highest similarity to frame F.

Bigrams and Trigrams: In addition to unigrams, we also explored the effects of political party *slogans* on frame prediction. Slogans are common catch phrases or sayings that people typically associate with different U.S. political parties. For example, Republicans are known for using the phrase "repeal and replace" when they discuss the ACA. Similarly, in the 2016 U.S. presidential election, Secretary Hillary Clinton's campaign slogan became "Love Trumps Hate". To visualize slogan usage by parties for different issues, we used the *entire* tweets dataset, including all unlabeled tweets, to extract the top bigrams and trigrams per party for each issue. The histograms in Figure 1 show these distributions for the top 100 bigrams and trigrams. Based on these results, we use the top 20 bigrams (e.g., *women's healthcare* and *immigration reform*) and trigrams (e.g. *prevent gun violence*) as input to PSL predicates BIGRAM$_{IP}$(T, B) and TRIGRAM$_{IP}$(T, TG). These rules represent that tweet T has bigram B or trigram TG from the respective issue I phrase lists of either party P.



(a) Democrat Bigrams (b) Republican Bigrams

(c) Democrat Trigrams (d) Republican Trigrams

Figure 1: Distributions of Bigrams and Trigrams by Party.

4.3 Twitter Behavior Based Models

In addition to language based features of tweets, we also exploit the behavioral and social features of Twitter including similarities between temporal activity and network relationships.

Temporal Similarity: We construct a temporal histogram for each politician which captures their Twitter activity over time. When an event happens politicians are most likely to tweet about that event within hours of its occurrence. Similarly, most politicians tweet about the event most frequently the day of the event and this frequency decreases over time. From these temporal histograms, we observed that the frames used the day of an event were similar and gradually changed over time. For example, once the public is notified of a shooting, politicians respond with Frame 17 to offer sympathy to the victims and their families. Over the next days or weeks, both parties slowly transition to using additional frames, e.g. Democrats use Frame 7 to argue for gun control legislation. To capture this behavior we use the PSL predicate SAMETIME(T1, T2). This indicates that tweet T1 occurs around the same time as tweet

Types of Models	Model Number	Basis of Model	Example of PSL Rules
Language Based	1	Unigrams	Unigram_F(T, U) →Frame(T, F)
	2	Bigrams	Unigram_F(T, U) ∧Bigram_{IP}(T, B) →Frame(T, F)
	3	Trigrams	Unigram_F(T, U) ∧Trigram_{IP}(T, TG) →Frame(T, F)
Behavior Based	4	Temporal Activity	SameTime(T1, T2) ∧Frame(T1, F) →Frame(T2, F)
	5	Retweet Patterns	Retweets(T1, T2) ∧Frame(T1, F) →Frame(T2, F)
	6	Following Network	Follows(T1, T2) ∧Frame(T1, F) →Frame(T2, F)

Table 3: Examples of PSL Model Rules. Each model adds to the rules of the previous model. The full list of rule combinations for each model is available with our dataset.

T2.[5] This information is used in Model 4 via rules such as: SameTime(T1, T2) & Frame(T1, F) →Frame(T2, F), as shown in line 4 of Table 3.

Network Similarity: Finally, we expect that politicians who share ideologies, and thus are likely to frame issues similarly, will retweet and/or follow each other on Twitter. Due to the compound nature of tweets, retweeting with additional comments can add more frames to the original tweet. Additionally, politicians on Twitter are more likely to follow members of their own party or similar non-political entities than those of the opposing party. To capture this network-based behavior we use two PSL predicates: Retweets(T1, T2) and Follows(T1, T2). These predicates indicate that the content of tweet T1 includes a retweet of tweet T2 and that the author of T1 follows the author of T2 on Twitter, respectively. The last two lines of Table 3 show examples of how network similarity is incorporated into PSL rules.

5 Experiments

Evaluation Metrics: Since each tweet can have more than one frame, our prediction task is a multilabel classification task. The precision of a multilabel model is the ratio of how many predicted labels are correct:

$$Precision = \frac{1}{T}\sum_{t=1}^{T}\frac{|Y_t \cap h(x_t)|}{|h(x_t)|} \quad (2)$$

The recall of this model is the ratio of how many of the actual labels were predicted:

$$Recall = \frac{1}{T}\sum_{t=1}^{T}\frac{|Y_t \cap h(x_t)|}{|Y_t|} \quad (3)$$

[5]We conducted experiments with different hour and day limits and found that using a time frame of one hour results in the best accuracy while limiting noise.

In both formulas, T is the number of tweets, Y_t is the true label for tweet *t*, x_t is a tweet example, and $h(x_t)$ are the predicted labels for that tweet. The F_1 score is computed as the harmonic mean of the precision and recall. Additionally, in Tables 4, 5, and 6 the reported average is the micro-weighted average F_1 scores over all frames.

Experimental Settings: We provide an analysis of our PSL models under both supervised and unsupervised settings. In the PSL supervised experiments, we used five-fold cross validation with randomly chosen splits.

Previous works typically use an SVM, with bag-of-words features, which is not used in a multilabel prediction, i.e., each frame is predicted individually. The results of this approach on our dataset are shown in column 2 of Table 4. In this scenario, the SVM tends to prefer the majority class, which results in many incorrect labels. Column 3 shows the results of using an SVM with bag-of-words features to perform multilabel classification. This approach decreases the F_1 score for a majority of frames. Both SVMs also result in F_1 scores of 0 for some frames, further lowering the overall performance. Finally, columns 4 and 5 show the results of using our worst and best PSL models, respectively. PSL Model 1, which uses our adapted unigram features instead of the bag-of-words features for multilabel classification, serves as our baseline to improve upon. Additionally, Model 6 of the supervised, collective network setting represents the best results we can achieve.

We also explore the results of our PSL models in an unsupervised setting because the highly dynamic nature of political discourse on Twitter makes it unrealistic to expect annotated data to generalize to future discussions. The only source of supervision comes from the initial unigrams lists and party information as described in Section 4. The labeled tweets are used for evaluation only. As seen in Table 4, we are able to improve

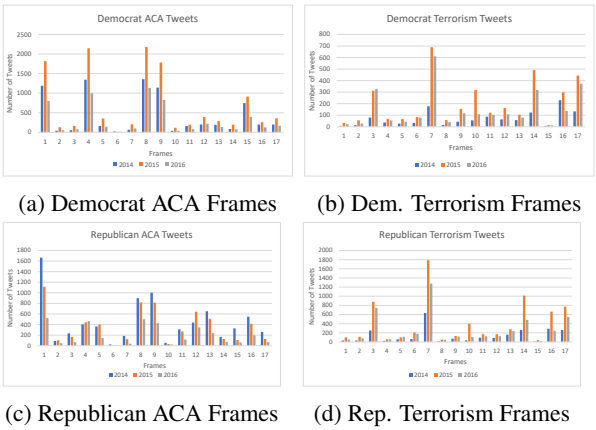
SETTING	SVM INDIV.	SVM MULTI.	PSL M1	PSL M6
SUP.	28.67	18.90	66.02	77.79
UNSUP.	—	—	37.14	58.66

Table 4: Baseline and Skyline Micro-weighted Average F_1 Scores. SVM INDIV. is the SVM trained to predict one frame. SVM MULTI. is the multiclass SVM. PSL M1 is the adapted unigram PSL Model 1. PSL M6 is the collective network.

the best unsupervised model to within an F_1 score of 7.36 points of the unigram baseline of 66.02, and 19.13 points of the best supervised score of 77.79.

Analysis of Supervised Experiments: Table 5 shows the results of our supervised experiments. Here we can see that by adding Twitter behavior (beginning with Model 4), our behavior-based models achieve the best F_1 scores across all frames. Model 4 achieves the highest results on two frames, suggesting retweeting and network follower information do not help improve the prediction score for these frames. Similarly, Model 5 achieves the highest prediction for 5 of the frames, suggesting network follower information cannot further improve the score for these frames. Overall, the Twitter behavior based models are able to outperform language based models alone, including the best performing language model (Model 3) which combines unigrams, bigrams, and trigrams together to collectively infer the correct frames.

Analysis of Unsupervised Experiments: In the unsupervised setting, Model 6, the combination of language and Twitter behavior features achieves the best results on 16 of the 17 issues, as shown in Table 6. There are a few interesting aspects of the unsupervised setting which differ from the supervised setting. Six of the frame predictions do worse in Model 2, which is double that of the supervised version. This is likely due to the presence of overlapping bigrams across frames and issues, e.g., "women's healthcare" could appear in both Frames 4 and 8 and the issues of ACA and abortion. However, all six are able to improve with the addition of trigrams (Model 3), whereas only 1 of 3 frames improves in the supervised setting. This suggests that bigrams may not be as useful as trigrams in an unsupervised setting. Finally, in Model 5, which adds retweet behaviors, we notice that 5 of the frames decrease in F_1 score and 11 of the frames have the same score as the previous model. These results suggest that retweet behaviors are not as useful as the follower network relationships in an unsupervised setting.



(a) Democrat ACA Frames (b) Dem. Terrorism Frames

(c) Republican ACA Frames (d) Rep. Terrorism Frames

Figure 2: Predicted Frames for Tweets from 2014 to 2016 by Party for ACA and Terrorism Issues.

6 Qualitative Analysis

To explore the usefulness of frame identification in political discourse analysis, we apply our best performing model (Model 6) on the *unlabeled* dataset to determine framing patterns over time, both by party and individual. Figure 2 shows the results of our frame analysis for both parties over time for two issues: ACA and terrorism.[6] We compiled the predicted frames for tweets from 2014 to 2016 for each party. Figure 3 presents the results of frame prediction for 2015 tweets of aisle-crossing individual politicians for these two issues.

Party Frames: From Figure 2(a) we can see that Democrats mainly use Frames 1, 4, 8, 9, and 15 to discuss ACA, while Figure 2(c) shows that Republicans predominantly use Frames 1, 8, 9, 12, and 13. Though the parties use similar frames, they are used to express different agendas. For example, Democrats use Frame 8 to indicate the positive effect that the ACA has had in granting more Americans health care access. Republicans, however, use Frame 8 (and Frame 13) to indicate their party's agenda to replace the ACA with access to different options for health care. Additionally, Democrats use the Fairness & Equality Frame (Frame 4) to convey that the ACA gives minority groups a better chance at accessing health care.

[6] Due to space, we omit the other 4 issues. These 2 were chosen because they are among the most frequently discussed issues in our dataset.

Frame Number	Frame	Results of Supervised PSL Model Frame Predictions					
		Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
1	Economic	85.19	85.19	86.73	87.72	87.72	**89.88**
2	Capacity & Resources	55.38	61.54	76.71	77.11	77.11	**79.55**
3	Morality	73.39	80.52	86.95	87.5	**87.43**	**87.43**
4	Fairness	63.56	67.83	65.19	69.91	79.53	**82.35**
5	Legality	80.41	80.78	80.79	**83.33**	81.79	82.16
6	Crime	54.55	54.55	66.67	**76.92**	**76.92**	**76.92**
7	Security	84.40	82.14	84.10	86.67	86.67	**88.48**
8	Health	73.50	75.76	75.59	77.46	**79.71**	**79.71**
9	Quality of Life	69.39	68.00	69.39	72.34	72.34	**82.93**
10	Cultural	75.86	78.57	81.25	81.25	81.25	**85.71**
11	Public Sentiment	12.25	15.25	24.62	24.24	26.24	**29.41**
12	Political	54.21	63.31	74.33	74.42	**74.52**	**74.52**
13	Policy	55.75	58.87	60.25	61.54	64.06	**65.06**
14	External Regulation	60.71	59.15	64.71	74.35	74.35	**85.71**
15	Factual	66.56	68.00	71.43	81.82	80.82	**82.85**
16	(Self) Promotion	85.71	86.46	86.58	87.34	87.33	**91.76**
17	Personal	71.79	71.71	74.73	75.00	**77.55**	**77.55**
	Weighted Average	66.02	68.78	72.49	74.40	75.71	**77.79**

Table 5: F_1 Scores of Supervised PSL Models. The highest prediction per frame is marked in bold.

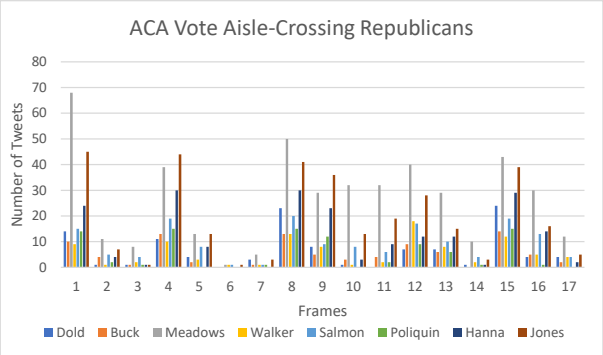
Frame Number	Frame	Results of Unsupervised PSL Model Frame Predictions					
		Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
1	Economic	31.82	31.52	69.57	72.22	72.22	**73.23**
2	Capacity & Resources	23.38	28.51	40.00	**41.18**	**41.18**	**41.18**
3	Morality	28.63	29.41	47.67	53.98	43.06	**53.99**
4	Fairness	33.49	47.19	59.15	63.50	63.50	**64.74**
5	Legality	44.58	46.93	58.02	60.64	60.63	**64.54**
6	Crime	7.89	7.62	73.33	75.00	75.00	**76.92**
7	Security	42.50	40.24	51.83	62.09	61.68	**64.09**
8	Health	48.36	48.79	79.43	86.49	86.49	**86.67**
9	Quality of Life	17.82	21.99	48.89	52.63	52.63	**54.35**
10	Cultural	15.38	15.67	51.22	52.63	52.63	**55.56**
11	Public Sentiment	15.22	15.72	50.79	53.97	41.03	**54.69**
12	Political	49.06	48.20	50.29	46.99	46.99	**47.23**
13	Policy	39.88	39.39	37.02	42.77	42.77	**43.79**
14	External Regulation	12.66	14.22	44.44	66.67	66.67	**71.43**
15	Factual	24.64	19.21	70.95	70.37	70.41	**78.95**
16	(Self) Promotion	40.11	46.41	48.16	50.96	50.96	**52.89**
17	Personal	45.36	46.15	59.66	62.99	62.13	**71.20**
	Weighted Average	37.14	38.79	53.13	56.49	55.54	**58.66**

Table 6: F_1 Scores of Unsupervised PSL Models. The highest prediction per frame is marked in bold.

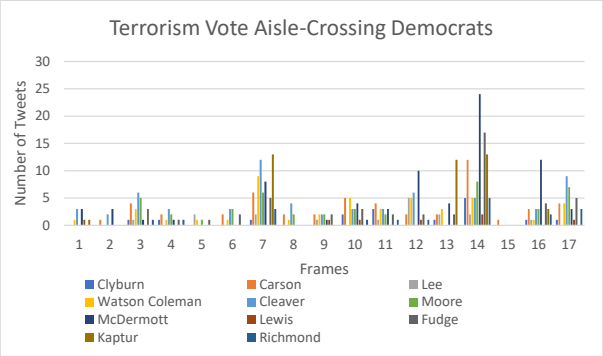
They also use Frame 15 to express statistics about enrollment of Americans under the ACA. Finally, Republicans use Frames 12 and 13 to bring attention to their own party's actions to "repeal and replace" the ACA with different policies.

Figures 2(b) and 2(d) show the party-based framing patterns over time for terrorism related tweets. For this issue both parties use similar frames: 3, 7, 10, 14, 16, and 17, but to express different views. For example, Democrats use Frame 3 to indicate a moral responsibility to fight ISIS. Republicans use Frame 3 to frame terrorists or their attacks as a result of "radical Islam". An interesting pattern to note is seen in Frames 10 and 14 for both parties. In 2015 there is a large increase in the usage of this frame. This seems to indicate that parties possibly adopt new frames *simultaneously or in response to the opposing party*, perhaps in an effort to be in control of the way the message is delivered through that frame.

Individual Frames: In addition to entire party analysis, we were interested in seeing if frames could shed light on the behavior of *aisle-crossing* politicians. These are politicians who do not vote the same as the majority vote of their party (i.e., they vote the same as the opposing party). Identifying such politicians can be useful in governments which are heavily split by party, i.e., governments such as the recent U.S. Congress (2015 to 2017), where politicians tend to vote the same



(a) Aisle-Crossing Republicans on ACA Votes.



(b) Aisle-Crossing Democrats on Terrorism Votes.

Figure 3: Predicted Frames for Tweets of Aisle-Crossing Politicians in 2015.

as the rest of their party members. For this analysis, we collected five 2015 votes from the House of Representatives on both issues and compiled a list of the politicians who voted opposite to their party. The most important descriptor we noticed was that all aisle-crossing politicians *tweet less frequently on the issue* than their fellow party members. This is true for both parties. This behavior could indicate lack of desire to draw attention to one's stance on the particular issue.

Figure 3(a) shows the framing patterns of aisle-crossing Republicans on ACA votes from 2015. Recall from Figure 2 that Democrats mostly use Frames 1, 4, 8, 9, and 15, while Republicans mainly use Frames 1, 8, and 9. In this example, these Republicans are considered aisle-crossing votes because they have voted the same as Democrats on this issue. The most interesting pattern to note here is that these Republicans use the same framing patterns as the Republicans (Frames 1, 8, and 9), but they also use the frames that are *unique to Democrats*: Frames 4 and 15. These latter two frames appear significantly less in the Republican tweets of our entire dataset as well. These results suggest that to predict aisle-crossing Republicans it would be useful to check for usage of typically Democrat-associated frames, especially if those frames are infrequently used by Republicans.

Figure 3(b) shows the predicted frames for aisle-crossing Democrats on terrorism-related votes. We see here that there are very few tweets from these Democrats on this issue and that overall they use the same framing patterns as seen previously: Frames 3, 7, 10, 14, 16, and 17. However, given the small scale of these tweets, we can also consider Frames 12 and 13 to show peaks for this example. This suggests that for aisle-crossing Democrats the use of additional frames not often used by their party for discussing an issue might indicate potentially different voting behaviors.

7 Conclusion

In this paper we present the task of collective classification of Twitter data for framing prediction. We show that by incorporating Twitter behaviors such as similar activity times and similar networks, we can increase F_1 score prediction. We provide an analysis of our approach in both supervised and unsupervised settings, as well as a real world analysis of framing patterns over time. Finally, our global PSL models can be applied to other domains, such as politics in other countries, simply by changing the initial unigram keywords to reflect the politics of those countries.

Acknowledgments

We thank the anonymous reviewers for their thoughtful comments and suggestions.

References

Rob Abbott, Marilyn Walker, Pranav Anand, Jean E. Fox Tree, Robeson Bowmani, and Joseph King. 2011. How can you say such things?!?: Recognizing disagreement in informal political argument. In *Proc. of the Workshop on Language in Social Media*.

Amjad Abu-Jbara, Ben King, Mona Diab, and Dragomir Radev. 2013. Identifying opinion subgroups in arabic online discussions. In *Proc. of ACL*.

Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2015. Hinge-loss markov random fields and probabilistic soft logic. *arXiv preprint arXiv:1505.04406* .

Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. 2013. Hinge-loss Markov random fields:

Convex inference for structured prediction. In *Proc. of UAI*.

Akshat Bakliwal, Jennifer Foster, Jennifer van der Puil, Ron O'Brien, Lamia Tounsi, and Mark Hughes. 2013. Sentiment analysis of political tweets: Towards an accurate classifier. In *Proc. of ACL*.

David Bamman and Noah A Smith. 2015. Open extraction of fine-grained political statements. In *Proc. of EMNLP*.

Eric Baumer, Elisha Elovic, Ying Qin, Francesca Polletta, and Geri Gay. 2015. Testing and comparing computational approaches for identifying the language of framing in political news. In *Proc. of NAACL*.

Adam Bermingham and Alan F Smeaton. 2011. On using twitter to monitor political sentiment and predict election results .

Amber Boydstun, Dallas Card, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2014. Tracking the development of media frames within and across policy issues.

Lauren M. Burch, Evan L. Frederick, and Ann Pegoraro. 2015. Kissing in the carnage: An examination of framing on twitter during the vancouver riots. *Journal of Broadcasting & Electronic Media* 59(3):399–415.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proc. of ACL*.

Eunsol Choi, Chenhao Tan, Lillian Lee, Cristian Danescu-Niculescu-Mizil, and Jennifer Spindel. 2012. Hedge detection as a lens on framing in the gmo debates: A position paper. In *Proc. of ACL Workshops*.

Dennis Chong and James N Druckman. 2007. Framing theory. *Annu. Rev. Polit. Sci.* 10:103–126.

Michael D Conover, Bruno Gonçalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. 2011. Predicting the political alignment of twitter users. In *Proc. of PASSAT*.

Sarah Djemili, Julien Longhi, Claudia Marinica, Dimitris Kotzinos, and Georges-Elia Sarfati. 2014. What does twitter have to say about ideology? In *NLP 4 CMC*.

Javid Ebrahimi, Dejing Dou, and Daniel Lowd. 2016. Weakly supervised tweet stance classification by relational bootstrapping. In *Proc. of EMNLP*.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proc. of NAACL*.

Robert M Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of communication* 43(4):51–58.

Dean Fulgoni, Jordan Carpenter, Lyle Ungar, and Daniel Preotiuc-Pietro. 2016. An empirical exploration of moral foundations theory in partisan news sources. In *Proc. of LREC*.

Sean Gerrish and David M Blei. 2012. How they vote: Issue-adjusted models of legislative behavior. In *Advances in Neural Information Processing Systems*. pages 2753–2761.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Proc. of NAACL*.

Jacob Groshek and Ahmed Al-Rawi. 2013. Public sentiment and critical framing in social media content during the 2012 u.s. presidential campaign. *Social Science Computer Review* 31(5):563–576.

Summer Harlow and Thomas Johnson. 2011. The arab spring— overthrowing the protest paradigm? how the new york times, global voices and twitter covered the egyptian revolution. *International Journal of Communication* 5(0).

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *Proc. of EMNLP*.

Bert Huang, Stephen H. Bach, Eric Norris, Jay Pujara, and Lise Getoor. 2012. Social group modeling with probabilistic soft logic. In *NIPS Workshops*.

Iyyer, Enns, Boyd-Graber, and Resnik. 2014. Political ideology detection using recursive neural networks. In *Proc. of ACL*.

S. Mo Jang and P. Sol Hart. 2015. Polarized frames on "climate change" and "global warming" across countries and states: Evidence from twitter big data. *Global Environmental Change* 32:11–17.

Kristen Johnson and Dan Goldwasser. 2016. All i know about politics is what i read in twitter: Weakly supervised models for extracting politicians' stances from twitter. In *Proc. of COLING*.

Jiwei Li, Alan Ritter, Claire Cardie, and Eduard H Hovy. 2014a. Major life event extraction from twitter based on congratulations/condolences speech acts. In *Proc. of EMNLP*.

Jiwei Li, Alan Ritter, and Eduard H Hovy. 2014b. Weakly supervised user profile extraction from twitter. In *Proc. of ACL*.

Micol Marchetti-Bowick and Nathanael Chambers. 2012. Learning for microblogs with distant supervision: Political forecasting with twitter. In *Proc. of EACL*.

Sharon Meraz and Zizi Papacharissi. 2013. Networked gatekeeping and networked framing on #egypt. *The International Journal of Press/Politics* 18(2):138–166.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, and Kristina Miler. 2015. Tea party in the house: A hierarchical ideal point topic model and its application to republican legislators in the 112th congress. In *Proc. of ACL*.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proc. of ICWSM*.

Ferran Pla and Lluís F Hurtado. 2014. Political tendency identification in twitter using sentiment analysis techniques. In *Proc. of COLING*.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proc. of ACL*.

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

Sim, Acree, Gross, and Smith. 2013. Measuring ideological proportions in political speeches. In *Proc. of EMNLP*.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proc. of ACL*.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proc. of NAACL Workshops*.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proc. of ACL*.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on twitter. In *Proc. of ACL*.

Oren Tsur, Dan Calacci, and David Lazer. 2015. A frame of mind: Using statistical models for detection of framing and agenda setting campaigns. In *Proc. of ACL*.

Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proc. of ICWSM*.

Chris J. Vargo, Lei Guo, Maxwell McCombs, and Donald L. Shaw. 2014. Network issue agendas on twitter during the 2012 u.s. presidential election. *Journal of Communication* 64(2):296–316.

Svitlana Volkova, Yoram Bachrach, Michael Armstrong, and Vijay Sharma. 2015. Inferring latent user properties from texts published in social media. In *Proc. of AAAI*.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring user political preferences from streaming communications. In *Proc. of ACL*.

Marilyn A. Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proc. of NAACL*.

Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. 2014. Exploiting social network structure for person-to-person sentiment analysis. *TACL* .

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational linguistics* .

Tae Yano, Dani Yogatama, and Noah A Smith. 2013. A penny for your tweets: Campaign contributions and capitol hill microblogs. In *Proc. of ICWSM*.

A Supplementary Material

In this section we provide additional information about our congressional tweets dataset, as well as the lists of keywords and phrases used to filter tweets by issue and the unigrams used to extract information used for the Unigram and MaxSim PSL predicates. It is important to note that during preprocessing capitalization, stop words, URLs, and punctuation have been removed from tweets in our dataset. Additional word lists along with our PSL scripts and dataset are available at: http://purduenlp.cs.purdue.edu/projects/twitterframing.

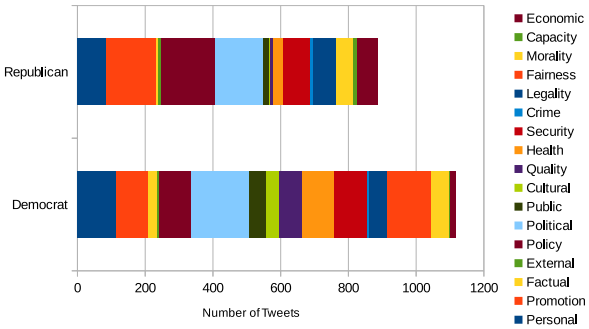


Figure 4: Coverage of Frames by Party.

Dataset Statistics: Figure 4 shows the coverage of the labeled frames by party. From this, general patterns can be observed. For example, Republicans use Frames 12 and 17 more frequently than Democrats, while Democrats tend to use Frames 4, 9, 10, and 11. Table 7 shows the count of each type of frame that appears in each issue in our labeled dataset.

Issue	Frames																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Abortion	4	7	23	55	40	0	2	32	10	0	4	46	20	0	1	13	8
ACA	65	9	6	28	24	0	3	128	21	3	18	116	174	2	21	100	15
Guns	2	2	37	16	30	21	93	8	36	14	49	166	65	0	5	55	147
Immigration	16	7	6	6	42	3	15	0	29	19	7	81	52	1	1	32	2
LGBTQ	0	0	9	99	23	2	2	3	10	17	7	39	14	1	2	11	48
Terrorism	6	4	46	3	11	10	115	1	6	13	14	69	68	35	6	99	57

Table 7: Count of Each Type of Frame Per Issue in Labeled Dataset.

Issue and Keywords or Phrases
Abortion: *abortion, pro-life, pro-choice, Planned Parenthood, StandWithPP, Hobby Lobby, birth control, women's choice, women's rights, women's health*
ACA: *patient protection, affordable care act, ACA, obamacare, health care, healthcare, Burwell, Medicare, Medicaid, repeal and replace*
Guns: *Charleston, gun, shooting, Emanuel, Second Amendment, Oregon, San Bernadino, gun violence, gun control, 2A, NRA, Orlando, Pulse*
Immigration: *immigration, immigrants, illegal immigrants, border, amnesty, wall, Dreamers, Dream Act*
LGBTQ: *equality, marriage, gay, transgender, marriage equality, same sex, gay marriage, religious freedom, RFRA, bathroom bill*
Terrorism: *terrorism, terrorists, terror network, ISIS, ISIL, Al Qaeda, Boko Haram, extremist*

Table 8: Keywords or Phrases Used to Filter Tweets for Issue.

Frame Number, Frame, and Adapted Unigrams
1. Economic: *premium(s), small, business(es), tax(es), economy, economic, cost(s), employment, market, spending, billion(s), million(s), company, companies, funding, regulation, benefit(s), health*
2. Capacity & Resources: *resource(s), housing, infrastructure, IRS, national, provide(s), providing, fund(s), funding, natural, enforcement*
3. Morality & Ethics: *moral, religion(s), religious, honor(able), responsible, responsibility, illegal, protect, god(s), sanctity, Islam, Muslim, Christian, radical, violence, victim(s), church*
4. Fairness & Equality: *fair(ness), equal(ity), inequality, law(s), right(s), race, gender, class, access, poor, civil, justice, social, women(s), LGBT, LGBTQ, discrimination, decision(s)*
5. Legality, Constitutionality, & Jurisdiction: *right(s), law(s), executive, ruling, constitution(al), amnesty, decision(s), reproductive, legal, legality, court, SCOTUS, immigration, amendment(s), judge, authority, precedent, legislation*
6. Crime & Punishment: *crime(s), criminal(s), gun(s), violate(s), enforce(s), enforced, enforcement, civil, tribunals, justice, victim(s), civilian(s), kill, murder, hate, genocide, consequences*
7. Security & Defense: *security, secure, defense, defend, threat(s), terror, terrorism, terrorist(s), gun(s), attack(s), wall, border, safe, safety, violent, violence, ISIS, ISIL, suspect(s), domestic, prevent, protect*
8. Health & Safety: *health(y), care, healthcare, obamacare, access, disease(s), mental, physical, affordable, coverage, quality, (un)insured, disaster, relief, unsafe, cancer, abortion*
9. Quality of Life: *quality, happy, social, community, life, benefit(s), adopt, fear, deportation, living, job(s), activities, family, families, health, support*
10. Cultural Identity: *identity, social, value(s), Reagan, Lincoln, conservative(s), liberal(s), nation, America, American(s), community, communities, country, dreamers, immigrants, refugees, history, historical*
11. Public Sentiment: *public, sentiment, opinion, poll(s), turning, survey, support, American(s), reform, action, want, need, vote*
12. Political Factors & Implications: *politic(s), political, stance, view, (bi)partisan, filibuster, lobby, Republican(s), Democrat(s), House, Senate, Congress, committee, party, POTUS, SCOTUS, administration, GOP*
13. Policy Description, Prescription, & Evaluation: *policy, fix(ing), work(s), working, propose(d), proposing, proposal, solution, solve, outcome(s), bill, law, amendment, plan, support, repeal, reform*
14. External Regulation and Reputation: *regulation, US, ISIS, ISIL, relations, international, national, trade, foreign, state, border, visa, ally, allies, united, refugees, leadership, issues, Iraq, Iran, Syria, Russia, Europe, Mexico, Canada*
15. Factual: *health, insurance, affordable, deadline, enroll, sign, signed, program, coverage*
16. (Self) Promotion: *statement, watch, discuss, hearing, today, tonight, live, read, floor, talk, tune, opinion, TV, oped*
17. Personal Sympathy & Support: *victims, thoughts, prayer(s), pray(ing), family, stand, support, tragedy, senseless, heartbroken, people, condolences, love, remember, forgive(ness), saddened*

Table 9: Frame and Corresponding Unigrams Used for Initial Supervision.

Word Lists: Table 8 lists the keywords or phrases used to filter the entire dataset to only tweets related to the six issues studied in this paper. Table 9 lists the unigrams that were designed based on the descriptions for Frames 1 through 14 provided in the Policy Frames Codebook (Boydstun et al., 2014). These unigrams provide the initial supervision for our models as described in Section 4.

A Nested Attention Neural Hybrid Model for Grammatical Error Correction

Jianshu Ji[†], **Qinlong Wang**[†], **Kristina Toutanova**[‡,*]
Yongen Gong[†], **Steven Truong**[†], **Jianfeng Gao**[§]
[†]Microsoft AI & Research [‡]Google Research [§]Microsoft Research, Redmond
[†§]{jianshuj,qinlwang,yongeg,stevetr,jfgao}@microsoft.com
[‡]kristout@google.com

Abstract

Grammatical error correction (GEC) systems strive to correct both global errors in word order and usage, and local errors in spelling and inflection. Further developing upon recent work on neural machine translation, we propose a new hybrid neural model with nested attention layers for GEC. Experiments show that the new model can effectively correct errors of both types by incorporating word and character-level information, and that the model significantly outperforms previous neural models for GEC as measured on the standard CoNLL-14 benchmark dataset. Further analysis also shows that the superiority of the proposed model can be largely attributed to the use of the nested attention mechanism, which has proven particularly effective in correcting local errors that involve small edits in orthography.

1 Introduction

One of the most successful approaches to grammatical error correction (GEC) is to cast the problem as (monolingual) machine translation (MT), where we translate from possibly ungrammatical English sentences to corrected ones (Brockett et al., 2006; Gao et al., 2010; Junczys-Dowmunt and Grundkiewicz, 2016). Such systems, which are based on phrase-based MT models that are typically trained on large sets of sentence-correction pairs, can correct global errors such as word order and usage and local errors in spelling and inflection. The approach has proven superior to systems based on local classifiers that can only fix focused errors in prepositions, determiners, or inflected forms (Rozovskaya and Roth, 2016).

Recently, neural machine translation (NMT) systems have achieved substantial improvements in translation quality over phrase-based MT systems (Sutskever et al., 2014; Bahdanau et al., 2015). Thus, there is growing interest in applying neural systems to GEC (Yuan and Briscoe, 2016; Xie et al., 2016). In this paper, we significantly extend previous work, and explore new neural models to meet the unique challenges of GEC.

The core component of most NMT systems is a sequence-to-sequence (S2S) model which encodes a sequence of source words into a vector and then generates a sequence of target words from the vector. Unlike the phrase-based MT models, the S2S model can capture long-distance, or even global, word dependencies, which are crucial to correcting global grammatical errors and helping users achieve native speaker fluency (Sakaguchi et al., 2016). Thus, the S2S model is expected to perform better on GEC than phrase-based models. However, as we will show in this paper, to achieve the best performance on GEC, we still need to extend the standard S2S model to address several task-specific challenges, which we will describe below.

First, a GEC model needs to deal with an extremely large vocabulary that consists of a large number of words and their (mis)spelling variations. Second, the GEC model needs to capture structure at different levels of granularity in order to correct errors of different types. For example, while correcting spelling and local grammar errors requires only word-level or sub-word level information, e.g., *violets* → *violates* (spelling) or *violate* → *violates* (verb form), correcting errors in word order or usage requires global semantic relationships among phrases and words.

Standard approaches in neural machine translation, also applied to grammatical error correction by Yuan and Briscoe (2016), address the large vocabulary problem by restricting the vocabulary to a limited number of high-frequency words and re-

*This work was conducted while the third author worked at Microsoft Research.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 753–762
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1070

sorting to standard word translation dictionaries to provide translations for the words that are out of the vocabulary (OOV). However, this approach often fails to take into account the OOVs in context for making correction decisions, and does not generalize well to correcting words that are unseen in the parallel training data. An alternative approach, proposed by Xie et al. (2016), applies a character-level sequence to sequence neural model. Although the model eliminates the OOV issue, it cannot effectively leverage word-level information for GEC, even if it is used together with a separate word-based language model.

Our solution to the challenges mentioned above is a novel, hybrid neural model with nested attention layers that infuse both word-level and character-level information. The architecture of the model is illustrated in Figure 1. The word-level information is used for correcting global grammar and fluency errors while the character-level information is used for correcting local errors in spelling or inflected forms. Contextual information is crucial for GEC. Using the proposed model, by combining embedding vectors and attention at both word and character levels, we model all contextual words, including OOVs, in a unified context vector representation. In particular, as we will discuss in Section 5, the character-level attention layer captures most useful information for correcting local errors that involve small edits in orthography.

Our model differs substantially from the word-level S2S model of Yuan and Briscoe (2016) and the character-level S2S model of Xie et al. (2016) in the way we infuse information at both the word level and the character level. We extend the word-character hybrid model of Luong and Manning (2016), which was originally developed for machine translation, by introducing a character attention layer. This allows the model to learn substitution patterns at both the character level and the word level in an end-to-end fashion, using sentence-correction pairs.

We validate the effectiveness of our model on the CoNLL-14 benchmark dataset (Ng et al., 2014). Results show that the proposed model outperforms all previous neural models for GEC, including the hybrid model of Luong and Manning (2016), which we apply to GEC for the first time. When integrated with a large word-based n-gram language model, our GEC system achieves an $F_{0.5}$ of 45.15 on CoNLL-14, substantially exceeding the previously reported top performance of 40.56 achieved by using a neural model and an external language model (Xie et al., 2016).

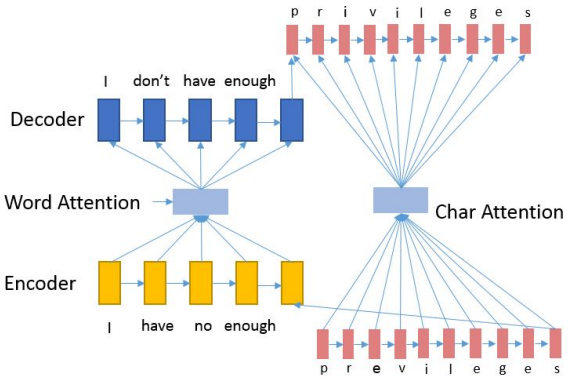


Figure 1: Architecture of Nested Attention Hybrid Model

2 Related Work

A variety of classifier-based and MT-based techniques have been applied to grammatical error correction. The CoNLL-14 shared task overview paper of Ng et al. (2014) provides a comparative evaluation of approaches. Two notable advances after the shared task have been in the areas of combining classifiers and phrase-based MT (Rozovskaya and Roth, 2016) and adapting phrase-based MT to the GEC task (Junczys-Dowmunt and Grundkiewicz, 2016). The latter work has reported the highest performance to date on the task of 49.5 in $F_{0.5}$ score on the CoNLL-14 test set. This method integrates discriminative training toward the task-specific evaluation function, a rich set of features, and multiple large language models. Neural approaches to the task are less explored. We believe that the advances from Junczys-Dowmunt and Grundkiewicz (2016) are complementary to the ones we propose for neural MT, and could be integrated with neural models to achieve even higher performance.

Two prior works explored sequence to sequence neural models for GEC (Xie et al., 2016; Yuan and Briscoe, 2016), while Chollampatt et al. (2016) integrated neural features in a phrase-based system for the task. Neural models were also applied to the related sub-task of grammatical error identification (Schmaltz et al., 2016). Yuan and Briscoe (2016) demonstrated the promise of neural MT for GEC but did not adapt the basic sequence-to-sequence with attention to its unique challenges, falling back to traditional word-alignment models to address vocabulary coverage with a post-processing heuristic. Xie et al. (2016) built a character-level sequence

to sequence model, which achieves open vocabulary and character-level modeling, but has difficulty with global word-level decisions.

The primary focus of our work is integration of character and word-level reasoning in neural models for GEC, to capture global fluency errors and local errors in spelling and closely related morphological variants, while obtaining open vocabulary coverage. This is achieved with the help of character and word-level encoders and decoders with two nested levels of attention. Our model is inspired by advances in sub-word level modeling in neural machine translation. We build mostly on the hybrid model of Luong and Manning (2016) to expand its capability to correct rare words by fine-grained character-level attention. We directly compare our model to the one of Luong and Manning (2016) on the grammar correction task. Alternative methods for MT include modeling of word pieces to achieve open vocabulary (Sennrich et al., 2016), and more recently, fully character-level modeling (Lee et al., 2017). None of these models integrate two nested levels of attention although an empirical evaluation of these approaches for GEC would also be interesting.

3 Nested Attention Hybrid Model

Our model is hybrid, and uses both word-level and character-level representations. It consists of a word-based sequence-to-sequence model as a backbone, and additional character-level encoder, decoder, and attention components, which focus on words that are outside the word-level model's vocabulary.

3.1 Word-based sequence-to-sequence model as backbone

The word-based backbone closely follows the basic neural sequence-to-sequence architecture with attention as proposed by Bahdanau et al. (2015) and applied to grammatical error correction by Yuan and Briscoe (2016). For completeness, we give a sketch here. It uses recurrent neural networks to encode the input sentence and to decode the output sentence.

Given a sequence of embedding vectors, corresponding to a sequence of input words $\mathbf{x}$:

$$\mathbf{x} = (x_1, \ldots, x_T), \tag{1}$$

the encoder creates a corresponding context-specific sequence of hidden state vectors $\mathbf{e}$:

$$\mathbf{e} = (h_1, \ldots, h_T)$$

The hidden state h_t at time t is computed as: $f_t = \text{GRU}_{\text{enc}_f}(f_{t-1}, x_t)$, $b_t = \text{GRU}_{\text{enc}_b}(b_{t+1}, x_t)$, $h_t = [f_t; b_t]$, where $\text{GRU}_{\text{enc}_f}$ and $\text{GRU}_{\text{enc}_b}$ stand for gated recurrent unit functions as described in Cho et al. (2014). We use the symbol GRU with different subscripts to represent GRU functions using different sets of parameters (for example, we used the enc_f and enc_b subscripts to denote the parameters of the forward and backward word-level encoder units.)

The decoder network is also an RNN using GRU units, and defines a sequence of hidden states $\bar{d}_1, \ldots, \bar{d}_S$ used to define the probability of an output sequence $y_1, \ldots, y_S$ as follows:

The context vector c_s at time step s is computed as follows:

$$c_s = \sum_{j=1}^{T} \alpha_{sj} h_j \tag{2}$$

where:

$$\alpha_{sk} = \frac{u_{sk}}{\sum_{j=1}^{T} u_{sj}} \tag{3}$$

$$u_{sk} = \phi_1(d_s)^T \phi_2(h_k) \tag{4}$$

Here ϕ_1 and ϕ_2 denote feedforward linear transformations followed by a $\tanh$ nonlinearity. The next hidden state $\bar{d}_s$ is then defined as:

$$d_s = \text{GRU}_{dec}(\bar{d_{s-1}}, y_{s-1}),$$

$$\bar{d}_s = \text{ReLU}(W[c_s; d_s])$$

where y_{s-1} is the embedding of the output token at time s-1. ReLU indicates rectified linear units (Hahnloser et al., 2000).

The probability of each target word y_s is computed as: $p(y_s|y_{<s}, \mathbf{x}) = \text{softmax}(g(\bar{d}_s))$, where g is a function that maps the decoder state into a vector of size the dimensionality of the target vocabulary.

The model is trained by minimizing the cross-entropy loss, which for a given $(\mathbf{x}, \mathbf{y})$ pair is:

$$Loss(\mathbf{x}, \mathbf{y}) = -\sum_{s=1}^{S} \log p(y_s|y_{<s}, \mathbf{x}) \tag{5}$$

For parallel training data $\mathbb{C}$, the loss is:

$$Loss = -\sum_{(\mathbf{x},\mathbf{y})\in\mathbb{C}} \sum_{s=1}^{S} \log p(y_s|y_{<s}, \mathbf{x})$$

3.2 Hybrid encoder and decoder with two nested levels of attention

The word-level backbone models a limited vocabulary of source and target words, and represents out-of-vocabulary tokens with special UNK symbols. In the standard word-level NMT approach, valuable information is lost for source OOV words and target OOV words are predicted using post-processing heuristics.

Hybrid encoder

Our hybrid architecture overcomes the loss of source information in the word-level backbone by building up compositional representations of the source OOV words using a character-level recurrent neural network with GRU units. These representations are used in place of the special source UNK embeddings in the backbone, and contribute to the contextual encoding of all source tokens.

For example, a three word input sentence where the last term is out-of-vocabulary will be represented as the following vector of embeddings in the word-level model: $\mathbf{x} = (x_1, x_2, x_3)$, where x_3 would be the embedding for the UNK symbol.

The hybrid encoder builds up a word embedding for the third word based on its character sequence: ${x^c}_1, \ldots, {x^c}_M$. The encoder computes a sequence of hidden states $\mathbf{e_c}$ for this character sequence, by a forward character-level GRU network:

$$\mathbf{e_c} = ({h^c}_1, \ldots, {h^c}_M), \tag{6}$$

The last state ${h^c}_M$ is used as an embedding of the unknown word. The sequence of embeddings for our example three-word sequence becomes: $\mathbf{x} = (x_1, x_2, {h^c}_M)$. We use the same dimensionality for word embedding vectors x_i and composed character sequence vectors ${h^c}_M$ to ensure the two ways to define embeddings are compatible. Our hybrid source encoder architecture is similar to the one proposed by Luong and Manning (2016).

Nested attention hybrid decoder

In traditional word-based sequence-to-sequence models special target UNK tokens are used to represent outputs that are outside the target vocabulary. A post-processing UNK-replacement method is then used (Cho et al., 2015; Yuan and Briscoe, 2016) to replace these special tokens with target words. The hybrid model of (Luong and Manning, 2016) uses a jointly trained character-level decoder to generate target words corresponding to UNK tokens, and outperforms the traditional approach in the machine translation task.

However, unlike machine translation, models for grammar correction conduct "translation" in the same language, and often need to apply a small number of local edits to the character sequence of a source word corresponding to the target UNK word. For example, rare but correct words such as entity names need to be copied as is, and local spelling errors or errors in inflection need to be corrected. The architecture of Luong and Manning (2016) does not have direct access to a source character sequence, but only uses a single fixed-dimensionality embedding of source unknown words aggregated with additional contextual information from the source.

To address the needs of the grammatical error correction task, we propose a novel hybrid decoder with two nested levels of attention: word level and character-level. The character-level attention serves to provide the decoder with direct access to the relevant source character sequence.

More specifically, the probability of each target word is defined as follows: For words in the target vocabulary, the probability is defined by the word-level backbone. For words outside the vocabulary, the probability of each token is the probability of UNK according to the backbone, multiplied by the probability of the word's character sequence.

The probability of the target character sequence corresponding to an UNK token at position s in the target is defined using a character-level decoder. As in Luong and Manning (2016), the "separate path" architecture is used to capture the relevant context and define the initial state for the character-level decoder:

$$\hat{d}_s = \text{ReLU}(\hat{W}[c_s; d_s])$$

where $\hat{W}$ are parameters different from W, and $\hat{d}_s$ is not used by the word-level model in predicting the subsequent tokens, but is only used to initialize the character-level decoder.

To be able to attend to the relevant source character sequence when generating the target character sequence, we use the concept of hard attention (Xu et al., 2015), but use an arg-max approximation for inference instead of sampling. A similar approach to represent discrete hidden structure in a variety of architectures is used in Kong et al. (2017).

The source index z_s corresponding to the target

position s is defined according to the word-level attention model:

$$z_s = \arg \max_{k \in 0...T-1} \alpha_{sk}$$

where α_{sk} are the intermediate outputs of the word-level attention model we described in Eq.(3).

The character-level decoder generates a character sequence $\mathbf{y^c} = ({y^c}_1, \ldots, {y^c}_N)$, conditioned on the initial vector $\hat{d}_s$ and the source index z_s. The characters are generated using a hidden state vector ${d^c}_n$ at each time step, via a softmax$(gc({d^c}_n))$, where gc maps the state to the target character vocabulary space.

If the source word x_{z_s} is in the source vocabulary, the model is analogous to the one of Luong and Manning (2016) and does not use character-level attention: the source context is available only in aggregated form to initialize the state of the decoder. The state ${d^c}_n$ for step n in the character-level decoder is defined as follows, where ${\text{GRU}^c}_{\text{dec}}$ are parameters for the gated recurrent cell of this decoder:

$${d^c}_n = \begin{cases} {\text{GRU}^c}_{\text{dec}}(\hat{d}_s, {y^c}_{n-1}) & n = 0 \\ {\text{GRU}^c}_{\text{dec}}({d^c}_{n-1}, {y^c}_{n-1}) & n > 0 \end{cases}$$

In contrast, if the corresponding token in the source x_{z_s} is also an out-of-vocabulary word, we define a second nested level of character attention and use it in the character-level decoder. The character-level attention focuses on individual characters from the source word x_{z_s}. If $\mathbf{e_c}$ are the source character hidden vectors computed as in Eq.(6), the recurrence equations for the character-level decoder with nested attention are:

$$\bar{{d^c}}_n = \text{ReLU}(W_c[{c^c}_n; {d^c}_n])$$

$${d^c}_n = \begin{cases} {\text{GRU}^c}_{\text{decNested}}(\hat{d}_s, {y^c}_{n-1}) & n = 0 \\ {\text{GRU}^c}_{\text{decNested}}(\bar{{d^c}}_{n-1}, {y^c}_{n-1}) & n > 0 \end{cases}$$

where ${c^c}_n$ is the context vector obtained using character-level attention on the sequence $\mathbf{e^c}$ and the last state of the character-level decoder ${d^c}_n$, computed following equations 2, 3 and 4, but using a different set of parameters.

These equations show that the character-level decoder with nested attention can use both the word-level state $\hat{d}_s$, and the character-level context ${c^c}_n$ and hidden state ${d^c}_n$ to perform global and local editing operations.

Since we introduced two architectures for the character-level decoder depending on whether the source word x_{z_s} is OOV, the combined loss function is defined as follows for end-to-end training:

$$Loss_{total} = Loss_w + \alpha Loss_{c1} + \beta Loss_{c2}$$

Here $Loss_w$ is the standard word-level loss in Eq.(5); character level losses $Loss_{c1}$ and $Loss_{c2}$ are losses for target OOV words corresponding to source known and unknown tokens, respectively. α and β are hyper-parameters to balance the loss terms.

As seen, our proposed nested attention hybrid model uses character-level attention only when both a predicted target word and its corresponding source input word are OOV. While the model can be naturally generalized to integrate character-level attention for known words, the original hybrid model proposed by Luong and Manning (2016) does not use any character-level information for known words. Thus for a controlled evaluation of the impact of the addition of character-level attention only, in this paper we limit character-level attention to OOV words, which already use characters as a basis for building their embedding vectors. A thorough investigation of the impact of character-level information in the encoder, attention, and decoder for known words as well is an interesting topic for future research.

Decoding for word-level and hybrid models

Beam-search is used to decode hypotheses according to the word-level backbone model. For the hybrid model architecture, word-level beam search is conducted first; for each target UNK token, character-level beam-search is used to generate a corresponding target word.

4 Experiments

4.1 Dataset and Evaluation

We use standard publicly available datasets for training and evaluation. One data source is the NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013), which is provided as a training set for the CoNLL-13 and CoNLL-14 shared tasks. From the original corpus of size about 60K parallel sentences, we randomly selected close to 5K sentence pairs for use as a validation set, and 45K parallel sentences for use in training. A second data source

	Training	Validation	Development	Test
#Sent pairs	2,608,679	4,771	1,381	1,312

Table 1: Overview of the datasets used.

Source	#Sent pairs
NUCLE	45,422
CLC	1,517,174
lang-8	1,046,083
Total	2,608,679

Table 2: Training data by source.

is the Cambridge Learner Corpus (CLC) (Nicholls, 2003), from which we extracted a substantially larger set of parallel sentences. Finally, we used additional training examples from the Lang-8 Corpus of Learner English v1.0 (Tajiri et al., 2012). As Lang-8 data is crowd-sourced, we used heuristics to filter out noisy examples: we removed sentences longer than 100 words and sentence pairs where the correction was substantially shorter than the input text. Table 2 shows the number of sentence pairs from each source used for training.

We evaluate the performance of the models on the standard sets from the CoNLL-14 shared task (Ng et al., 2014). We report final performance on the CoNLL-14 test set without alternatives, and analyze model performance on the CoNLL-13 development set (Dahlmeier et al., 2013). We use the development and validation sets for model selection. The sizes of all datasets in number of sentences are shown in Table 1. We report performance in $F_{0.5}$-measure, as calculated by the `m2scorer`—the official implementation of the scoring metric in the shared task. [1] Given system outputs and gold-standard edits, `m2scorer` computes the $F_{0.5}$ measure of a set of system edits against a set of gold-standard edits.

4.2 Baseline

We evaluate our model in comparison to the strong baseline of a word-based neural sequence-to-sequence model with attention, with post-processing for handling out-of-vocabulary words (Yuan and Briscoe, 2016); we refer to this model as word NMT+UNK replacement. Like Yuan and Briscoe (2016), we use a traditional word-alignment model (GIZA++) to derive a word-correction lexicon from the parallel training set. However, in decoding, we don't use GIZA++ to find the corresponding source word for each target OOV, but follow Cho et al. (2015), Section 3.3 to use the NMT system's attention weights instead. The target OOV is then replaced by the most likely correction of the source word from the word-correction lexicon, or by the source word itself if there are no available corrections.

[1] `http://www.comp.nus.edu.sg/~nlp/sw/m2scorer.tar.gz`

4.3 Training Details and Results

The embedding size for all word and character-level encoders and decoders is set to 1000, and the hidden unit size is also 1000. To reproduce the model of Yuan and Briscoe (2016), we selected the word vocabulary for the baseline by choosing the 30K most frequent words in the source and target respectively to form the source and target vocabularies. In preliminary experiments for the hybrid models, we found that selecting the same vocabulary of 30K words for the source and target based on combined frequency was better (.003 in $F_{0.5}$) and use that method for vocabulary selection instead. However, there was no gain observed by using such a vocabulary selection method in the baseline. Although the source and target vocabularies in the hybrid models are the same, like in the word-level model, the embedding parameters for source and target words are not shared.

The hyper-parameters for the losses in our models are selected based on the development set and set as follows: $\alpha = \beta = 0.5$. All models are trained with mini-batch size of 128 (batches are shuffled), initial learning rate of 0.0003 and a 0.95 decay ratio if the cost increases in two consecutive 100 iterations. The gradient is rescaled whenever its norm exceeds 10, and dropout is used with a probability of 0.15. Parameters are uniformly initialized in $[-\frac{\sqrt{(3)}}{\sqrt{1000}}, \frac{\sqrt{(3)}}{\sqrt{1000}}]$.

We perform inference on the validation set every 5000 iterations to log word-level cost and character-level costs; we save parameter values for the model every 10000 iterations as well as the end of each epoch. The stopping point for training is selected based on development set $F_{0.5}$ among the top 20 parameter sets with best validation set value of the loss function. Training of the nested attention hybrid model takes approximately five days on a Tesla k40m GPU. The basic hybrid model trains in around four days and the word-level backbone trains in approximately three days.

Table 3 shows the performance of the baseline and our nested attention hybrid model on the development and test sets. In addition to the word-level

Model	Performance	
	Dev	Test
Word NMT + UNK replacement	26.17	38.77
Hybrid model	28.49	40.44
Nested Attention Hybrid Model	**28.61**	**41.53**

Table 3: $F_{0.5}$ results on the CoNLL-13 and CoNLL-14 test sets of main model architectures.

baseline, we include the performance of a hybrid model with a single level of attention, which follows the work of Luong and Manning (2016) for machine translation, and is the first application of a hybrid word/character-level model to grammatical error correction. Based on hyper-parameter selection, the character-level component weight of the loss is $\alpha = 1$ for the basic hybrid model.

As shown in Table 3, our implementation of the word NMT+UNK replacement baseline approaches the performance of the one reported in Yuan and Briscoe (2016) (38.77 versus 39.9). We attribute the difference to differences in the training set and the word-alignment methods used. Our reimplementation serves to provide a controlled experimental evaluation of the impact of hybrid models and nested attention on the GEC task. As seen, our nested attention hybrid model substantially improves upon the baseline, achieving a gain of close to 3 points on the test set. The hybrid word/character model with a single level of attention brings a large improvement as well, showing the importance of character-level information for this task. We delve deeper into the impact of nested attention for the hybrid model in Section 5.

4.4 Integrating a Web-scale Language Model

The value of large language models for grammatical error correction is well known, and such models have been used in classifier and MT-based systems. To establish the potential of such models in word-based neural sequence-to-sequence systems, we integrate a web-scale count-based language model. In particular, we use the modified Kneser-Ney 5-gram language model trained from Common Crawl (Buck et al., 2014), made available for download by Junczys-Dowmunt and Grundkiewicz (2016).

Candidates generated by neural models are reranked using the following linear interpolation of log probabilities: $s_{y|x} = \log P_{NN}(y|x) + \lambda \log P_{LM}(y)$. Here λ is a hyper-parameter that balances the weights of the neural network model and the language model. We tuned λ separately

Model	Performance	
	Dev	Test
Character-based NMT + LM (Xie et al., 2016)		40.56
Word NMT + UNK replacement + LM	31.73	42.82
Hybrid model + LM	33.21	44.99
Nested Attention Hybrid Model + LM	**33.47**	**45.15**

Table 4: $F_{0.5}$ results on the CoNLL-13 and CoNLL-14 test sets of main model architectures, when combined with a large language model.

for each neural model variant, by exploring values in the range $[0.0, 2.0]$ with step size 0.1, and selecting according to development set $F_{0.5}$. The selected values of λ are: 1.6 for word NMT + UNK replacement and 1.0 for the nested attention model.

Table 4 shows the impact of the LM when combined with the neural models implemented in this work. The table also lists the results reported by Xie et al. (2016), for their character-level neural model combined with a large word-level language model. Our best results exceed the ones reported in the prior work by more than 4 points, although we should note that Xie et al. (2016) used a smaller parallel data set for training.

5 Analysis

We analyze the impact of sub-word level information and the two nested levels of attention in more detail by looking at the performance of the models on different segments of the data. In particular, we analyze the performance of the models on sentences containing OOV source words versus ones without OOV words, and corrections to orthographically similar versus dissimilar word forms.

5.1 Performance by Segment: OOV versus Non-OOV

We present a comparative performance analysis of models on the CoNLL-13 development set. First, we divide the set into two segments: OOV and NonOOV, based on whether there is at least one OOV word in the given source input. Table 5 shows that both hybrid architectures substantially outperform the word-level model in both segments of the data. The additional nested character-level attention of our hybrid model brings a sizable improvement over the basic hybrid model in the OOV segment and a small degradation in the non-OOV segment. We should note that in future work character-level attention can be added for non-OOV source words in the nested attention model, which could improve performance on this segment as well.

Model	NonOOV	OOV	Overall
Word NMT + UNK replacement	27.61	21.57	26.17
Hybrid model	**29.36**	25.92	28.49
Nested Attention Hybrid Model	29.00	**27.39**	**28.61**

Table 5: $F_{0.5}$ results on the CoNLL-13 set of main model architectures, on different segments of the set according to whether the input contains OOVs.

source	This greatly **violets** the rights of people .
gold	This greatly **violates** the rights of people .
word NMT + UNK replacement	This greatly **violets** the rights of people .
Nested Attention Hybrid Model	This greatly **violates** the rights of people .

Table 6: An example sentence from the OOV segment where the nested attention hybrid model improves performance.

Table 6 shows an example where the nested attention hybrid model successfully corrects a misspelling resulting in an OOV word on the source, whereas the baseline word-level system simply copies the source word without fixing the error (since this particular error is not observed in the parallel training set).

5.2 Impact of Nested Attention on Different Error Types

To analyze more precisely the impact of the additional character-level attention introduced by our design, we continue to investigate the OOV segment in more detail.

The concept of *edit*, which is also used by the official M2 score metric, is defined as a minimal pair of corresponding sub-strings in a source sentence and a correction. For example, in the sentence fragment pair: "Even though there is a risk of causing **harms** to someone, people still **are prefers** to keep their pets without a leash." → "Even though there is a risk of causing **harm** to someone, people still **prefer** to keep their pets without a leash.", the minimal edits are "harms → harm" and "are prefers → prefer". The $F_{0.5}$ score is computed using weighted precision and recall of the set of a system's edits against one or more sets of reference edits.

For our in-depth analysis, we classify edits in the OOV segment into two types: *small changes* and *large changes*, based on whether the source and target phrase of the edit are orthographically similar or not. More specifically, we say that the target and source phrases are orthographically similar, iff: the character edit distance is at most 2 and the source or target is at most 8 characters long, or $edit\_ratio < 0.25$, where $edit\_ratio = \frac{character\_edit\_distance}{\min(len(src),len(tar))+0.1}$, $len(*)$ denotes number of characters in $*$, and src and tgt denote the pairs in the edit. There are 307 gold edits in the "small changes" portion of the CoNLL-13 OOV segment, and 481 gold edits in the "large changes" portion.

Model	Performance		
	P	R	$F_{0.5}$
Small Changes Portion			
Hybrid model	43.86	16.29	32.77
Nested Attention Hybrid Model	48.25	17.92	36.04
Large Changes Portion			
Hybrid model	32.52	8.32	20.56
Nested Attention Hybrid Model	33.05	8.11	20.46

Table 7: Precision, Recall and $F_{0.5}$ results on CoNLL-13,on the "small changes" and "large changes" portions of the OOV segment.

Our hypothesis is that the additional character-level attention layer is particularly useful to model edits among orthographically similar words. Table 7 contrasts the impact of character-level attention on the two portions of the data. We can see that the gains in the "small changes" portion are indeed quite large, indicating that the fine-grained character-level attention empowers the model to more accurately correct confusions among phrases with high character-level similarity. The impact in the "large changes" portion is slightly positive in precision and slightly negative in recall. Thus most of the benefit of the additional character-level attention stems from improvements in the "small changes" portion.

Table 8 shows an example input which illustrates the precision gain of the nested attention hybrid model. The input sentence has a source OOV word which is correct. The hybrid model introduces an error in this word, because it uses only a single source context vector, aggregating the character-level embedding of the source OOV word together with other source words. The additional character-level attention layer in the nested hybrid model enables the correct copying of this long source OOV word, without employing the heuristic mechanism of the word-level NMT system.

source	Population ageing : A more and more **attention-getting** topic
gold	Population ageing : A more and more **attention-getting** topic
Word NMT + UNK replacement	Population ageing : A more and more **attention-getting** topic
Hybrid Model	Population ageing : A more and more **attention-teghting** topic
Nested Attention Hybrid Model	Population ageing : A more and more **attention-getting** topic

Table 8: An example where the nested attention hybrid model outperforms the non-nested model.

6 Conclusions

We have introduced a novel hybrid neural model with two nested levels of attention: word-level and character-level. The model addresses the unique challenges of the grammatical error correction task and achieves the best reported results on the CoNLL-14 benchmark among fully neural systems. Our nested attention hybrid model deeply combines the strengths of word and character level information in all components of an end-to-end neural model: the encoder, the attention layers, and the decoder. This enables it to correct both global word-level and local character-level errors in a unified way. The new architecture contributes substantial improvement in correction of confusions among rare or orthographically similar words compared to word-level sequence-to-sequence and non-nested hybrid models.

Acknowledgements

We would like to thank the ACL reviewers for their insightful suggestions, Victoria Zayats for her help with reproducing the baseline word-level NMT system and Yu Shi, Daxin Jiang and Michael Zeng for the helpful discussions.

References

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

Chris Brockett, William B Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. pages 249–256.

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. 2014. N-gram counts and language models from the Common Crawl. In *LREC*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sébastien Jean Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL*.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural network translation models for grammatical error correction. In *Proceedings of IJCAI*.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. pages 22–31.

Jianfeng Gao, Xiaolong(Shiao-Long) Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *The 23rd International Conference on Computational Linguistics*.

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature* 405(6789):947–951.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *EMNLP*.

Lingpeng Kong, Chris Alberti, Daniel Andor, Ivan Bogatyy, and David Weiss. 2017. Dragnn: A transition-based framework for dynamically connected neural networks.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *TACL* 5.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *CoNLL Shared Task*. pages 1–14.

Diane Nicholls. 2003. The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*. volume 16, pages 572–581.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 2205–2215.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics* 4:169–182.

Allen Schmaltz, Yoon Kim, Alexander M Rush, and Stuart M Shieber. 2016. Sentence-level grammatical error identification as sequence-to-sequence correction. *arXiv preprint arXiv:1604.04677* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. pages 198–202.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. 2016. Neural language correction with character-based attention. *CoRR* abs/1603.09727. http://arxiv.org/abs/1603.09727.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. volume 14, pages 77–81.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

TextFlow: A Text Similarity Measure based on Continuous Sequences

Yassine Mrabet mrabety@mail.nih.gov
Halil Kilicoglu kilicogluh@mail.nih.gov
Dina Demner-Fushman ddemner@mail.nih.gov
Lister Hill National Center for Biomedical Communications
U.S. National Library of Medicine
8600 Rockville Pike, 20894, Bethesda, MD, USA

Abstract

Text similarity measures are used in multiple tasks such as plagiarism detection, information ranking and recognition of paraphrases and textual entailment. While recent advances in deep learning highlighted further the relevance of sequential models in natural language generation, existing similarity measures do not fully exploit the sequential nature of language. Examples of such similarity measures include n-grams and skip-grams overlap which rely on distinct slices of the input texts. In this paper we present a novel text similarity measure inspired from a common representation in DNA sequence alignment algorithms. The new measure, called *TextFlow*, represents input text pairs as continuous curves and uses both the actual position of the words and sequence matching to compute the similarity value. Our experiments on eight different datasets show very encouraging results in paraphrase detection, textual entailment recognition and ranking relevance.

1 Background

The number of pages required to print the content of the World Wide Web was estimated to 305 billion in a 2015 article[1]. While a big part of this content consists of visual information such as pictures and videos, texts also continue growing at a very high pace. A recent study shows that the average webpage weights 1,200 KB with plain text accounting for up to 16% of that size[2].

While efficient distribution of textual data and computations are the key to deal with the increasing scale of textual search, similarity measures still play an important role in refining search results to more specific needs such as the recognition of paraphrases and textual entailment, plagiarism detection and fine-grained ranking of information. These tasks are also often performed on dedicated document collections for domain-specific applications where text similarity measures can be directly applied.

Finding relevant approaches to compute text similarity motivated a lot of research in the last decades (Sahami and Heilman, 2006; Hatzivassiloglou et al., 1999), and more recently with deep learning methods (Socher et al., 2011; Yih et al., 2011; Severyn and Moschitti, 2015). However, most of the recent advances focused on designing high performance classification methods, trained and tested for specific tasks and did not offer a standalone similarity measure that could be applied (i) regardless of the application domain and (ii) without requiring training corpora.

For instance, Yih and Meek (2007) presented an approach to improve text similarity measures for web search queries with a length ranging from one word to short sequences of words. The proposed method is tailored to this specific task, and the training models are not expected to perform well on different kinds of data such as sentences, questions or paragraphs. In a more general study, Achananuparp et al. (2008) compared several text similarity measures for paraphrase recognition, textual entailment, and the TREC 9 question variants task. In their experiments the best performance was obtained with a linear combination of semantic and lexical similarities, including a word order similarity proposed in (Li et al., 2006). This word order similarity is computed by constructing first two vectors representing the common words between two given sentences and using their respective positions in the sentences as term

[1] http://goo.gl/p9lt7V
[2] http://goo.gl/c41wpa

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 763–772
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1071

weights. The similarity value is then obtained by subtracting the two vectors and taking the absolute value. While such representation takes into account the actual positions of the words, it does not allow detecting sub-sequence matches and takes into account missing words only by omission.

More generally, existing standalone (or traditional) text similarity measures rely on the intersections between token sets and/or text sizes and frequency, including measures such as the Cosine similarity, Euclidean distance, Levenshtein (Sankoff and Kruskal, 1983), Jaccard (Jain and Dubes, 1988) and Jaro (Jaro, 1989). The sequential nature of natural language is taken into account mostly through word n-grams and skip-grams which capture distinct slices of the analysed texts but do not preserve the order in which they appear.

In this paper, we use intuitions from a common representation in DNA sequence alignment to design a new standalone similarity measure called *TextFlow* (*XF*). The proposed measure uses at the same time the full sequence of input texts in a natural sub-sequence matching approach together with individual token matches and mismatches. Our contributions can be detailed further as follows:

- A novel standalone similarity measure which:
  - exploits the full sequence of words in the compared texts.
  - is asymmetric in a way that allows it to provide the best performance on different tasks (e.g., paraphrase detection, textual entailment and ranking).
  - when required, it can be trained with a small set of parameters controlling the impact of sub-sequence matching, position gaps and unmatched words.
  - provides consistent high performance across tasks and datasets compared to traditional similarity measures.
- A neural network architecture to train *TextFlow* parameters for specific tasks.
- An empirical study on both performance consistency and standard evaluation measures, performed with eight datasets from three different tasks.

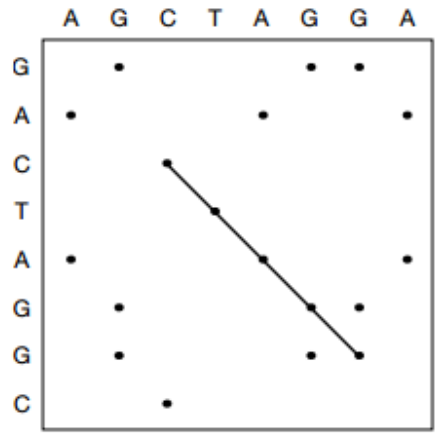


Figure 1: Dot matrix example for 2 DNA sequences (Mount, 2004)

- A new evaluation measure, called *CORE*, used to better show the consistency of a system at high performance using both its rank average and rank variance when compared to competing systems over a set of datasets.

2 The *TextFlow* Similarity

XF is inspired from a dot matrix representation commonly used in pairwise DNA sequence alignment (cf. figure 1). We use a similar dot matrix representation for text pairs and draw a curve oscillating around the diagonal (cf. figure 2). The area under the curve is considered to be the distance between the two text pairs which is then normalized with the matrix surface. For practical computation, we transform this first intuitive representation using the delta of positions as in figure 3. In this setting, the Y axis is the delta of positions of a word occurring in the two texts being compared. If the word does not occur in the target text, the delta is considered to be a maximum reference value (l in figure 2).

The semantics are: the bigger the area under curve is, the lower the similarity between the compared texts. XF values are real numbers in the [0,1] interval, with 1 indicating a perfect match, and 0 indicating that the compared texts do not have any common tokens. With this representation, we are able to take into account all matched words and sub-sequences at the same time. The exact value for the XF similarity between two texts $X = \{x_1, x_2, .., x_n\}$ and $Y = \{y_1, y_2, .., y_m\}$ is therefore computed as:

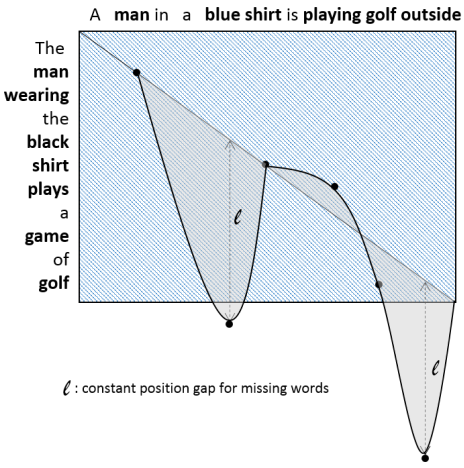


Figure 2: Illustration of *TextFlow* Intuition

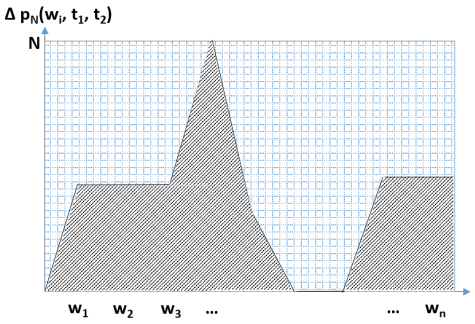


Figure 3: Practical *TextFlow* Computation

$$XF(X,Y) = 1 - \frac{1}{nm}\sum_{i=2}^{n}\frac{1}{S_i}T_{i,i-1}(X,Y) - \frac{1}{nm}\sum_{i=2}^{n}\frac{1}{S_i}R_{i,i-1}(X,Y) \quad (1)$$

With $T_{i,i-1}(X,Y)$ corresponding to the triangular area in the $[i-1,i]$ step (cf. figure 3) and $R_{i,i-1}(X,Y)$ corresponding to the rectangular component. They are expressed as:

$$T_{i,i-1}(X,Y) = \frac{|\Delta P(x_i,X,Y) - \Delta P(x_{i-1},X,Y)|}{2} \quad (2)$$

and:

$$R_{i,i-1}(X,Y) = Min(\Delta P(x_i,X,Y), \Delta P(x_{i-1},X,Y)) \quad (3)$$

With:

- $\Delta P(x_i,X,Y)$ the minimum difference between x_i positions in X and Y. x_i position in X is multiplied by the factor $\frac{|Y|}{|X|}$ for normalization. If $x_i \notin X \cap Y$, $\Delta P(x_i,X,Y)$ is set to the same reference value equal to m, (i.e., the cost of a missing word is set by default to the length of the target text), and:
- S_i is the length of the longest matching sequence between X and Y including the word x_i, if $x_i \in X \cap Y$, or 1 otherwise.

XF computation is performed in $O(nm)$ in the worst case where we have to check all tokens in the target text Y for all tokens in the input text X. XF is an asymmetric similarity measure. Its asymmetric aspect has interesting semantic applications as we show in the example below (cf. figure 2). The minimum value of XF provided the best differentiation between positive and negative text pairs when looking for semantic equivalence (i.e., paraphrases), the maximum value was among the top three for the textual entailment example. We conduct this comparison at a larger scale in the evaluation section.

We add 3 parameters to *XF* in order to represent the importance that should be given to position deltas (Position factor α), missing words (sensitivity factor β), and sub-sequence matching (sequence factor γ), such that:

$$XF_{\alpha,\beta,\gamma}(X,Y) = 1 - \frac{1}{\beta nm}\sum_{i=2}^{n}\frac{\alpha}{S_i^{\gamma}}T_{i,i-1}^{\beta}(X,Y) - \frac{1}{\beta nm}\sum_{i=2}^{n}\frac{\alpha}{S_i^{\gamma}}R_{i,i-1}^{\beta}(X,Y) \quad (4)$$

With:

$$T_{i,i-1}^{\beta}(X,Y) = \frac{|\Delta_\beta P(x_i,X,Y) - \Delta_\beta P(x_{i-1},X,Y)|}{2} \quad (5)$$

$$R_{i,i-1}^{\beta}(X,Y) = Min(\Delta_\beta P(x_i,X,Y), \Delta_\beta P(x_{i-1},X,Y)) \quad (6)$$

and:

- $\Delta_\beta P(x_i,X,Y) = \beta m$, if $x_i \notin X \cap Y$
- $\alpha < \beta$: forces missing words to always cost more than matched words.
- $S_i^{\gamma} = \begin{cases} 1 if S_i = 1 or x_i \notin X \cap Y \\ \gamma\ S_i for S_i > 1 \end{cases}$

The γ factor increases or decreases the impact of sub-sequence matching, α applies to individual token matches whether inside or outside a sequence, and β increases or decreases the impact of

Positive Entailment	E_1	Under a blue sky with white clouds, a child reaches up to touch the propeller of a plane standing parked on a field of grass.
	E_2	A child is reaching to touch the propeller of a plane.
Negative Entailment	E_3	Two men on bicycles competing in a race.
	E_4	Men are riding bicycles on the street.
Positive Paraphrase	P_1	The most serious breach of royal security in recent years occurred in 1982 when 30-year-old Michael Fagan broke into the queen's bedroom at Buckingham Palace.
	P_2	It was the most serious breach of royal security since 1982 when an intruder, Michael Fagan, found his way into the Queen's bedroom at Buckingham Palace.
Negative Paraphrase	P_3	"Americans don't cut and run, we have to see this misadventure through," she said.
	P_4	She also pledged to bring peace to Iraq: "Americans don't cut and run, we have to see this misadventure through."

Task	Entailment Recognition			Paraphrase Detection		
Sentence Pair	(E_1, E_2)	(E_3, E_4)	(E_1, E_2) - (E_3, E_4)	(P_1, P_2)	(P_3, P_4)	(P_1, P_2) - (P_3, P_4)
Example class (Pos/Neg)	(Pos)	(Neg)	(Gap)	(Pos)	(Neg)	(Gap)
Jaro-Winkler	0.629	0.712*	-0.083**	**0.884**	0.738	0.146
Levenshtein	0.351	0.259	0.092	0.708	0.577	0.130
Jaccard	0.250*	**0.143**	0.107	0.571*	0.583	-0.012
Cosine	0.462	0.250	0.212	0.730	0.746**	-0.016
Word Overlap	**0.800**	0.250	**0.550**	0.800	0.875*	-0.075
MIN(XF (x,y), XF(y,x))	0.260**	0.563**	-0.303*	0.693**	**0.497**	**0.196**
MAX(XF(x,y), XF(y,x))	0.707	0.563**	0.144	0.832	0.739	0.093

Figure 4: Example sentences and similarity values. The best value per column is highlighted. The second best is underlined. Worst and second worst values are followed by one and two stars. Entailment examples are taken from SNLI (Bowman et al., 2015). Paraphrase examples are taken from MSRP [4].

missing tokens as well as the normalization quantity βnm in equation 4 to keep the similarity values in the [0,1] range.

2.1 Parameter Training

By default *XF* has canonical parameters set to 1. However, when needed, α, β, and γ can be learned on training data for a specific task. We designed a neural network to perform this task, with a hidden layer dedicated to compute the exact XF value. To do so we compute, for each input text pair, the coefficients vector that would lead exactly to the XF value when multiplied by the vector $< \frac{\alpha}{\beta}, \frac{\alpha}{\beta\gamma}, 1 >$. Figure 5) presents the training neural network considering several types of sequences (or translations) of the input text pairs (e.g., lemmas, words, synsets).

We use identity as activation function in the dedicated XF layer in order to have a correct comparison with the other similarity measures, including canonical XF where the similarity value is provided in the input layer (cf. figure 6).

3 Evaluation

Datasets. This evaluation was performed on 8 datasets from 3 different classification tasks: Textual Entailment Recognition, Paraphrase Detection, and ranking relevance. The datasets are as follows:

- **RTE 1, 2, and 3**: the first three datasets from the Recognizing Textual Entailment (RTE) challenge (Dagan et al., 2006). Each dataset consists of sentence pairs which are annotated with 2 labels: entailment, and non-entailment. They contain respectively (200, 800), (800, 800), and (800, 800) (train, test) pairs.

- **Guardian**: an RTE dataset collected from 78,696 Guardian articles[5] published from January 2004 onwards and consisting of 32K pairs which we split randomly in 90%/10% training/test sets. Positive examples were collected from the titles and first sentences. Negative examples were collected from the same source by selecting consecutive sentences and random sentences.

- **SNLI**: a recent RTE dataset consisting of 560K training sentence pairs annotated with

[5] https://github.com/daoudclarke/rte-experiment

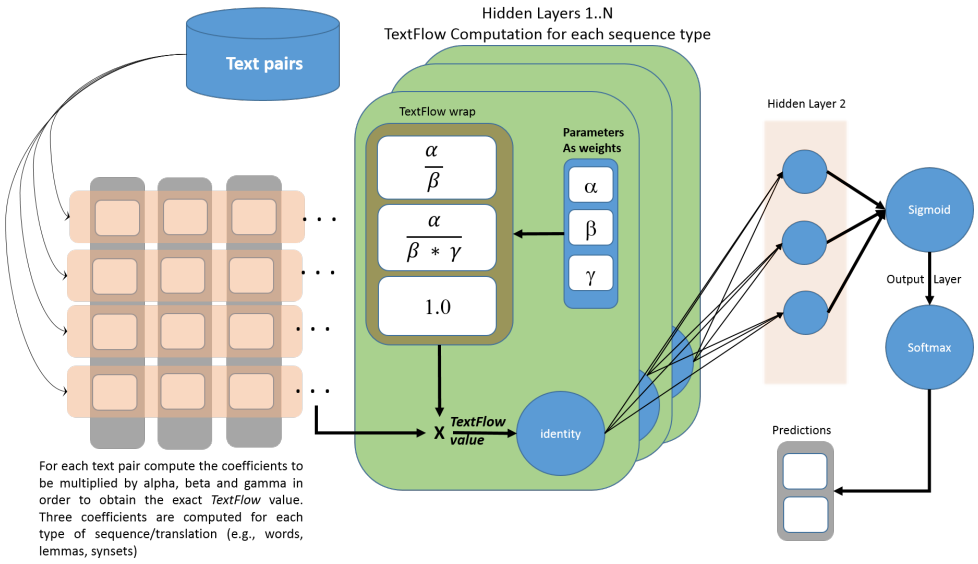


Figure 5: NN architecture **A1** for XF Parameter Training

3 labels: entailment, neutral and contradiction (Bowman et al., 2015). We discarded the contradiction pairs as they do not necessarily represent dissimilar sentences and are therefore a random noise w.r.t. our similarity measure evaluation.

- **MSRP**: the Microsoft Research Paraphrase corpus, consisting of 5,800 sentence pairs annotated with a binary label indicating whether the two sentences are paraphrases or not.

- **Semeval-16-3B**: a dataset of question-question similarity collected from StackOverflow (Nakov et al., 2016). The dataset contains 3,169 training pairs and 700 test pairs. Three labels are considered: "Perfect Match", "Relevant" or "Irrelevant". We combined the first two into the same positive category for our evaluation.

- **Semeval-14-1**: a corpus of Sentences Involving Compositional Knowledge (Marelli et al., 2014) consisting of 10,000 English sentence pairs annotated with both similarity scores and relevance labels.

Features. After a preprocessing step where we removed stopwords, we computed the similarity values using 7 different types of sequences constructed, respectively, with the following value from each token:

- Word (plain text value)
- Lemma
- Part-Of-Speech (POS) tag
- WordNet Synset[6] OR Lemma
- WordNet Synset OR Lemma for Nouns
- WordNet Synset OR Lemma for Verbs
- WordNet Synset OR Lemma for Nouns and Verbs.

In the last 4 types of sequences the lemma is used when there is no corresponding WordNet synset. In a first experiment we compare different aggregation functions on top of XF (minimum, maximum and average) in table 1. We used the LibLinear[7] SVM classifier for this task.

In the second part of the evaluation, we use neural networks to compare the efficiency of XF_c, XF_t and other similarity measures with in the same setting. We use the neural net described in figure 5 for the trained version XF_t and the equivalent architecture presented in figure 6 for all other similarity measures. For canonical XF_c we use by default the best aggregation for the task as observed in table 3.

[6] https://wordnet.princeton.edu/
[7] https://www.csie.ntu.edu.tw/~cjlin/liblinear/

Task	Entailment Recognition					Paraphrase Detection	Ranking Relevance	
Datasets	RTE 1	RTE 2	RTE 3	Guardian	SNLI	MSRP	Semeval16-t3B	Semeval12-t17
XF MIN	**55.3**	53.8	60.0	77.3	58.0	**72.1**	77.4	77.8
XF AVG	51.4 1	57.2	62.5	84.9	62.0	72.0	**77.6**	**79.5**
XF MAX	53.9	**61.3**	**64.7**	**86.7**	**64.3**	71.4	76.7	77.7

Table 1: Accuracy evaluation with different aggregations of XF using an SVM classifier.

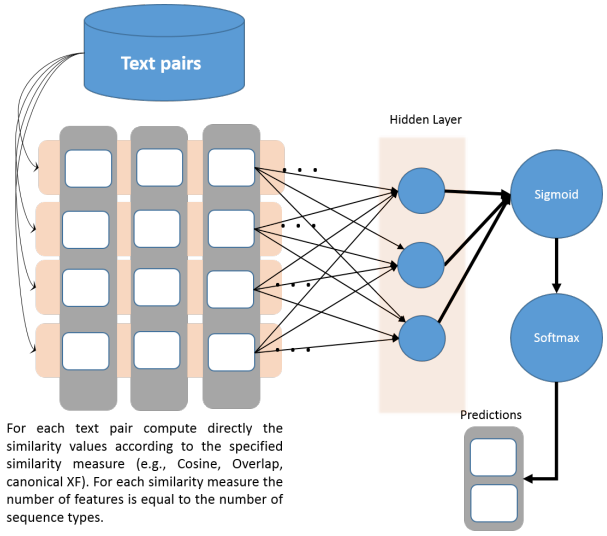


Figure 6: NN Architecture **A2** for the equivalent evaluation of other similarity measures.

Similarity Measures. We considered nine traditional similarity measures included in the Simmetrics distribution[8]: Cosine, Euclidean distance, Word Overlap, Dice coefficient (Dice, 1945), Jaccard(Jain and Dubes, 1988), Damerau, Jaro-Winkler (JW) (Porter et al., 1997), Levenshtein (LEV) (Sankoff and Kruskal, 1983), and Longest Common Subsequence (LCS) (Friedman and Sideli, 1992).

Implementation. XF was implemented in Java as an extension of the Simmetrics package, made available at this address[9]. The neural networks were implemented in Python with TensorFlow[10]. We also share the training sets used for both parameter training and evaluation. The evaluation was performed on a 4-core laptop with 32GB of RAM. The initial parameters for XF_t were chosen with a random function.

Evaluation Measures. We use the standard accuracy values and F1, precision and recall for the positive class (i.e., entailment, paraphrase, and ranking relevance). We also study the relative rank in performance of each similarity measure across all datasets using the average rank, the rank variance and a new evaluation measure called Consistent peRformancE (CORE), computed as follows for a system m, a set of datasets D, a set of systems S, and an evaluation measure $E \in \{F1, Precision, Recall, Accuracy\}$:

$$\underset{D,S,E}{CORE}(m) = \frac{\underset{p \in S}{MIN}\left(\underset{d \in D}{AVG}(R_S(E_d(p)) + V_{d \in D}(R_S(E_d(p)))\right)}{\underset{d \in D}{AVG}\left(R_S(E_d(m))\right) + V_{d \in D}\left(R_S(E_d(m))\right)} \quad (7)$$

With $R_S(E_d(m))$ the rank of m according to the evaluation measure E on dataset d w.r.t. competing systems S. $V_{d \in D}(R_S(E_d(m)))$ is the rank variance of m over datasets. The results in tables 2, 3, and 4 demonstrate the intuition. Basically, $CORE$ tells us how consistent a system/method is in having high performance, relatively to the set of competing systems S. The maximum value of $CORE$ is 1 for the best performing system according to its rank. It also allows quantifying how consistently a system achieves high performance for the remaining systems.

TextFlow outperformed the results obtained with a combination of word order similarity and semantic similarities tested in (Achananuparp et al., 2008), with gaps of +1.0 in F1 and +6.1 accuracy on MSRP and +4.2 F1 and +2.7% accuracy on RTE 3.

4 Discussion

4.1 Canonical *Text Flow*

TF_c had the best average and micro-average accuracy on the 8 classification datasets, with a gap of +0.4 to +6.3 when compared to the state-of-the-art measures. It also reached the best precision average with a gap of +1.8 to +6.3. On the F1 score level XF_c achieved the second best performance with a gap of -1.7, mainly caused by its underperformance in recall, where it had the third best performance with a gap of -6.3 (cf. table 3). On a rank level, XF_c had the best consistent rank for

[8] https://github.com/Simmetrics/simmetrics

[9] https://github.com/ymrabet/TextFlow

[10] https://www.tensorflow.org/

	Cosine	Euc	Overlap	Dice	Jaccard	Damerau	JW	LEV	LCS	XF_C	XF_T
RTE 1	.561	.564	.550	.504	.511	.557	.532	.561	.568	.550	**.575**
RTE 2	.575	.555	.598	.566	.572	.548	.541	.551	.548	.597	**.612**
RTE 3	**.652**	.562	.636	.637	.630	.567	.538	.567	.562	.627	.647
Guardian	.748	.750	.820	.778	.780	.847	.726	.847	.848	.867	**.876**
SNLI	.621	.599	**.665**	.612	.608	.631	.556	.630	.619	.641	.656
MSRP	.719	.689	.720	.729	.731	.687	.699	.685	.717	.724	**.732**
Semeval-16-3B	.756	.734	.769	.781	.780	.759	.751	.759	.737	.777	**.782**
Semeval-14-1	.790	.756	.779	.783	.786	.749	.719	.749	.757	.783	**.798**
AVG	.678	.651	.692	.674	.675	.668	.633	.669	.670	.696	**.710**
Micro Avg	.699	.675	.725	.700	.700	.701	.646	.701	.701	.726	**.739**
RANK Avg	5.1	8.2	4.5	5.6	5.5	6.9	10.1	6.7	6.7	4.1	**1.2**
RANK Var.	9.0	5.9	4.3	10.0	8.6	5.3	1.6	6.2	8.2	2.7	**0.2**
CORE	0.104	0.103	0.167	0.094	0.104	0.121	0.125	0.113	0.098	0.215	**1.000**

Table 2: Accuracy values using. The best result is highlighted, the second best is underlined.

	Cosine	Euc	Overlap	Dice	Jaccard	Damerau	JW	LEV	LCS	XF_C	XF_T
RTE 1	.612	.564	**.636**	.512	.523	.578	.513	.583	.494	.565	.599
RTE 2	.579	.590	**.662**	.565	.558	.549	.516	.551	.555	.616	.646
RTE 3	**.705**	.598	.682	.695	.682	.608	.556	.607	.603	.665	.690
Guardian	.742	.749	.816	.774	.776	.849	.713	.849	.850	.862	**.873**
SNLI	.582	.576	**.641**	.562	.564	.627	.479	.627	.611	.594	.585
MSRP	.808	.797	.812	**.814**	.813	.784	.802	.783	.804	.804	.810
Semeval-16-3B	.632	.462	.625	.648	.644	.544	.545	.547	.508	.633	**.662**
Semeval-14-1	.764	.707	.748	.753	.746	.706	.680	.706	.714	.744	.673
AVG	.678	.630	**.702**	.665	.663	.655	.600	.656	.642	.685	.692
Micro Avg	.684	.656	**.716**	.679	.677	.691	.608	.692	.688	.702	.687
RANK Avg	4.5	8.12	3.12	5.12	5.5	6.89	9.88	6.62	7.12	4.62	**3.88**
RANK Var.	9.7	4.7	4.4	14.7	6.6	8.7	1.8	9.1	8.1	**2.3**	11.0
CORE	0.485	0.538	0.915	0.348	0.571	0.443	0.588	0.438	0.452	**1.000**	0.464

Table 3: F1 scores. The best result is highlighted, the second best is underlined.

	Cosine	Euc	Overlap	Dice	Jaccard	Damerau	JW	LEV	LCS	XF_C	XF_T
RTE 1	.548	.564	.534	.503	.510	.552	.535	.555	**.596**	.546	.566
RTE 2	.574	.547	.571	.567	.578	.547	.546	.551	.546	.588	**.594**
RTE 3	.624	.565	.618	.611	.610	.568	.547	.568	.564	.616	**.627**
Guardian	.759	.753	.836	.789	.789	.839	.749	.840	.839	.891	**.894**
SNLI	.644	.608	.690	.642	.632	.631	.577	.630	.621	.679	**.735**
MSRP	.740	.705	.732	.749	.755	.723	.713	.722	.743	.760	**.765**
Semeval-16-3B	.634	**.708**	.678	.698	.698	.732	.698	.729	.674	.700	.686
Semeval-14-1	.745	.738	.738	.743	**.769**	.716	.672	.716	.727	.762	.740
AVG	.659	.649	.675	.663	.668	.664	.630	.664	.664	.693	**.701**
Micro Avg	.693	.674	.721	.699	.704	.694	.645	.693	.693	.737	**.752**
RANK Avg.	5.6	7.5	5.9	5.9	5.1	6.1	9.6	6.1	7.1	3.2	**2.5**
RANK Var.	9.4	10.0	6.4	5.3	7.8	7.0	4.6	7.6	11.6	**3.1**	6.9
CORE	0.420	0.361	0.515	0.567	0.488	0.482	0.446	0.462	0.338	**1.000**	0.676

Table 4: Precision values. The best result is highlighted, the second best is underlined.

	Cosine	Euc	Overlap	Dice	Jaccard	Damerau	JW	LEV	LCS	XF_C	XF_T
RTE 1	.693	.564	**.786**	.521	.536	.607	.493	.614	.421	.585	.635
RTE 2	.585	.640	**.787**	.562	.540	.550	.490	.550	.565	.647	.707
RTE 3	**.810**	.634	.761	.805	.773	.654	.566	.651	.649	.724	.768
Guardian	.726	.744	.797	.758	.764	.858	.681	.858	**.862**	.835	.853
SNLI	.531	.548	.600	.499	.510	.624	.409	**.625**	.601	.527	.486
MSRP	.890	**.916**	.912	.890	.881	.857	.915	.856	.876	.854	.860
Semeval-16	.631	.343	.579	.605	.597	.433	.446	.438	.408	.579	**.639**
SICK	**.784**	.678	.759	.763	.724	.696	.688	.695	.701	.727	.616
AVG	**.706**	.633	.748	.675	.665	.660	.586	.661	.635	.685	.696
Micro Avg	.683	649	**.720**	.668	.659	.695	.587	.695	.688	.677	.645
RANK Avg.	3.9	7.1	**3.5**	5.5	6.4	6.1	9.0	6.1	6.6	5.9	5.4
RANK Var.	9.6	12.4	**3.4**	9.4	5.4	8.1	10.0	11.0	11.7	5.8	14.3
CORE	0.516	0.355	**1.000**	0.464	0.588	0.486	0.365	0.405	0.378	0.591	0.353

Table 5: Recall values. The best result is highlighted, the second best is underlined.

accuracy, F1 and precision, and the second best for recall.

4.2 Trained Text Flow

When compared to state-of-the-art measures and to canonical XF, the trained version, XF_t, obtained the best accuracy with a gap ranging from +1.4 to +7.8. XF_t also obtained the second best F1 average with a -1.0 gap, but with clear inconsistencies according to the dataset. XF_t obtained the best precision with a gap ranging from +0.8 to +7.1. XF_t did not perform well on recall with 64.5% micro-average compared to WordOverlap with 72%. Both its recall and F1 performance can be explained by the fact that the measure was trained to optimize accuracy, and not the F1 score for the positive class; which also suggests that the approach could be adapted to F1 optimization if needed.

4.3 Synthesis

Canonical XF was more consistent than trained XF on all dimensions except accuracy, for which XF_t was optimized. We argue that this consistency was made possible through the asymmetry of XF which allowed it to adapt to different kinds of similarities (i.e., equivalence/paraphrase, inference/entailment, and mutual distance/ranking). These results also show that the actual position difference is a relevant factor for text similarity. We explain it mainly by the natural flow of language where the important entities and relations are often expressed first, in contrast with a purely logical-driven approach which has to consider, for instance, that active forms and passive forms are equivalent in meaning. The difference in positions is also not read literally, both because of the higher impact associated to missed words and to the α parameter which allows leveraging their impact in the trained version.

4.4 Additional Experiments

In additional experiments, we compared TF_c and TF_t with the other similarity measures when applied to bi-grams and tri-grams instead of individual tokens. The results were significantly lower on all datasets (between 3 and 10 points loss in accuracy) for both the soa measures and *TextFlow* variants. This result could be explained by the fact that n-grams are too rigid when a sub-sequence varies even slightly, e.g., the insertion of a new word inside a 3-words sequence leads to a tri-gram mismatch and reduces bi-gram overlap from 100% to 50% for the considered sub-sequence. This issue is not encountered with TextFlow as it relies on the token level, and such an insertion will not cancel or reduce significantly the gains from the correct ordering of the words. It must be noted here that not all languages grant the same level of importance to sequences and that additional multilingual tests have to be carried out.

In addition to binary classification output such as textual entailment and paraphrase recognition, text similarity measures can be evaluated more precisely when we consider the correlation of their values for ranking purposes.

We conducted ranking correlation experiments on three test datasets provided at the semantic text similarity task at Semeval 2012, with gold score values for their text pairs. The datasets have 750 sentence pairs each, and are extracted from

the Microsoft Research video descriptions corpus, MSRP and the SMTeuroparl[11]. When compared to the traditional similarity measures, *TextFlow* had the best correlation on the first two datasets with, for instance, 0.54 and 0.46 pearson correlation on the lemmas sequences and the second best on the MSRP extract where the Cosine similarity had the best performance with 0.71 vs 0.68, noting that the Cosine similarity uses word frequencies when the evaluated version of TextFlow did not use word-level weights.

Including word weights is one of the promising perspectives in line with this work as it could be done simply by making the deltas vary according to the weight/importance of the (un)matched word. Also, in such a setting, the impact of a sequence of N words will naturally increase or decrease according to the word weights (cf. figure 3). We conducted a preliminary test using the inverse document frequency of the words as extracted from Wikipedia with Gensim[12], which led to an improvement of up to 2% for most datasets, with performance decreasing slightly on two of them. Other kinds of weights could also be included just as easily, such as contextual word relatedness using embeddings or other semantic relatedness factors such as WordNet distances (Pedersen et al., 2004).

5 Conclusion

We presented a novel standalone similarity measure that takes into account continuous word sequences. An evaluation on eight datasets show promising results for textual entailment recognition, paraphrase detection and ranking. Among the potential extensions of this work are the inclusion of different kinds of weights such as TF-IDF, embedding relatedness and semantic relatedness. We also intend to test other variants around the same concept, including considering the matched words and sequences to have a negative weight to balance further the weight of missing words.

Acknowledgements

This work was supported in part by the Intramural Research Program of the NIH, National Library of Medicine.

[11] goo.gl/NVnybD

[12] https://radimrehurek.com/gensim/

References

Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. 2008. The evaluation of sentence similarity measures. In *Data warehousing and knowledge discovery*, Springer, pages 305–316.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* .

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, Springer, pages 177–190.

Lee R Dice. 1945. Measures of the amount of ecologic association between species. *Ecology* 26(3):297–302.

Carol Friedman and Robert Sideli. 1992. Tolerating spelling errors during patient validation. *Computers and Biomedical Research* 25(5):486–509.

Vasileios Hatzivassiloglou, Judith L Klavans, and Eleazar Eskin. 1999. Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. In *Proceedings of the 1999 joint sigdat conference on empirical methods in natural language processing and very large corpora*. Citeseer, pages 203–212.

Anil K Jain and Richard C Dubes. 1988. *Algorithms for clustering data*. Prentice-Hall, Inc.

Matthew A Jaro. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association* 84(406):414–420.

Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea, and Keeley Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE transactions on knowledge and data engineering* 18(8):1138–1150.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014* .

David W Mount. 2004. *Bioinformatics: sequence and genome analysis*. Cold Spring Harbor Laboratory Press.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the*

10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016. pages 525–545. http://aclweb.org/anthology/S/S16/S16-1083.pdf.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*. Association for Computational Linguistics, pages 38–41.

Edward H Porter, William E Winkler, et al. 1997. Approximate string comparison and its effect on an advanced record linkage system. In *Advanced record linkage system. US Bureau of the Census, Research Report*. Citeseer.

Mehran Sahami and Timothy D Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web*. AcM, pages 377–386.

David Sankoff and Joseph B Kruskal. 1983. Time warps, string edits, and macromolecules: the theory and practice of sequence comparison. *Reading: Addison-Wesley Publication, 1983, edited by Sankoff, David; Kruskal, Joseph B.* 1.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 373–382.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*. volume 24, pages 801–809.

Wen-Tau Yih and Christopher Meek. 2007. Improving similarity measures for short segments of text. In *AAAI*. volume 7, pages 1489–1494.

Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 247–256.

Friendships, Rivalries, and Trysts: Characterizing Relations between Ideas in Texts

Chenhao Tan* **Dallas Card†** **Noah A. Smith***
*Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA 98195, USA
chenhao@chenhaot.com
†School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA
dcard@cmu.edu
nasmith@cs.washington.edu

Abstract

Understanding how ideas relate to each other is a fundamental question in many domains, ranging from intellectual history to public communication. Because ideas are naturally embedded in texts, we propose the first framework to systematically characterize the relations between ideas based on their occurrence in a corpus of documents, independent of how these ideas are represented. Combining two statistics—*cooccurrence within documents* and *prevalence correlation over time*—our approach reveals a number of different ways in which ideas can cooperate and compete. For instance, two ideas can closely track each other's prevalence over time, and yet rarely cooccur, almost like a "cold war" scenario. We observe that pairwise cooccurrence and prevalence correlation exhibit different distributions. We further demonstrate that our approach is able to uncover intriguing relations between ideas through in-depth case studies on news articles and research papers.

1 Introduction

Ideas exist in the mind, but are made manifest in language, where they compete with each other for the scarce resource of human attention. Milton (1644) used the "marketplace of ideas" metaphor to argue that the truth will win out when ideas freely compete; Dawkins (1976) similarly likened the evolution of ideas to natural selection of genes. We propose a framework to quantitatively characterize competition and cooperation between ideas in texts, independent of how they might be represented.

By "ideas", we mean any discrete conceptual units that can be identified as being present or absent in a document. In this work, we consider representing ideas using keywords and topics obtained in an unsupervised fashion, but our way of characterizing the *relations* between ideas could be applied to many other types of textual representations, such as frames (Card et al., 2015) and hashtags.

What does it mean for two ideas to compete in texts, quantitatively? Consider, for example, the issue of immigration. There are two strongly competing narratives about the roughly 11 million people[1] who are residing in the United States without permission. One is "illegal aliens", who "steal" jobs and deny opportunities to legal immigrants; the other is "undocumented immigrants", who are already part of the fabric of society and deserve a path to citizenship (Merolla et al., 2013).

Although prior knowledge suggests that these two narratives compete, it is not immediately obvious what measures might reveal this competition in a corpus of writing about immigration. One question is whether or not these two ideas cooccur in the same documents. In the example above, these narratives are used by distinct groups of people with different ideologies. The fact that they don't cooccur is one clue that they may be in competition with each other.

However, cooccurrence is insufficient to express the selection process of ideas, i.e., some ideas fade out over time, while others rise in popularity, analogous to the populations of species in nature. Of the two narratives on immigration, we may expect one to win out at the expense of another as public opinion shifts. Alternatively, we might expect to see these narratives reinforcing each other, as both sides intensify their messaging in response to growing opposition, much like the U.S.S.R. and

[1] As of 2014, according to the most recent numbers from the Center for Migration Studies (Warren, 2016).

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 773–783
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1072

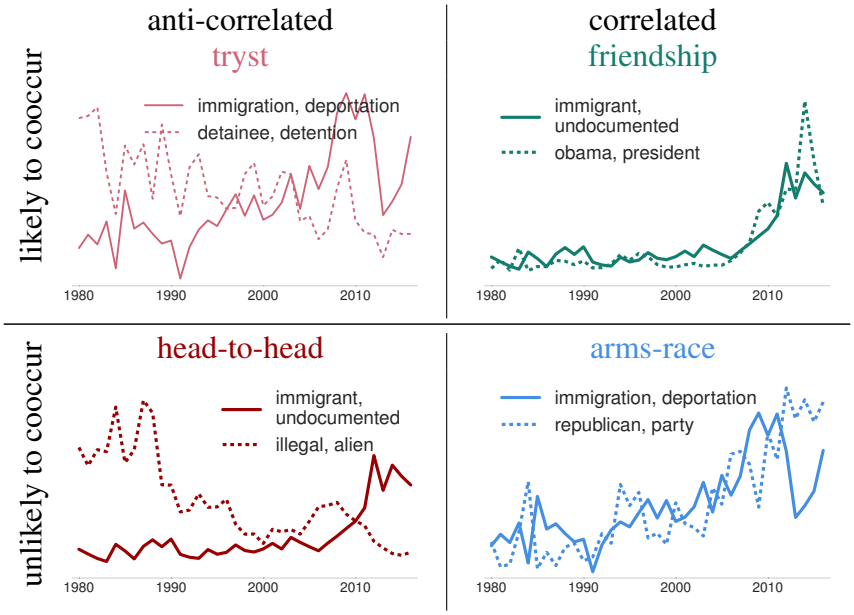


Figure 1: Relations between ideas in the space of cooccurrence and prevalence correlation (prevalence correlation is shown explicitly and cooccurrence is encoded in row captions). We use topics from LDA (Blei et al., 2003) to represent ideas. Each topic is named with a pair of words that are most strongly associated with the topic in LDA. Subplots show examples of relations between topics found in U.S. newspaper articles on immigration from 1980 to 2016, color coded to match the description in text. The y-axis represents the proportion of news articles in a year (in our corpus) that contain the corresponding topic. All examples are among the top 3 strongest relations in each type except ("immigrant, undocumented", "illegal, alien"), which corresponds to the two competing narratives. We explain the formal definition of strength in §2.

the U.S. during the cold war. To capture these possibilities, we use prevalence correlation over time.

Building on these insights, we propose a framework that combines cooccurrence within documents and prevalence correlation over time. This framework gives rise to four possible types of relation that correspond to the four quadrants in Fig. 1. We explain each type using examples from news articles in U.S. newspapers on immigration from 1980 to 2016. Here, we have used LDA to identify ideas in the form of topics, and we denote each idea with a pair of words most strongly associated with the corresponding topic.

Friendship (correlated over time, likely to cooccur). The "immigrant, undocumented" topic tends to cooccur with "obama, president" and both topics have been rising during the period of our dataset, likely because the "undocumented immigrants" narrative was an important part of Obama's framing of the immigration issue (Haynes et al., 2016).

Head-to-head (anti-correlated over time, unlikely to cooccur). "immigrant, undocumented" and "illegal, alien" are in a head-to-head competition: these two topics rarely cooccur, and "immigrant, undocumented" has been growing in prevalence, while the usage of "illegal, alien" in newspapers has been declining. This observation agrees with a report from Pew Research Center (Guskin, 2013).

Tryst (anti-correlated over time, likely to cooccur). The two off-diagonal examples use topics related to law enforcement. Overall, "immigration, deportation" and "detention, jail" often cooccur but "detention, jail" has been declining, while "immigration, deportation" has been rising. This possibly relates to the promises to overhaul the immigration detention system (Kalhan, 2010).[2]

Arms-race (correlated over time, unlikely to cooccur). One of the above law enforcement topics ("immigration, deportation") and a topic on the Republican party ("republican, party") hold an arms-race relation: they are both growing in prevalence over time, but rarely cooccur, perhaps suggesting an underlying common cause.

[2]The tryst relation is the least intuitive, yet is nevertheless observed. The pattern of being anti-correlated yet likely to cooccur is typically found when two ideas exhibit a friendship pattern (cooccurring and correlated), but only briefly, and then diverge.

Note that our terminology describes the relations between ideas in texts, not necessarily between the entities to which the ideas refer. For example, we find that the relation between "Israel" and "Palestine" is "friendship" in news articles on terrorism, based on their prevalence correlation and cooccurrence in that corpus.

We introduce the formal definition of our framework in §2 and apply it to news articles on five issues and research papers from ACL Anthology and NIPS as testbeds. We operationalize ideas using topics (Blei et al., 2003) and keywords (Monroe et al., 2008).

To explore whether the four relation types exist and how strong these relations are, we first examine the marginal and joint distributions of cooccurrence and prevalence correlation (§3). We find that cooccurrence shows a unimodal normal-shaped distribution but prevalence correlation demonstrates more diverse distributions across corpora. As we would expect, there are, in general, more and stronger friendship and head-to-head relations than arms-race and tryst relations.

Second, we demonstrate the effectiveness of our framework through in-depth case studies (§4). We not only validate existing knowledge about some news issues and research areas, but also identify hypotheses that require further investigation. For example, using keywords to represent ideas, a top pair with the tryst relation in news articles on terrorism is "arab" and "islam"; they are likely to cooccur, but "islam" is rising in relative prevalence while "arab" is declining. This suggests a conjecture that the news media have increasingly linked terrorism to a religious group rather than an ethnic group. We also show relations between topics in ACL that center around machine translation.

Our work is a first step towards understanding relations between ideas from text corpora, a complex and important research question. We provide some concluding thoughts in §6.

2 Computational Framework

The aim of our computational framework is to explore *relations* between ideas. We thus assume that the set of relevant ideas has been identified, and those expressed in each document have been tabulated. Our open-source implementation is at `https://github.com/Noahs-ARK/idea_relations/`. In the following, we introduce our formal definitions and datasets.

$$\forall x, y \in \mathcal{I}, \widehat{\mathrm{PMI}}(x,y) = \log \frac{\hat{P}(x,y)}{\hat{P}(x)\hat{P}(y)}$$
$$= C + \log \frac{1+\sum_t \sum_k \mathbf{1}\{x \in d_{t_k}\} \cdot \mathbf{1}\{y \in d_{t_k}\}}{\left(1+\sum_t \sum_k \mathbf{1}\{x \in d_{t_k}\}\right) \cdot \left(1+\sum_t \sum_k \mathbf{1}\{y \in d_{t_k}\}\right)} \quad (1)$$

$$\hat{r}(x,y) = \frac{\sum_t \left(\hat{P}(x|t) - \overline{\hat{P}(x|t)}\right)\left(\hat{P}(y|t) - \overline{\hat{P}(y|t)}\right)}{\sqrt{\sum_t \left(\hat{P}(x|t) - \overline{\hat{P}(x|t)}\right)^2} \sqrt{\sum_t \left(\hat{P}(y|t) - \overline{\hat{P}(y|t)}\right)^2}} \quad (2)$$

Figure 2: Eq. 1 is the empirical pointwise mutual information for two ideas, our measure of cooccurrence of ideas; note that we use add-one smoothing in estimating PMI. Eq. 2 is the Pearson correlation between two ideas' prevalence over time.

2.1 Cooccurrence and Prevalence Correlation

As discussed in the introduction, we focus on two dimensions to quantify relations between ideas:

1. cooccurrence reveals to what extent two ideas tend to occur in the same contexts;

2. similarity between the relative prevalence of ideas over time reveals how two ideas relate in terms of popularity or coverage.

Our input is a collection of documents, each represented by a set of ideas and indexed by time. We denote a *static* set of ideas as $\mathcal{I}$ and a text corpus that consists of these ideas as $C = \{D_1, \ldots, D_T\}$, where $D_t = \{d_{t_1}, \ldots, d_{t_{N_t}}\}$ gives the collection of documents at timestep t, and each document, d_{t_k}, is represented as a subset of ideas in $\mathcal{I}$. Here T is the total number of timesteps, and N_t is the number of documents at timestep t. It follows that the total number of documents $N = \sum_{t=1}^{T} N_t$.

In order to formally capture the two dimensions above, we employ two commonly-used statistics. First, we use empirical pointwise mutual information (PMI) to capture the cooccurrence of ideas within the same document (Church and Hanks, 1990); see Eq. 1 in Fig. 2. Positive $\widehat{\mathrm{PMI}}$ indicates that ideas occur together more frequently than would be expected if they were independent, while negative $\widehat{\mathrm{PMI}}$ indicates the opposite.

Second, we compute the correlation between normalized document frequency of ideas to capture the relation between the relative prevalence of ideas across documents over time; see Eq. 2 in Fig. 2. Positive $\hat{r}$ indicates that two ideas have similar prevalence over time, while negative $\hat{r}$ sug-

gests two anti-correlated ideas (i.e., when one goes up, the other goes down).

The four types of relations in the introduction can now be obtained using $\widehat{\text{PMI}}$ and $\hat{r}$, which capture cooccurrence and prevalence correlation respectively. We further define the *strength* of the relation between two ideas as the absolute value of the product of their $\widehat{\text{PMI}}$ and $\hat{r}$ scores:

$$\forall x, y \in \mathcal{I}, \text{strength}(x, y) = |\widehat{\text{PMI}}(x, y) \times \hat{r}(x, y)|. \quad (3)$$

2.2 Datasets and Representation of Ideas

We use two types of datasets to validate our framework: news articles and research papers. We choose these two domains because competition between ideas has received significant interest in history of science (Kuhn, 1996) and research on framing (Chong and Druckman, 2007; Entman, 1993; Gitlin, 1980; Lakoff, 2014). Furthermore, interesting differences may exist in these two domains as news evolves with external events and scientific research progresses through innovations.

- News articles. We follow the strategy in Card et al. (2015) to obtain news articles from LexisNexis on five issues: abortion, immigration, same-sex marriage, smoking, and terrorism. We search for relevant articles using LexisNexis subject terms in U.S. newspapers from 1980 to 2016. Each of these corpora contains more than 25,000 articles. Please refer to the supplementary material for details.
- Research papers. We consider full texts of papers from two communities: our own ACL community captured by papers from ACL, NAACL, EMNLP, and TACL from 1980 to 2014 (Radev et al., 2009); and the NIPS community from 1987 to 2016.[3] There are 4.8K papers from the ACL community and 6.6K papers from the NIPS community. The processed datasets are available at `https://chenhaot.com/pages/idea-relations.html`.

In order to operationalize ideas in a text corpus, we consider two ways to represent ideas.

- Topics. We extract topics from each document by running LDA (Blei et al., 2003) on each corpus C. In all datasets, we set the number of topics to 50.[4] Formally, $\mathcal{I}$ is the 50 topics learned from the corpus, and each document is represented as the set of topics that are present with greater than 0.01 probability in the topic distribution for that document.
- Keywords. We identify a list of distinguishing keywords for each corpus by comparing its word frequencies to the background frequencies found in other corpora using the informative Dirichlet prior model in Monroe et al. (2008). We set the number of keywords to 100 for all corpora. For news articles, the background corpus for each issue is comprised of all articles from the other four issues. For research papers, we use NIPS as the background corpus for ACL and vice versa to identify what are the core concepts for each of these research areas. Formally, $\mathcal{I}$ is the 100 top distinguishing keywords in the corpus and each document is represented as the set of keywords within $\mathcal{I}$ that are present in the document. Refer to the supplementary material for a list of example keywords in each corpus.

In both procedures, we lemmatize all words and add common bigram phrases to the vocabulary following Mikolov et al. (2013). Note that in our analysis, ideas are only present or absent in a document, and a document can in principle be mapped to any subset of ideas in $\mathcal{I}$. In our experiments 90% of documents are marked as containing between 7 and 14 ideas using topics, 8 and 33 ideas using keywords.

3 Characterizing the Space of Relations

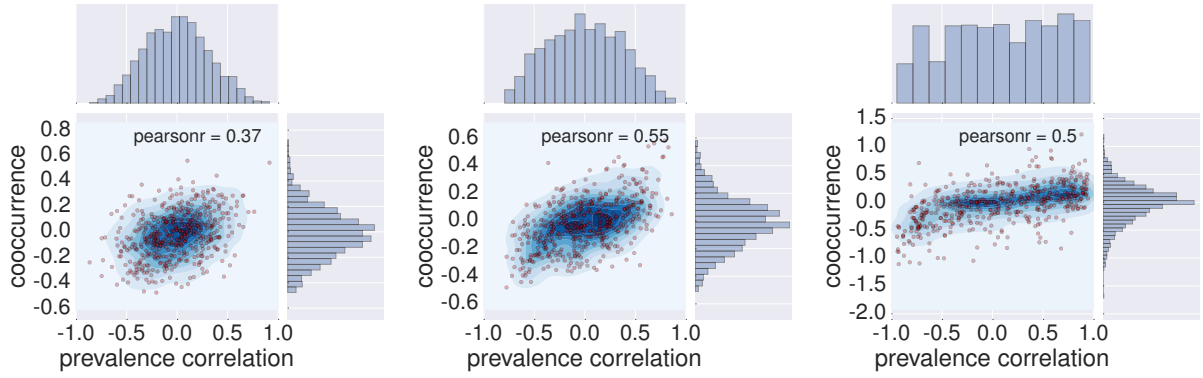
To provide an overview of the four relation types in Fig. 1, we first examine the empirical distributions of the two statistics of interest across pairs of ideas. In most exploratory studies, however, we are most interested in pairs that exemplify each type of relation, i.e., the most extreme points in each quadrant. We thus look at these pairs in each corpus to observe how the four types differ in salience across datasets.

3.1 Empirical Distribution Properties

To the best of our knowledge, the distributions of pairwise cooccurrence and prevalence correlation have not been examined in previous literature. We thus first investigate the marginal distributions of cooccurrence and prevalence correlation and then

[3] `http://papers.nips.cc/`.

[4] We chose 50 topics based on past experience, though this could be tuned for particular applications. Recall that our framework is to analyze *relations* between ideas, so this choice is not essential in this work.



(a) Terrorism topics (b) Immigration topics (c) ACL topics

Figure 3: Overall distributions of cooccurrence and prevalence correlation. In the main plot, each point represents a pair of ideas: color density shows the kernel density estimation of the joint distribution (Scott, 2015). The plots along the axes show the marginal distribution of the corresponding dimension. In each plot, we give the Pearson correlation, and all Pearson correlations' p-values are less than 10^{-40}. In these plots, we use topics to represent ideas.

their joint distribution. Fig. 3 shows three examples: two from news articles and one from research papers. We will also focus our case studies on these three corpora in §4. The corresponding plots for keywords have been relegated to supplementary material due to space limitations.

Cooccurrence tends to be unimodal but not normal. In all of our datasets, pairwise cooccurrence ($\widehat{\text{PMI}}$) presents a unimodal distribution that somewhat resembles a normal distribution, but it is rarely precisely normal. We cannot reject the hypothesis that it is unimodal for any dataset (using topics or keywords) using the dip test (Hartigan and Hartigan, 1985), though D'Agostino's K^2 test (D'Agostino et al., 1990) rejects normality in almost all cases.

Prevalence correlation exhibits diverse distributions. Pairwise prevalence correlation follows different distributions in news articles compared to research papers: they are unimodal in news articles, but not in ACL or NIPS. The dip test only rejects the unimodality hypothesis in NIPS. None follow normal distributions based on D'Agostino's K^2 test.

Cooccurrence is positively correlated with prevalence correlation. In all of our datasets, cooccurrence is positively correlated with prevalence correlation whether we use topics or keywords to represent ideas, although the Pearson correlation coefficients vary. This suggests that there are more friendship and head-to-head relations than tryst and arms-race relations. Based on the results of kernel density estimation, we also observe that this correlation is often loose, e.g., in ACL topics, cooccurrence spreads widely at each mode of prevalence correlation.

3.2 Relative Strength of Extreme Pairs

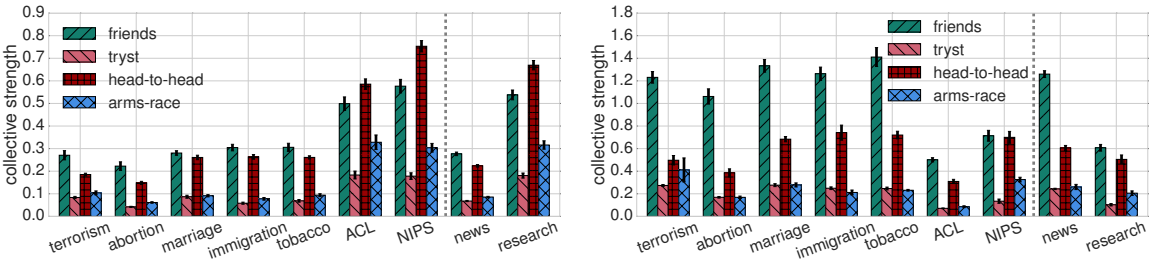
We are interested in how our framework can identify intriguing relations between ideas. These potentially interesting pairs likely correspond to the extreme points in each quadrant instead of the ones around the origin, where PMI and prevalence correlation are both close to zero. Here we compare the relative strength of extreme pairs in each dataset. We will discuss how these extreme pairs confirm existing knowledge and suggest new hypotheses via case studies in §4.

For each relation type, we average the strengths of the 25 pairs with the strongest relations in that type, with strength defined in Eq. 3. This heuristic (henceforth *collective strength*) allows us to collectively compare the strengths of the most prominent friendship, tryst, arms-race, and head-to-head relations. The results are not sensitive to the choice of 25.

Fig. 4 shows the collective strength of the four types in all of our datasets. The most common ordering is:

$$\text{friendship} > \text{head-to-head} > \text{arms-race} > \text{tryst}.$$

The fact that friendship and head-to-head relations are strong is consistent with the positive correlation between cooccurrence and prevalence correlation. In news, friendship is the strongest relation type, but head-to-head is the strongest in ACL topics and NIPS topics. This suggests, unsurprisingly, that there are stronger head-to-head competitions



(a) Topics

(b) Keywords

Figure 4: Collective strength of the four relation types in each dataset (*news* is the average of the news corpora and *research* is for ACL and NIPS). Fig. 4a uses topics to represent ideas, while Fig. 4b uses keywords to represent ideas. Each bar presents the average strength of the top 25 pairs in a relation type in the corresponding dataset. Error bars represent standard errors calculated in the usual way, but note that since the top 25 pairs are not random samples, they cannot be interpreted in the usual way.

(i.e., one idea takes over another) between ideas in scientific research than in news. We also see that topics show greater strength in our scientific article collections, while keywords dominate in news, especially in friendship. We conjecture that terms in scientific literature are often overloaded (e.g., a *tree* could be a parse tree or a decision tree), necessitating some abstraction when representing ideas. In contrast, news stories are more self-contained and seek to employ consistent usage.

4 Exploratory Studies

We present case studies based on strongly related pairs of ideas in the four types of relation. Throughout this section, "rank" refers to the rank of the relation strength between a pair of ideas in its corresponding relation type.

4.1 International Relations in Terrorism

Following a decade of declining violence in the 90s, the events of September 11, 2001 precipitated a dramatic increase in concern about terrorism, and a major shift in how it was framed (Kern et al., 2003). As a showcase, we consider a topic which encompasses much of the U.S. government's response to terrorism: "federal, state".[5] We observe two topics engaging in an "arms race" with this one: "afghanistan, taliban" and "pakistan, india". These correspond to two geopolitical regions closely linked to the U.S. government's concern with terrorism, and both were sites of U.S. military action during the period of our dataset. Events abroad and the U.S. government's responses follow the arms-race pattern, each holding increasing

[5]As in §1, we summarize each topic using a pair of strongly associated words, instead of assigning a name.

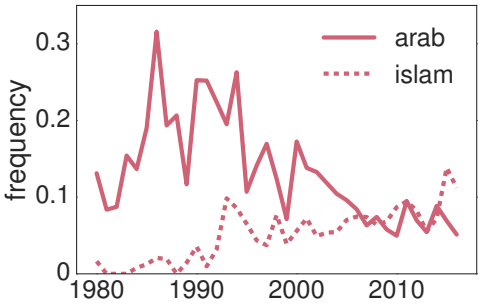
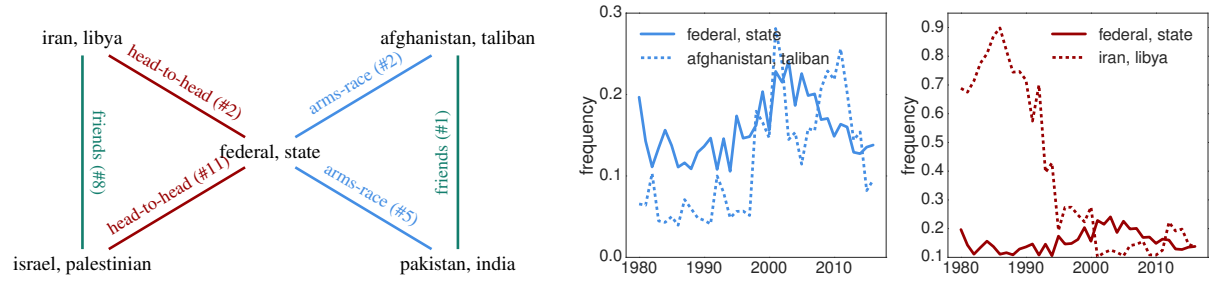


Figure 6: Tryst relation between arab and islam using keywords to represent ideas (#2 in tryst): these two words tend to cooccur but are anti-correlated in prevalence over time. In particular, islam was rarely used in coverage of terrorism in the 1980s.

attention with the other, likely because they share the same underlying cause.

We also observe two head-to-head rivals to the "federal, state" topic: "iran, libya" and "israel, palestinian". While these topics correspond to regions that are hotly debated in the U.S., their coverage in news tends not to correlate temporally with the U.S. government's responses to terrorism, at least during the time period of our corpus. Discussion of these regions was more prevalent in the 80s and 90s, with declining media coverage since then (Kern et al., 2003).

The relations between these topics are consistent with structural balance theory (Cartwright and Harary, 1956; Heider, 1946), which suggests that the enemy of an enemy is a friend. The "afghanistan, taliban" topic has the strongest friendship relation with the "pakistan, india" topic, i.e., they are likely to cooccur and are positively correlated in prevalence. Similarly, the "iran, libya" topic is a close "friend" with the "israel, palestinian" topic (ranked 8th in friendship).



(a) Relations between a United States topic and international topics.

(b) ("federal, state", "afghanistan, taliban")

(c) ("federal, state", "iran, libya")

Figure 5: Fig. 5a shows the relations between the "federal, state" topic and four international topics. Edge colors indicate relation types and the number in an edge label presents the ranking of its strength in the corresponding relation type. Fig. 5b and Fig. 5c represent concrete examples in Fig. 5a: "federal, state" and "afghanistan, taliban" follow similar trends, although "afghanistan, taliban" fluctuates over time due to significant events such as the September 11 attacks in 2001 and the death of Bin Laden in 2011; while "iran, lybia" is negatively correlated with "federal, state". In fact, more than 70% of terrorism news in the 80s contained the "iran, lybia" topic.

When using keywords to represent ideas, we observe similar relations between the term homeland security and terms related to the above foreign countries. In addition, we highlight an interesting but unexpected tryst relation between arab and islam (Fig. 6). It is not surprising that these two words tend to cooccur in the same news articles, but the usage of islam in the news is increasing while arab is declining. The increasing prevalence of islam and decreasing prevalence of arab over this time period can also be seen, for example, using Google's n-gram viewer, but it of course provides no information about cooccurrence.

This trend has not been previously noted to the best of our knowledge, although an article in the *Huffington Post* called for news editors to distinguish Muslim from Arab.[6] Our observation suggests a conjecture that the news media have increasingly linked terrorism to a religious group rather than an ethnic group, perhaps in part due to the tie between the events of 9/11 and Afghanistan, which is not an Arab or Arabic-speaking country. We leave it to further investigation to confirm or reject this hypothesis.

To further demonstrate the effectiveness of our approach, we compare a pair's rank using only cooccurrence or prevalence correlation with its rank in our framework. Table 1 shows the results for three pairs above. If we had looked at only cooccurrence or prevalence correlation, we would probably have missed these interesting pairs.

[6] http://www.huffingtonpost.com/haroon-moghul/even-the-new-york-times-d_b_766658.html

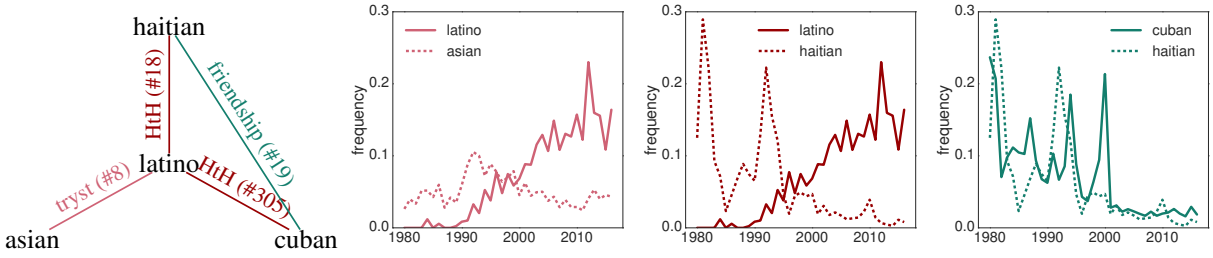
	PMI	Corr
"federal, state", "afghanistan, taliban" (#2 in arms-race)	43	99
"federal, state", "iran, lybia" (#2 in head-to-head)	36	56
arab, islam (#2 in tryst)	106	1,494

Table 1: Ranks of pairs by using the absolute value of only cooccurrence or prevalence correlation.

4.2 Ethnicity Keywords in Immigration

In addition to results on topics in §1, we observe unexpected patterns about ethnicity keywords in immigration news. Our observation starts with a top tryst relation between latino and asian. Although these words are likely to cooccur, their prevalence trajectories differ, with the discussion of Asian immigrants in the 1990s giving way to a focus on the word latino from 2000 onward. Possible theories to explain this observation include that undocumented immigrants are generally perceived as a Latino issue, or that Latino voters are increasingly influential in U.S. elections.

Furthermore, latino holds head-to-head relations with two subgroups of Latin American immigrants: haitian and cuban. In particular, the strength of the relation with haitian is ranked #18 in head-to-head relations. Meanwhile, haitian and cuban have a friendship relation, which is again consistent with structural balance theory. The decreasing prevalence of haitian and cuban perhaps speaks to the shifting geographical focus of recent immigration to the U.S., and issues of the Latino pan-ethnicity. In fact, a majority of Latinos prefer to identify with their national origin relative to the



(a) Relations graph. (b) (latino, asian) (c) (latino, haitian) (d) (cuban, haitian)

Figure 7: Relations between ethnicity keywords in immigration news (HtH for head-to-head): latino holds a tryst relation with asian and head-to-head relations with two subgroups from Latin America, haitian and cuban. We do not show the relations between asian and haitian, cuban, because their strength is close to 0.

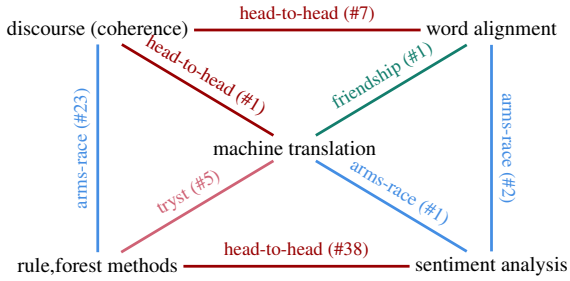


Figure 8: Top relations between the topics in ACL Anthology. The top 10 words for the rule, forest methods topic are *rule, grammar, derivation, span, algorithm, forest, parsing, figure, set, string*.

pan-ethnic terms (Taylor et al., 2012). However, we should also note that much of this coverage relates to a set of specific refugee crises, temporarily elevating the political importance of these nations in the U.S. Nevertheless, the underlying social and political reasons behind these head-to-head relations are worth further investigation.

4.3 Relations between Topics in ACL

Finally, we analyze relations between topics in the ACL Anthology. It turns out that "machine translation" is at a central position among top ranked relations in all the four types (Fig. 8).[7] It is part of the strongest relation in all four types except tryst (ranked #5).

The full relation graph presents further patterns. Friendship demonstrates transitivity: both "machine translation" and "word alignment" have similar relations with other topics. But such transitivity does not hold for tryst: although the prevalence of "rule, forest methods" is anti-correlated with both "machine translation" and "sentiment analysis", "sentiment analysis" seldom cooccurs with "rule, forest methods" because "sentiment analysis" is seldom built on parsing algorithms. Similarly, "rule, forest methods" and "discourse (coherence)" hold an arms-race relation: they do not tend to cooccur and both decline in relative prevalence as "machine translation" rises.

The prevalence of each of these ideas in comparison to machine translation is shown in in Fig. 9, which reveals additional detail.

5 Related Work

We present two strands of related studies in addition to what we have discussed.

Trends in ideas. Most studies have so far examined the trends of ideas individually (Michel et al., 2011; Hall et al., 2008; Rule et al., 2015). For instance, Hall et al. (2008) present various trends in our own computational linguistics community, including the rise of statistical machine translation. More recently, rhetorical framing has been used to predict these sorts of patterns (Prabhakaran et al., 2016). An exception is that Shi et al. (2010) use prevalence correlation to analyze lag relations between topics in publications and research grants. Anecdotally, Grudin (2009) observes a "head-to-head" relation between artificial intelligence and human-computer interaction in research funding. However, to our knowledge, our work is the first study to systematically characterize relations *between* ideas.

Representation of ideas. In addition to topics and keywords, studies have also sought to operationalize the "memes" metaphor using quotes and text reuse in the media (Leskovec et al., 2009; Niculae et al., 2015; Smith et al., 2013; Wei et al., 2013). In topic modeling literature, Blei and Lafferty (2006) also point out that topics do not cooccur independently and explicitly model the cooccurrence within documents.

[7] In the ranking, we filtered a topic where the top words are *ion, ing, system, process, language, one, input, natural language, processing, grammar*. For the purposes of this corpus, this is effectively a stopword topic.

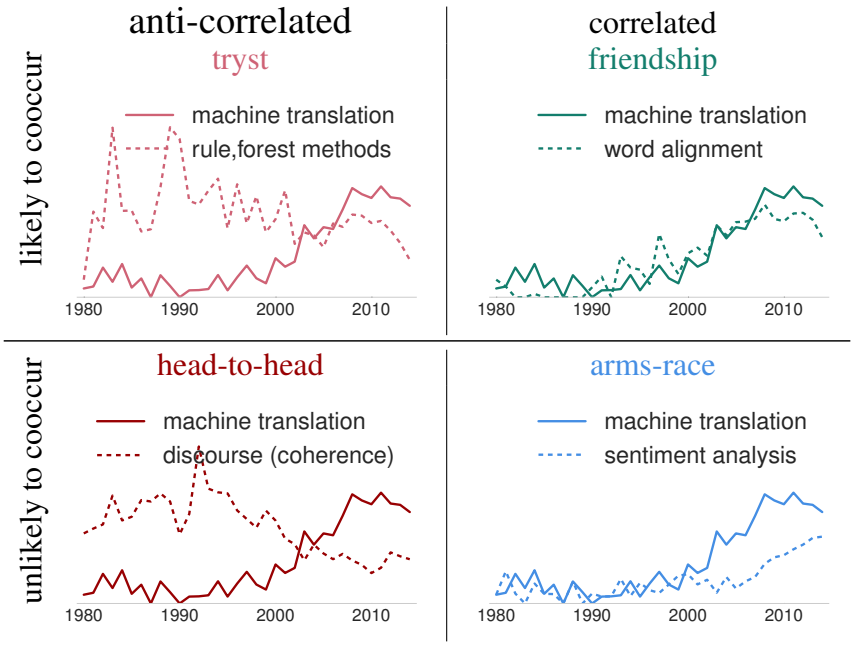


Figure 9: Relations between topics in ACL Anthology in the space of cooccurrence and prevalence correlation (prevalence correlation is shown explicitly and cooccurrence is encoded in row captions), color coded to match the text. The y-axis represents the relative proportion of papers in a year that contain the corresponding topic. The top 10 words for the rule, forest methods topic are *rule, grammar, derivation, span, algorithm, forest, parsing, figure, set, string*.

6 Concluding Discussion

We proposed a method to characterize relations between ideas in texts through the lens of cooccurrence within documents and prevalence correlation over time. For the first time, we observe that the distribution of pairwise cooccurrence is unimodal, while the distribution of pairwise prevalence correlation is not always unimodal, and show that they are positively correlated. This combination suggests four types of relations between ideas, and these four types are all found to varying extents in our experiments.

We illustrate our computational method by exploratory studies on news corpora and scientific research papers. We not only confirm existing knowledge but also suggest hypotheses around the usage of arab and islam in terrorism and latino and asian in immigration.

It is important to note that the relations found using our approach depend on the nature of the representation of ideas and the source of texts. For instance, we cannot expect relations found in news articles to reflect shifts in public opinion if news articles do not effectively track public opinion.

Our method is entirely observational. It remains as a further stage of analysis to understand the underlying reasons that lead to these relations between ideas. In scientific research, for example, it could simply be the progress of science, i.e., newer ideas overtake older ones deemed less valuable at a given time; on the other hand, history suggests that it is not always the correct ideas that are most expressed, and many other factors may be important. Similarly, in news coverage, underlying sociological and political situations have significant impact on which ideas are presented, and how.

There are many potential directions to improve our method to account for complex relations between ideas. For instance, we assume that both ideas and relations are statically grounded in keywords or topics. In reality, ideas and relations both evolve over time: a tryst relation might appear as friendship if we focus on a narrower time period. Similarly, new ideas show up and even the same idea may change over time and be represented by different words.

Acknowledgments. We thank Amber Boydstun, Justin Gross, Lillian Lee, anonymous reviewers, and all members of Noah's ARK for helpful comments and discussions. This research was made possible by a Natural Sciences and Engineering Research Council of Canada Postgraduate Scholarship (to D.C.) and a University of Washington Innovation Award.

References

David M. Blei and John Lafferty. 2006. Correlated topic models. In *NIPS*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The Media Frames Corpus: Annotations of frames across issues. In *Proceedings of ACL*.

Dorwin Cartwright and Frank Harary. 1956. Structural balance: A generalization of Heider's theory. *Psychological Review* 63(5):277.

Dennis Chong and James N. Druckman. 2007. A theory of framing and opinion formation in competitive elite environments. *Journal of Communication* 57(1):99–118.

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

Ralph B. D'Agostino, Albert Belanger, and Ralph B. D'Agostino Jr. 1990. A suggestion for using powerful and informative tests of normality. *The American Statistician* 44(4):316–321.

Richard Dawkins. 1976. *The Selfish Gene*. Oxford University Press.

Robert M. Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of Communication* 43(4):51–58.

Todd Gitlin. 1980. *The Whole World is Watching: Mass Media in the Making and Unmaking of the New Left*. Berkeley: University of California Press.

Jonathan Grudin. 2009. AI and HCI: Two fields divided by a common focus. *AI Magazine* 30(4):48.

Emily Guskin. 2013. 'Illegal', 'undocumented', 'unauthorized': News media shift language on immigration. Pew Research Center.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of EMNLP*.

John A. Hartigan and P. M. Hartigan. 1985. The dip test of unimodality. *The Annals of Statistics* pages 70–84.

Chris Haynes, Jennifer L. Merolla, and S. Karthick Ramakrishnan. 2016. *Framing Immigrants: News Coverage, Public Opinion, and Policy*. Russell Sage Foundation.

Fritz Heider. 1946. Attitudes and cognitive organization. *The Journal of Psychology* 21(1):107–112.

Anil Kalhan. 2010. Rethinking immigration detention. *Columbia Law Review Sidebar* 110:42.

Montague Kern, Marion Just, and Pippa Norris. 2003. The lessons of framing terrorism. In Pippa Norris, Montague Kern, and Marion Just, editors, *Framing Terrorism: The News Media, the Government and the Public*, Routledge.

Thomas S. Kuhn. 1996. *The Structure of Scientific Revolutions*. University of Chicago Press.

George Lakoff. 2014. *The All New Don't Think of an Elephant!: Know your Values and Frame the Debate*. Chelsea Green Publishing.

Jure Leskovec, Lars Backstrom, and Jon M. Kleinberg. 2009. Meme-tracking and the dynamics of the news cycle. In *Proceedings of KDD*.

Jennifer Merolla, S. Karthick Ramakrishnan, and Chris Haynes. 2013. "Illegal,", "undocumented," or "unauthorized": Equivalency frames, issue frames, and public opinion on immigration. *Perspectives on Politics* 11(03):789–807.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science* 331(6014):176–182.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

John Milton. 1644. *Areopagitica, A speech of Mr. John Milton for the Liberty of Unlicenc'd Printing to the Parliament of England*.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis* 16(4):372–403.

Vlad Niculae, Caroline Suen, Justine Zhang, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Quotus: The structure of political media coverage as revealed by quoting patterns. In *Proceedings of WWW*.

Vinodkumar Prabhakaran, William L. Hamilton, Dan McFarland, and Dan Jurafsky. 2016. Predicting the rise and fall of scientific topics from trends in their rhetorical framing. In *Proceedings of ACL*.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The ACL anthology network corpus. In *Proceedings of ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*.

Alix Rule, Jean-Philippe Cointet, and Peter S. Bearman. 2015. Lexical shifts, substantive changes, and

continuity in state of the union discourse, 1790-2014. *Proceedings of the National Academy of Sciences* 112(35):10837–10844.

David W. Scott. 2015. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons.

Xiaolin Shi, Ramesh Nallapati, Jure Leskovec, Dan McFarland, and Dan Jurafsky. 2010. Who leads whom: Topical lead-lag analysis across corpora. In *Proceedings of NIPS Workshop on Computational Social Science*.

David A. Smith, Ryan Cordell, and Elizabeth M. Dillon. 2013. Infectious texts: Modeling text reuse in nineteenth-century newspapers. In *Proceedings of the Workshop on Big Humanities*.

Paul Taylor, Mark H. Lopez, Jessica Martínez, and Gabriel Velasco. 2012. When labels don't fit: Hispanics and their views of identity. *Washington, DC: Pew Hispanic Center* .

Robert Warren. 2016. US undocumented population drops below 11 million in 2014, with continued declines in the Mexican undocumented population. *Journal on Migration and Human Security* 4(1):1–15.

Xuetao Wei, Nicholas Valler, B. Aditya Prakash, Iulian Neamtiu, Michalis Faloutsos, and Christos Faloutsos. 2013. Competing memes propagation on networks: A network science perspective. *IEEE Journal on Selected Areas in Communications* 31:1049–1060.

Polish evaluation dataset for compositional distributional semantics models

Alina Wróblewska **Katarzyna Krasnowska-Kieraś**
Institute of Computer Science, Polish Academy of Sciences
alina@ipipan.waw.pl kasia.krasnowska@gmail.com

Abstract

The paper presents a procedure of building an evaluation dataset[1]. for the validation of compositional distributional semantics models estimated for languages other than English. The procedure generally builds on steps designed to assemble the SICK corpus, which contains pairs of English sentences annotated for semantic relatedness and entailment, because we aim at building a comparable dataset. However, the implementation of particular building steps significantly differs from the original SICK design assumptions, which is caused by both lack of necessary extraneous resources for an investigated language and the need for language-specific transformation rules. The designed procedure is verified on Polish, a fusional language with a relatively free word order, and contributes to building a Polish evaluation dataset. The resource consists of 10K sentence pairs which are human-annotated for semantic relatedness and entailment. The dataset may be used for the evaluation of compositional distributional semantics models of Polish.

1 Introduction and related work

1.1 Distributional semantics

The basic idea of distributional semantics, i.e. determining the meaning of a word based on its co-occurrence with other words, is derived from the empiricists – Harris (1954) and Firth (1957). John R. Firth drew attention to the context-dependent nature of meaning especially with his famous maxim "You shall know a word by the company it keeps" (Firth, 1957, p. 11).

Nowadays, distributional semantics models are estimated with various methods, e.g. word embedding techniques (Bengio et al., 2003, 2006; Mikolov et al., 2013). To ascertain the purport of a word, e.g. *bath*, you can use the context of other words that surround it. If we assume that the meaning of this word expressed by its lexical context is associated with a distributional vector, the distance between distributional vectors of two semantically similar words, e.g *bath* and *shower*, should be smaller than between vectors representing semantically distinct words, e.g. *bath* and *tree*.

1.2 Compositional distributional semantics

Based on empirical observations that distributional vectors encode certain aspects of word meaning, it is expected that similar aspects of the meaning of phrases and sentences can also be represented with vectors obtained via composition of distributional word vectors. The idea of semantic composition is not new. It is well known as the *principle of compositionality*:[2] "The meaning of a compound expression is a function of the meaning of its parts and of the way they are syntactically combined." (Janssen, 2012, p. 19).

Modelling the meaning of textual units larger than words using compositional and distributional information is the main subject of compositional distributional semantics (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011; Socher et al., 2012, to name a few studies). The fundamental principles of compositional distributional semantics, henceforth referred to as CDS, are mainly propagated with papers written on the topic. Apart from the papers, it was the SemEval-2014 Shared Task 1

[1] The dataset is obtainable at: http://zil.ipipan.waw.pl/Scwad/CDSCorpus

[2] As the principle of compositionality is attributed to Gottlob Frege, it is often called *Frege's principle*.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 784–792
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1073

(Marelli et al., 2014) that essentially contributed to the expansion of CDS and increased an interest in this domain. The goal of the task was to evaluate CDS models of English in terms of semantic relatedness and entailment on proper sentences from the SICK corpus.

1.3 The SICK corpus

The SICK corpus (Bentivogli et al., 2014) consists of 10K pairs of English sentences containing multiple lexical, syntactic, and semantic phenomena. It builds on two external data sources – the 8K ImageFlickr dataset (Rashtchian et al., 2010) and SemEval-2012 Semantic Textual Similarity dataset (Agirre et al., 2012). Each sentence pair is human-annotated for relatedness in meaning and entailment.

The relatedness score corresponds to the degree of semantic relatedness between two sentences and is calculated as the average of ten human ratings collected for this sentence pair on the 5-point Likert scale. This score indicates the extent to which the meanings of two sentences are related.

The entailment relation between two sentences, in turn, is labelled with *entailment*, *contradiction*, or *neutral*. According to the SICK guidelines, the label assigned by the majority of human annotators is selected as the valid entailment label.

1.4 Motivation and organisation of the paper

Studying approaches to various natural language processing (henceforth NLP) problems, we have observed that the availability of language resources (e.g. training or testing data) stimulates the development of NLP tools and the estimation of NLP models. English is undoubtedly the most prominent in this regard and English resources are the most numerous. Therefore, NLP methods are mostly designed for English and tested on English data, even if there is no guarantee that they are universal. In order to verify whether an NLP algorithm is adequate, it is not enough to evaluate it solely for English. It is also valuable to have high-quality resources for languages typologically different to English. Hence, we aim at building datasets for the evaluation of CDS models in languages other than English, which are often underresourced. We strongly believe that the availability of test data will encourage development of CDS models in these languages and allow to better test the universality of CDS methods.

We start with a high-quality dataset for Polish, which is a completely different language than English in at least two dimensions. First, it is a rather under-resourced language in contrast to the resource-rich English. Second, it is a fusional language with a relatively free word order in contrast to the isolated English with a relatively fixed word order. If some heuristics is tested on e.g. Polish, the evaluation results can be approximately generalised to other Slavic languages. We hope the Slavic NLP community will be interested in designing and evaluating methods of semantic modelling for Slavic languages.

The procedure of building an evaluation dataset for validating compositional distributional semantics models of Polish generally builds on steps designed to assemble the SICK corpus (described in Section 1.3) because we aim at building an evaluation dataset which is comparable to the SICK corpus. However, the implementation of particular building steps significantly differs from the original SICK design assumptions, which is caused by both lack of necessary extraneous resources for Polish (see Section 2.1) and the need for Polish-specific transformation rules (see Section 2.2). Furthermore, the rules of arranging sentences into pairs (see Section 2.3) are defined anew taking into account the characteristic of data and bidirectional entailment annotations, since an entailment relation between two sentences must not be symmetric. Even if our assumptions of annotating sentence pairs coincide with the SICK principles to a certain extent (see Section 3.1), the annotation process differs from the SICK procedure, in particular by introducing an element of human verification of correctness of automatically transformed sentences (see Section 3.2) and some additional post-corrections (see Section 3.3). Finally, a summary of the dataset is provided in Section 4.1 and the dataset evaluation is given in Section 4.2.

2 Procedure of collecting data

2.1 Selection and description of images

The first step of building the SICK corpus consisted in the random selection of English sentence pairs from existing datasets (Rashtchian et al., 2010; Agirre et al., 2012). Since we are not aware of accessibility of analogous resources for Polish, we have to select images first and then describe the selected images.

Images are selected from the 8K ImageFlickr

dataset (Rashtchian et al., 2010). At first we wanted to take only these images the descriptions of which were selected for the SICK corpus. However, a cursory check shows that these images are quite homogeneous, with a predominant number of dogs depictions. Therefore, we independently extract 1K images and split them into 46 thematic groups (e.g. *children*, *musical instruments*, *motorbikes*, *football*, *dogs*). The numbers of images within individual thematic groups vary from 6 images in the *volleyball* and *telephoning* groups to 94 images in the *various people* group. The second largest groups are *children* and *dogs* with 50 images each.

The chosen images are given to two authors who independently of each other formulate their descriptions based on a short instruction. The authors are instructed to write one single sentence (with a sentence predicate) describing the action in a displayed image. They should not describe an imaginable context or an interpretation of what may lie behind the scene in the picture. If some details in the picture are not obvious, they should not be described either. Furthermore, the authors should avoid multiword expressions, such as idioms, metaphors, and named entities, because those are not compositional linguistic phenomena. Finally, descriptions should contain Polish diacritics and proper punctuation.

2.2 Transformation of descriptions

The second step of building the SICK corpus consisted in pre-processing extracted sentences, i.e. normalisation and expansion (Bentivogli et al., 2014, p. 3–4). Since the authors of Polish descriptions are asked to follow the guidelines (presented in Section 2.1), the normalisation step is not essential for our data. The expansion step, in turn, is implemented and the sentences provided by the authors are lexically and syntactically transformed in order to obtain derivative sentences with similar, contrastive, or neutral meanings. The following transformations are implemented:

1. *dropping conjunction* concerns sentences with coordinated predicates sharing a subject, e.g. *Rowerzysta odpoczywa i obserwuje morze.* (Eng. ‘A cyclist is resting and watching the sea.’). The finite form of one of the coordinated predicates is transformed into:
    - an active adjectival participle, e.g. *Odpoczywający rowerzysta obserwuje morze.* (Eng. ‘A resting cyclist is watching the sea.’) or *Obserwujący morze rowerzysta odpoczywa.* (Eng. ‘A cyclist, who is watching the sea, is resting.’),
    - a contemporary adverbial participle, e.g. *Rowerzysta, odpoczywając, obserwuje morze.* (Eng. ‘A cyclist is watching the sea, while resting.’) or *Rowerzysta odpoczywa, obserwując morze.* (Eng. ‘A cyclist is resting, while watching the sea.’).
2. *removing conjunct in adjuncts*, i.e. the deletion of one of coordinated elements of an adjunct, e.g. *Mały, ale zwinny kot miauczy.* (Eng. ‘A small but agile cat miaows.’) can be changed into either *Mały kot miauczy.* (Eng. ‘A small cat miaows.’) or *Zwinny kot miauczy.* (Eng. ‘An agile cat miaows.’).
3. *passivisation*, e.g. *Człowiek ujeżdża byka.* (Eng. ‘A man is breaking a bull in.’) can be transformed into *Byk jest ujeżdżany przez człowieka.* (Eng. ‘A bull is being broken in by a man.’).
4. *removing adjuncts*, e.g. *Dwa białe króliki siedzą na trawie.* (Eng. ‘Two small rabbits are sitting on the grass.’) can be changed into *Króliki siedzą.* (Eng. ‘The rabbits are sitting.’).
5. *swapping relative clause for participles*, i.e. a relative clause swaps with a participle (and vice versa), e.g. *Kobieta przytula psa, którego trzyma na smyczy.* (Eng. ‘A woman hugs a dog which she keeps on a leash.’). The relative clause is interchanged for a participle construction, e.g. *Kobieta przytula trzymanego na smyczy psa.* (Eng. ‘A woman hugs a dog kept on a leash.’).
6. *negation*, e.g. *Mężczyźni w turbanach na głowach siedzą na słoniach.* (Eng. ‘Men in turbans on their heads are sitting on elephants.’) can be transformed into *Nikt nie siedzi na słoniach.* (Eng. ‘Nobody is sitting on elephants.’), *Żadni mężczyźni w turbanach na głowach nie siedzą na słoniach.* (Eng. ‘No men in turbans on their heads are sitting on elephants.’), and *Mężczyźni w turbanach na głowach nie siedzą na słoniach.* (Eng. ‘Men in turbans on their heads are not sitting on elephants.’).

7. *constrained mixing of dependents from various sentences*, e.g. *Dwoje dzieci siedzi na wielbłądach w pobliżu wysokich gór.* (Eng. 'Two children are sitting on camels near high mountains.') can be changed into *Dwoje dzieci siedzi przy zastawionym stole w pobliżu wysokich gór.* (Eng. 'Two children are sitting at the table laid with food near high mountains.').

The first five transformations are designed to produce sentences with a similar meaning, the sixth transformation outputs sentences with a contradictory meaning, and the seventh transformation should generate sentences with a neutral (or unrelated) meaning. All transformations are performed on the dependency structures of input sentences (Wróblewska, 2014).

Some of the transformations are very productive (e.g. mixing dependents). Other, in turn, are sparsely represented in the output (e.g. dropping conjunction). The number of transformed sentences randomly selected to build the dataset is in the second column of Table 1.

transformation	**selected**	
dropping conjunction	139	2.0%
removing conjunct in adjunct	485	6.9%
passivisation	893	12.8%
removing adjuncts	1013	14.5%
swapping rc↔ptcp	1291	18.4%
negation	1304	18.6%
mixing dependents	1878	26.8%

Table 1: Numbers of transformed sentences selected for annotation.

2.3 Data ensemble

The final step of building the SICK corpus consisted in arranging normalised and expanded sentences into pairs. Since our data diverges from SICK data, the process of arranging Polish sentences into pairs also differs from pairing in the SICK corpus. The general idea behind the pair-ensembling procedure was to introduce sentence pairs with different levels of relatedness into the dataset. Apart from pairs connecting two sentences originally written by humans (as described in Section 2.1), there are also pairs in which an original sentence is connected with a transformed sentence. For each of the 1K images, the following 10 pairs are constructed (for A being the set of all sentences originally written by the first author, B being the set of all sentences originally written by the second author, $\mathbf{a} \in A$ and $\mathbf{b} \in B$ being the original descriptions of the picture):

1. $(\mathbf{a}, \mathbf{b})$
2. $(\mathbf{a}, \mathbf{a}_1)$, where $\mathbf{a}_1 \in t(\mathbf{a})$, and $t(\mathbf{a})$ is the set of all transformations of the sentence $\mathbf{a}$
3. $(\mathbf{b}, \mathbf{b}_1)$, where $\mathbf{b}_1 \in t(\mathbf{b})$
4. $(\mathbf{a}, \mathbf{b}_2)$, where $\mathbf{b}_2 \in t(\mathbf{b})$
5. $(\mathbf{b}, \mathbf{a}_2)$, where $\mathbf{a}_2 \in t(\mathbf{a})$
6. $(\mathbf{a}, \mathbf{a}_3)$, where $\mathbf{a}_3 \in t(\mathbf{a}'), \mathbf{a}' \in A, \mathcal{T}(\mathbf{a}') = \mathcal{T}(\mathbf{a}), \mathbf{a}' \neq \mathbf{a}$, for $\mathcal{T}(\mathbf{a})$ being the thematic group[3] of $\mathbf{a}$
7. $(\mathbf{b}, \mathbf{b}_3)$, where $\mathbf{b}_3 \in t(\mathbf{b}'), \mathbf{b}' \in B, \mathcal{T}(\mathbf{b}') = \mathcal{T}(\mathbf{b}), \mathbf{b}' \neq \mathbf{b}$
8. $(\mathbf{a}, \mathbf{a}_4)$, where $\mathbf{a}_4 \in A, \mathcal{T}(\mathbf{a_4}) \neq \mathcal{T}(\mathbf{a})$[4]
9. $(\mathbf{b}, \mathbf{b}_4)$, where $\mathbf{b_4} \in B, \mathcal{T}(\mathbf{b_4}) \neq \mathcal{T}(\mathbf{b})$
10. $(\mathbf{a}, \mathbf{a}_5)$, where $\mathbf{a}_5 \in t(\mathbf{a}), \mathbf{a}_5 \neq \mathbf{a}_1$ for 50% images, $(\mathbf{b}, \mathbf{b}_5)$ (analogously) for other 50%.[5]

For each sentence pair $(\mathbf{a}, \mathbf{b})$ created according to this procedure, its reverse $(\mathbf{b}, \mathbf{a})$ is also included in our corpus. As a result, the working set consists of 20K sentence pairs.

3 Corpus annotation

3.1 Annotation assumptions

The degree of semantic relatedness between two sentences is calculated as the average of all human ratings on the Likert scale with the range from 0 to 5. Since we do not want to excessively influence

[3]The thematic group of a sentence $\mathbf{a}$ corresponds to the thematic group of an image being the source of $\mathbf{a}$ (as described in Section 2.1).

[4]The pairs $(\mathbf{a}, \mathbf{a}_4)$ of the same authors' descriptions of two images from different thematic groups are expected to be unrelated. The same applies to $(\mathbf{b}, \mathbf{b}_4)$.

[5]A repetition of point 2 with a restriction that a different pair is created (pairs of very related sentences are expected). We alternate between authors A and B to obtain equal author proportions in the final ensemble of pairs.

the annotations, the guidelines given to annotators are mainly example-based:[6]

- 5 (very related): *Kot siedzi na płocie.* (Eng. 'A cat is sitting on the fence.') vs. *Na płocie jest duży kot.* (Eng. 'There is a large cat on the fence.'),

- 1–4 (more or less related):
  Kot siedzi na płocie. (Eng. 'A cat is sitting on the fence.') vs. *Kot nie siedzi na płocie.* (Eng. 'A cat is not sitting on the fence.');
  Kot siedzi na płocie. (Eng. 'A cat is sitting on the fence.') vs. *Właściciel dał kotu chrupki.* (Eng. 'The owner gave kibble to his cat.');
  Kot siedzi na płocie. (Eng. 'A cat is sitting on the fence.') vs. *Kot miauczy pod płotem.* (Eng. 'A cat miaows by the fence.').

- 0 (unrelated): *Kot siedzi na płocie.* (Eng. 'A cat is sitting on the fence.') vs. *Zaczął padać deszcz.* (Eng. 'It started to rain.').

Apart from these examples, there is a note in the annotation guidelines indicating that the degree of semantic relatedness is not equivalent to the degree of semantic similarity. Semantic similarity is only a special case of semantic relatedness, semantic relatedness is thus a more general term than the other one.

Polish entailment labels correspond directly to the SICK labels (i.e. *entailment*, *contradiction*, *neutral*). The entailment label assigned by the majority of human judges is selected as the gold label. The entailment labels are defined as follows:

- **a wynika z b** (**b** entails **a**) – if a situation or an event described by sentence **b** occurs, it is recognised that a situation or an event described by **a** occurs as well, i.e. **a** and **b** refer to the same event or the same situation,

- **a jest zaprzeczeniem b** (**a** is the negation of **b**) – if a situation or an event described by **b** occurs, it is recognised that a situation or an event described by **a** may not occur at the same time,

- **a jest neutralne wobec b** (**a** is neutral to **b**) – the truth of a situation described by **a** cannot be determined on the basis of **b**.

3.2 Annotation procedure

Similar to the SICK corpus, each Polish sentence pair is human-annotated for semantic relatedness and entailment by 3 human judges experienced in Polish linguistics.[7] Since for each annotated pair $(\mathbf{a}, \mathbf{b})$, its reverse $(\mathbf{b}, \mathbf{a})$ is also subject to annotation, the entailment relation is in practice determined 'in both directions' for 10K sentence pairs. For the task of relatedness annotation, the order of sentences within pairs seems to be irrelevant, we can thus assume to obtain 6 relatedness scores for 10K unique pairs.

Since the transformation process is fully automatic and to a certain extent based on imperfect dependency parsing, we cannot ignore errors in the transformed sentences. In order to avoid annotating erroneous sentences, the annotation process is divided into two stages:

1. a sentence pair is sent to a judge with the *leader* role, who is expected to edit and to correct the transformed sentence from this pair before annotation, if necessary,

2. the verified and possibly enhanced sentence pair is sent to the other two judges, who can only annotate it.

The *leader* judges should correct incomprehensible and ungrammatical sentences with a minimal number of necessary changes. Unusual sentences which could be accepted by Polish speakers should not be modified. Moreover, the modified sentence may not be identical with the other sentence in the pair. The classification and statistics of distinct corrections made by the *leader* judges are provided in Table 2.

A strict classification of error types is quite hard to provide because some sentences contain more than one error. We thus order the error types from the most serious errors (i.e. 'sense' errors) to the redundant corrections (i.e. 'other' type). If a sentence contains several errors, it is qualified for the higher order error type.

In the case of sentences with 'sense' errors, the need for correction is uncontroversial and

[6] We realise that the boundary between semantic perception of a sentence by various speakers is fuzzy (it depends on speakers' education, origin, age, etc.). It was thus our well-thought-out decision to draw only general annotation frames and to enable annotators to rely on their feel for language.

[7] Our annotators have relatively strong linguistic background. Five of them have PhD in linguistics, five are PhD students, one is a graduate, and one is an undergraduate.

error type	# of errors	% of errors
sense	171	12.3
semantic	407	29.2
grammatical	243	17.4
word order	141	10.1
punctuation	366	26.2
other	68	4.9

Table 2: Classification and statistics of corrections.

arises from an internal logical contradiction.[8] The sentences with 'semantic' changes are syntactically correct, but deemed unacceptable by the *leader* annotators from the semantic or pragmatic point of view.[9] The 'grammatical' errors mostly concern missing agreement.[10] The majority of 'word order' corrections are unnecessary, but we found some examples which can be classified as actual word or phrase order errors.[11] The correction of punctuation consists in adding or deleting a comma.[12] The sentences in the 'other' group, in turn, could as well have been left unchanged because they are proper Polish sentences, but were apparently considered odd by the *leader* annotators.

[8] An example of 'sense' error: the sentence *Chłopak w zielonej bluzie i czapce zjeżdża na rolkach* ***na leżąco***. (Eng. 'A boy in a green sweatshirt and a cap roller-skates downhill **in a lying position**.') is corrected into *Chłopak w zielonej bluzie i czapce zjeżdża na rolkach.* (Eng. 'A boy in a green sweatshirt and a cap roller-skates downhill.').

[9] An example of 'semantic' correction: the sentence *Dziewczyna trzyma w* ***pysku*** *patyk.* (Eng. 'A girl holds a stick in her **muzzle**.') is corrected into *Dziewczyna trzyma w* ***ustach*** *patyk.* (Eng. 'A girl holds a stick in her **mouth**.').

[10] An example of 'grammatical' error: the sentence *Grupa*$_{\mathbf{sg}.nom}$ *uśmiechających się ludzi* ***tańczą***$_{pl}$. (Eng. *'A group of smiling people are dancing.') is corrected into *Grupa*$_{\mathbf{sg}.nom}$ *uśmiechających się ludzi* ***tańczy***$_{sg}$. (Eng. 'A group of smiling people is dancing.').

[11] An example of word order error: the sentence *Samochód, który jest uszkodzony,* ***koloru białego*** *stoi na lawecie dużego auta.* (lit. 'A car that is damaged, **of the white color** stands on the trailer of a large car.', Eng. 'A **white** car that is damaged is standing on the trailer of a large car.') is corrected into *Samochód* ***koloru białego****, który jest uszkodzony, stoi na lawecie dużego auta.*

[12] An example of punctuation correction: the wrong comma in the sentence *Nad brzegiem wody, stoją dwaj mężczyźni z wędkami.* (lit. 'On the water's edge, two men are standing with rods.'; Eng. 'Two men with rods are standing on the water's edge.') should be deleted, i.e. *Nad brzegiem wody stoją dwaj mężczyźni z wędkami.*

3.3 Impromptu post-corrections

During the annotation process it came out that sentences accepted by some human annotators are unacceptable for other annotators. We thus decided to garner annotators' comments and suggestions for improving sentences. After validation of these suggestions by an experienced linguist, it turns out that most of these proposals concern punctuation errors (e.g. missing comma) and typos in 312 distinct sentences. These errors are fixed directly in the corpus because they should not impact the annotations of sentence pairs. The other suggestions concern more significant changes in 29 distinct sentences (mostly minor grammatical or semantic problems overlooked by the *leader* annotators). The annotations of pairs with modified sentences are resent to the annotators so that they can verify and update them.

4 Corpus summary and evaluation

4.1 Corpus statistics

Tables 3 and 4 summarise the annotations of the resulting 10K sentence pairs corpus. Table 3 aggregates the occurrences of 6 possible relatedness scores, calculated as the mean of all 6 individual annotations, rounded to an integer.

relatedness	# of pairs
0	1978
1	1428
2	1082
3	2159
4	2387
5	966

Table 3: Final relatedness scores rounded to integers (total: 10K pairs).

Table 4 shows the number of the particular entailment labels in the corpus. Since each sentence pair is annotated for entailment in both directions, the final entailment label is actually a pair of two labels:

- *entailment+neutral* points to 'one-way' entailment,
- *contradiction+neutral* points to 'one-way' contradiction,
- *entailment+entailment*, *contradiction+contradiction*, and *neutral+neutral* point to equivalence.

While the actual corpus labels are ordered in the sense that there is a difference between e.g. *entailment+neutral* and *neutral+entailment* (the entailment occurs in different directions), we treat all labels as unordered for the purpose of this summary (e.g. *entailment+neutral* covers *neutral+entailment* as well, representing the same type of relation between two sentences).

entailment	**# of pairs**
neutral+neutral	6483
entailment+neutral	1748
entailment+entailment	933
contradiction+contradiction	721
contradiction+neutral	115

Table 4: Final entailment labels (total: 10K pairs).

4.2 Inter-annotator agreement

The standard measure of inter-annotator agreement in various natural language labelling tasks is Cohen's kappa (Cohen, 1960). However, this coefficient is designed to measure agreement between two annotators only. Since there are three annotators of each pair of ordered sentences, we decided to apply Fleiss' kappa[13] (Fleiss, 1971) designed for measuring agreement between multiple raters who give categorical ratings to a fixed number of items. An additional advantage of this measure is that different items can be rated by different human judges, which doesn't impact measurement. The normalised Fleiss' measure of inter-annotator agreement is:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$$

where the quantity $\bar{P} - \bar{P}_e$ measures the degree of agreement actually attained in excess of chance, while "[t]he quantity $1 - \bar{P}_e$ measures the degree of agreement attainable over and above what would be predicted by chance" (Fleiss, 1971, p. 379).

We recognise Fleiss' kappa as particularly useful for measuring inter-annotator agreement with respect to entailment labelling in our evaluation dataset. First, there are more than two raters. Second, entailment labels are categorically. Measured with Fleiss' kappa, there is an inter-annotator agreement of $\kappa = 0.734$ for entailment labels in Polish evaluation dataset, which is quite satisfactory as for a semantic labelling task.

Relative to semantic relatedness, the distinction in meaning of two sentences made by human judges is often very subtle. This is also reflected in the inter-annotator agreement scores measured with Fleiss' kappa. Inter-annotator agreement measured for six semantic relatedness groups corresponding to points on the Likert scale is quite low: $\kappa = 0.337$. If we measure inter-annotator agreement for three classes corresponding to the three relatedness groups from the annotation guidelines (see Section 3.1), i.e. <0>, <1, 2, 3, 4>, and <5>, the Fleiss' score is significantly higher: $\kappa = 0.543$. Hence, we conclude that Fleiss' kappa is not a reliable measure of inter-annotator agreement in relation to relatedness scores. Therefore, we decided to use Krippendorff's α instead.

Krippendorff's α (Krippendorff, 1980, 2013) is a coefficient appropriate for measuring the inter-annotator agreement of a dataset which is annotated with multiple judges and characterised by different magnitudes of disagreement and missing values. Krippendorff proposes distance metrics suitable for various scales: binary, nominal, interval, ordinal, and ratio. In ordinal measurement[14] the attributes can be rank-ordered, but distances between them do not have any meaning. Measured with Krippendorff's ordinal α, there is an inter-annotator agreement of $\alpha = 0.780$ for relatedness scores in the Polish evaluation dataset, which is quite satisfactory as well. Hence, we conclude that our dataset is a reliable resource for the purpose of evaluating compositional distributional semantics model of Polish.

5 Conclusions

The goal of this paper is to present the procedure of building a Polish evaluation dataset for the validation of compositional distributional semantics models. As we aim at building an evalua-

[13] As Fleiss' kappa is actually the generalisation of Scott's π (Scott, 1955), it is sometimes referred to as Fleiss' *multi-*π, cf. Artstein and Poesio (2008).

[14] Nominal measurement is useless for measuring agreement between relatedness scores ($\alpha = 0.340$ is the identical value as Fleiss' kappa, since all disagreements are considered equal). We also test interval measurement, in which the distance between the attributes does have meaning and an average of an interval variable is computed. The interval score measured for relatedness annotations is quite high $\alpha = 0.785$, but we doubt whether the distance between relatedness scores is meaningful in this case.

tion dataset which is comparable to the SICK corpus, the general assumptions of our procedure correspond to the design principles of the SICK corpus. However, the procedure of building the SICK corpus cannot be adapted without modifications. First, the Polish seed-sentences have to be written based on the images which are selected from 8K ImageFlickr dataset and split into thematic groups, since usable datasets are not publicly available. Second, since the process of transforming sentences seems to be language-specific, the linguistic transformation rules appropriate for Polish have to be defined from scratch. Third, the process of arranging Polish sentences into pairs is defined anew taking into account the data characteristic and bidirectional entailment annotations. The discrepancies relative to the SICK procedure also concern the annotation process itself. Since an entailment relation between two sentences must not be symmetric, each sentence pair is annotated for entailment in both directions. Furthermore, we introduce an element of human verification of correctness of automatically transformed sentences and some additional post-corrections.

The presented procedure of building a dataset was tested on Polish. However, it is very likely that the annotation framework will work for other Slavic languages (e.g. Czech with an excellent dependency parser).

The presented procedure results in building the Polish test corpus of relatively high quality, confirmed by the inter-annotator agreement coefficients of $\kappa = 0.734$ (measured with Fleiss' kappa) for entailment labels and of $\alpha = 0.780$ (measured with Krippendorff's ordinal alpha) for relatedness scores.

Acknowledgments

We would like to thank the reliable and tenacious annotators of our dataset: Alicja Dziedzic-Rawska, Bożena Itoya, Magdalena Król, Anna Latusek, Justyna Małek, Małgorzata Michalik, Agnieszka Norwa, Małgorzata Szajbel-Keck, Alicja Walichnowska, Konrad Zieliński, and some other. The research presented in this paper was supported by SONATA 8 grant no 2014/15/D/HS2/03486 from the National Science Centre Poland.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*. pages 385–393.

Ron Artstein and Massimo Poesio. 2008. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics* 34:557–596.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. pages 1183–1193.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research* 3:1137–1155.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural Probabilistic Language Models. In D.E. Holmes and L.C. Jain, editors, *Innovations in Machine Learning. Theory and Applications*, Springer-Verlag, Berlin Heidelberg, volume 194 of *Studies in Fuzziness and Soft Computing*, pages 137–186.

Luisa Bentivogli, Raffaella Bernardi, Marco Marelli, Stefano Menini, Marco Baroni, and Roberto Zamparelli. 2014. SICK through the SemEval Glasses. Lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *Journal of Language Resources and Evaluation* 50:95–124.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20:37–46.

John Rupert Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis. Special volume of the Philological Society* pages 1–32.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 75:378–382.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental Support for a Categorical Compositional Distributional Model of Meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*. pages 1394–1404.

Zellig Harris. 1954. Distributional structure. *Word* 10:146–162.

Theo M. V. Janssen. 2012. Compositionality: its historic context. In Wolfram Hinzen, Edouard Machery, and Markus Werning, editors, *The Oxford Handbook of Compositionality*, Oxford University Press, Studies in Fuzziness and Soft Computing, pages 19–46.

Klaus Krippendorff. 1980. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, Beverly Hills.

Klaus Krippendorff. 2013. *Content Analysis: An Introduction to Its Methodology*. Sage Publication, Thousand Oaks, 3rd edition.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. pages 1–8.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26. Proceedings of Neural Information Processing Systems 2013*. pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science* 34:1388–1429.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting Image Annotations Using Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. pages 139–147.

William A. Scott. 1955. Reliability of Content Analysis: The Case of Nominal Scale Coding. *Public Opinion Quarterly* 19:321–325.

Richard Socher, Brody Huval, Christopher Manning, and Andrew Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 1201–1211.

Alina Wróblewska. 2014. *Polish Dependency Parser Trained on an Automatically Induced Dependency Bank*. Ph.D. dissertation, Institute of Computer Science, Polish Academy of Sciences, Warsaw.

Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction

Christopher Bryant **Mariano Felice** **Ted Briscoe**
ALTA Institute
Computer Laboratory
University of Cambridge
Cambridge, UK
{cjb255, mf501, ejb}@cl.cam.ac.uk

Abstract

Until now, error type performance for Grammatical Error Correction (GEC) systems could only be measured in terms of recall because system output is not annotated. To overcome this problem, we introduce ERRANT, a grammatical ERRor ANnotation Toolkit designed to automatically extract edits from parallel original and corrected sentences and classify them according to a new, dataset-agnostic, rule-based framework. This not only facilitates error type evaluation at different levels of granularity, but can also be used to reduce annotator workload and standardise existing GEC datasets. Human experts rated the automatic edits as "Good" or "Acceptable" in at least 95% of cases, so we applied ERRANT to the system output of the CoNLL-2014 shared task to carry out a detailed error type analysis for the first time.

1 Introduction

Grammatical Error Correction (GEC) systems are often only evaluated in terms of overall performance because system hypotheses are not annotated. This can be misleading however, and a system that performs poorly overall may in fact outperform others at specific error types. This is significant because a robust *specialised* system is actually more desirable than a mediocre *general* system. Without an error type analysis however, this information is completely unknown.

The main aim of this paper is hence to rectify this situation and provide a method by which parallel error correction data can be automatically annotated with error type information. This not only facilitates error type evaluation, but can also be used to provide detailed error type feedback to non-native learners. Given that different corpora are also annotated according to different standards, we also attempted to standardise existing datasets under a common error type framework.

Our approach consists of two main steps. First, we automatically extract the edits between parallel original and corrected sentences by means of a linguistically-enhanced alignment algorithm (Felice et al., 2016) and second, we classify them according to a new, rule-based framework that relies solely on dataset-agnostic information such as lemma and part-of-speech. We demonstrate the value of our approach, which we call the ERRor ANnotation Toolkit (ERRANT)[1], by carrying out a detailed error type analysis of each system in the CoNLL-2014 shared task on grammatical error correction (Ng et al., 2014).

It is worth mentioning that despite an increased interest in GEC evaluation in recent years (Dahlmeier and Ng, 2012; Felice and Briscoe, 2015; Bryant and Ng, 2015; Napoles et al., 2015; Grundkiewicz et al., 2015; Sakaguchi et al., 2016), ERRANT is the only toolkit currently capable of producing error types scores.

2 Edit Extraction

The first stage of automatic annotation is *edit extraction*. Specifically, given an original and corrected sentence pair, we need to determine the start and end boundaries of any edits. This is fundamentally an alignment problem:

We	took	a	**guide**	tour	**on**		**center**	**city**	.
We	took	a	**guided**	tour	**of**	**the**	**city**	**center**	.

Table 1: A sample alignment between an original and corrected sentence (Felice et al., 2016).

[1] https://github.com/chrisjbryant/errant

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 793–805
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1074

The first attempt at automatic edit extraction was made by Swanson and Yamangil (2012), who simply used the Levenshtein distance to align parallel original and corrected sentences. As the Levenshtein distance only aligns individual tokens however, they also merged all adjacent non-matches in an effort to capture multi-token edits. Xue and Hwa (2014) subsequently improved on Swanson and Yamangil's work by training a maximum entropy classifier to predict whether edits should be merged or not.

Most recently, Felice et al. (2016) proposed a new method of edit extraction using a linguistically-enhanced alignment algorithm supported by a set of merging rules. More specifically, they incorporated various linguistic information, such as part-of-speech and lemma, into the cost function of the Damerau-Levenshtein[2] algorithm to make it more likely that tokens with similar linguistic properties aligned. This approach ultimately proved most effective at approximating human edits in several datasets (80-85% F_1), and so we use it in the present study.

3 Automatic Error Typing

Having extracted the edits, the next step is to assign them error types. While Swanson and Yamangil (2012) did this by means of maximum entropy classifiers, one disadvantage of this approach is that such classifiers are biased towards their particular training corpora. For example, a classifier trained on the First Certificate in English (FCE) corpus (Yannakoudakis et al., 2011) is unlikely to perform as well on the National University of Singapore Corpus of Learner English (NUCLE) (Dahlmeier and Ng, 2012) or vice versa, because both corpora have been annotated according to different standards (cf. Xue and Hwa (2014)). Instead, a dataset-agnostic error type classifier is much more desirable.

3.1 A Rule-Based Error Type Framework

To solve this problem, we took inspiration from Swanson and Yamangil's (2012) observation that most error types are based on part-of-speech (POS) categories, and wrote a rule to classify an edit based only on its automatic POS tags. We then added another rule to similarly differentiate between Missing, Unnecessary and Replacement errors depending on whether tokens were inserted, deleted or substituted. Finally, we extended our approach to classify errors that are not well-characterised by POS, such as Spelling or Word Order, and ultimately assigned all error types based solely on automatically-obtained, objective properties of the data.

In total, we wrote roughly 50 rules. While many of them are very straightforward, significant attention was paid to discriminating between different kinds of verb errors. For example, despite all having the same correction, the following sentences contain different types of common learner errors:

(a) He IS asleep now. [*IS* → *is*]: orthography

(b) He iss asleep now. [*iss* → *is*]: spelling

(c) He has asleep now. [*has* → *is*]: verb

(d) He being asleep now. [*being* → *is*]: form

(e) He was asleep now. [*was* → *is*]: tense

(f) He are asleep now. [*are* → *is*]: SVA

To handle these cases, we hence wrote the following ordered rules:

1. Are the lower case forms of both sides of the edit the same? (a)
2. Is the original token a real word? (b)
3. Do both sides of the edit have the same lemma? (c)
4. Is one side of the edit a gerund (VBG) or participle (VBN)? (d)
5. Is one side of the edit in the past tense (VBD)? (e)
6. Is one side of the edit in the 3rd person present tense (VBZ)? (f)

While the final three rules could certainly be reordered, we informally found the above sequence performed best during development. It is also worth mentioning that this is a somewhat simplified example and that there are additional rules to discriminate between auxiliary verbs, main verbs and multi verb expressions. Nevertheless, the above case exemplifies our approach, and a more complete description of all rules is provided with the software.

[2]Damerau-Levenshtein is an extension of Levenshtein that also handles transpositions; e.g. AB→BA

Code	Meaning	Description / Example
ADJ	Adjective	*big → wide*
ADJ:FORM	Adjective Form	Comparative or superlative adjective errors. *goodest → best, bigger → biggest, more easy → easier*
ADV	Adverb	*speedily → quickly*
CONJ	Conjunction	*and → but*
CONTR	Contraction	*n't → not*
DET	Determiner	*the → a*
MORPH	Morphology	Tokens have the same lemma but nothing else in common. *quick (adj) → quickly (adv)*
NOUN	Noun	*person → people*
NOUN:INFL	Noun Inflection	Count-mass noun errors. *informations → information*
NOUN:NUM	Noun Number	*cat → cats*
NOUN:POSS	Noun Possessive	*friends → friend's*
ORTH	Orthography	Case and/or whitespace errors. *Bestfriend → best friend*
OTHER	Other	Errors that do not fall into any other category (e.g. paraphrasing). *at his best → well, job → professional*
PART	Particle	*(look) in → (look) at*
PREP	Preposition	*of → at*
PRON	Pronoun	*ours → ourselves*
PUNCT	Punctuation	*! → .*
SPELL	Spelling	*genectic → genetic, color → colour*
UNK	Unknown	The annotator detected an error but was unable to correct it.
VERB	Verb	*ambulate → walk*
VERB:FORM	Verb Form	Infinitives (with or without "to"), gerunds (-ing) and participles. *to eat → eating, dancing → danced*
VERB:INFL	Verb Inflection	Misapplication of tense morphology. *getted → got, fliped → flipped*
VERB:SVA	Subject-Verb Agreement	*(He) have → (He) has*
VERB:TENSE	Verb Tense	Includes inflectional and periphrastic tense, modal verbs and passivization. *eats → ate, eats → has eaten, eats → can eat, eats → was eaten*
WO	Word Order	*only can → can only*

Table 2: The list of 25 main error categories in our new framework with examples and explanations.

3.2 A Dataset-Agnostic Classifier

One of the key strengths of a rule-based approach is that by being dependent only on automatic mark-up information, our classifier is entirely dataset independent and does not require labelled training data. This is in contrast with machine learning approaches which not only learn dataset specific biases, but also presuppose the existence of sufficient quantities of training data.

A second significant advantage of our approach is that it is also always possible to determine precisely why an edit was assigned a particular error category. In contrast, human and machine learning classification decisions are often much less transparent.

Finally, by being fully deterministic, our approach bypasses bias effects altogether and should hence be more consistent.

3.3 Automatic Markup

The prerequisites for our rule-based classifier are that each token in both the original and corrected sentence is POS tagged, lemmatized, stemmed and dependency parsed. We use spaCy[3] v1.7.3 for all but the stemming, which is performed by the Lancaster Stemmer in NLTK.[4] Since fine-grained POS tags are often too detailed for the purposes of error evaluation, we also map spaCy's Penn Treebank style tags to the coarser set of Universal Dependency tags.[5] We use the latest Hunspell GB-large word list[6] to help classify non-word errors. The marked-up tokens in an edit span are then input to the classifier and an error type is returned.

3.4 Error Categories

The complete list of 25 error types in our new framework is shown in Table 2. Note that most of them can be prefixed with 'M:', 'R:' or 'U:', depending on whether they describe a Missing, Replacement, or Unnecessary edit, to enable

[3] https://spacy.io/

[4] http://www.nltk.org/

[5] http://universaldependencies.org/tagset-conversion/en-penn-uposf.html

[6] https://sourceforge.net/projects/wordlist/files/speller/2017.01.22/

evaluation at different levels of granularity (see Appendix A for all valid combinations). This means we can choose to evaluate, for example, only *replacement* errors (anything prefixed by 'R:'), only *noun* errors (anything suffixed with 'NOUN') or only *replacement noun* errors ('R:NOUN'). This flexibility allows us to make more detailed observations about different aspects of system performance.

One caveat concerning error scheme design is that it is always possible to add new categories for increasingly detailed error types; for instance, we currently label [*could* → *should*] a tense error, when it might otherwise be considered a modal error. The reason we do not call it a modal error, however, is because it would then become less clear how to handle other cases such as [*can* → *should*] and [*has eaten* → *should eat*], which might be considered a more complex combination of modal and tense error. As it is impractical to create new categories and rules to differentiate between such narrow distinctions however, our final framework aims to be a compromise between informativeness and practicality.

3.5 Classifier Evaluation

As our new error scheme is based solely on automatically obtained properties of the data, there are no gold standard labels against which to evaluate classifier performance. For this reason, we instead carried out a small-scale manual evaluation, where we simply asked 5 GEC researchers to rate the appropriateness of the predicted error types for 200 randomly chosen edits in context (100 from FCE-test and 100 from CoNLL-2014) as "Good", "Acceptable" or "Bad". "Good' meant the chosen type was the most appropriate for the given edit, "Acceptable" meant the chosen type was appropriate, but probably not optimum, while "Bad" meant the chosen type was not appropriate for the edit. Raters were warned that the edit boundaries had been determined automatically and hence might be unusual, but that they should focus on the appropriateness of the error type regardless of whether they agreed with the boundary or not.

It is worth stating that the main purpose of this evaluation was not to evaluate the specific strengths and weaknesses of the classifier, but rather ascertain how well humans believed the predicted error types characterised each edit. GEC is known to be a highly subjective task (Bryant and Ng, 2015) and so we were more interested in overall judgements than specific disagreements.

Rater	Good	Acceptable	Bad
1	92.0%	4.0%	4.0%
2	89.5%	6.5%	4.0%
3	83.0%	13.0%	4.0%
4	84.5%	11.0%	4.5%
5	82.5%	15.5%	2.0%
OVERALL	**86.3%**	**10.0%**	**3.7%**

Table 3: The percent distribution for how each expert rated the appropriateness of the predicted error types. E.g. Rater 3 considered 83% of all predicted types to be "Good".

The results from this evaluation are shown in Table 3. Significantly, all 5 raters considered at least 95% of the predicted error types to be either "Good" or "Acceptable", despite the degree of noise introduced by automatic edit extraction. Furthermore, whenever raters judged an edit as "Bad", this could usually be traced back to a POS or parse error; e.g. [*ring* → *rings*] might be considered a NOUN:NUM or VERB:SVA error depending on whether the POS tagger considered both sides of the edit nouns or verbs. Inter-annotator agreement was also good at 0.724 κ_{free} (Randolph, 2005).

In contrast, although incomparable on account of the different metric and error scheme, the best results using machine learning were between 50-70% F_1 (Felice et al., 2016). Ultimately however, we believe the high scores awarded by the raters validates the efficacy of our rule-based approach.

4 Error Type Scoring

Having described how to automatically annotate parallel sentences with ERRANT, we now also have a method to annotate system hypotheses; this is the first step towards an error type evaluation. Since no scorer is currently capable of calculating error type performance however (Dahlmeier and Ng, 2012; Felice and Briscoe, 2015; Napoles et al., 2015), we instead built our own.

Fortunately, one benefit of explicitly annotating system hypotheses is that it makes evaluation much more straightforward. In particular, for each sentence, we only need to compare the edits in the hypothesis against the edits in each respective reference and measure the overlap. Any edit with the same span and correction in both files is hence a

true positive (TP), while unmatched edits in the hypothesis and references are false positives (FP) and false negatives (FN) respectively. These results can then be grouped by error type for the purposes of error type evaluation.

Finally, it is worth noting that this scorer is much simpler than other scorers in GEC which typically incorporate edit extraction or alignment directly into their algorithms. Our approach, on the other hand, treats edit extraction and evaluation as separate tasks.

4.1 Gold Reference vs. Auto Reference

Before evaluating an automatically annotated hypothesis against its reference, we must also address another mismatch: namely that hypothesis edits must be extracted and classified automatically, while reference edits are typically extracted and classified manually using a different framework. Since evaluation is now reduced to a straightforward comparison between two files however, it is especially important that the hypothesis and references are both processed in the same way. For instance, a hypothesis edit [*have eating* → *has eaten*] will not match the reference edits [*have* → *has*] and [*eating* → *eaten*] because the former is one edit while the latter is two edits, even though they equate to the same thing.

To solve this problem, we can reprocess the references in the same way as the hypotheses. In other words, we can apply ERRANT to the references such that each reference edit is subject to the same automatic extraction and classification criteria as each hypothesis edit. While it may seem unorthodox to discard gold reference information in favour of automatic reference information, this is necessary to minimise the difference between hypothesis and reference edits and also standardise error type annotations.

To show that automatic references are feasible alternatives to gold references, we evaluated each team in the CoNLL-2014 shared task using both types of reference with the M^2 scorer (Dahlmeier and Ng, 2012), the *de facto* standard of GEC evaluation, and our own scorer. Table 4 hence shows that there is little difference between the overall scores for each team, and we formally validated this hypothesis for precision, recall and $F_{0.5}$ by means of bootstrap significance testing (Efron and Tibshirani, 1993). Ultimately, we found no statistically significant difference between automatic and gold references (1,000 iterations, $p > .05$) which leads us to conclude that our automatic references are qualitatively as good as human references.

	M^2 Scorer		Our Scorer	
Team	**Gold**	**Auto**	**Gold**	**Auto**
AMU	35.01	35.05	31.95	32.25
CAMB	37.33	37.34	33.39	34.01
CUUI	36.79	37.59	33.32	34.64
IITB	5.90	5.96	5.67	5.74
IPN	7.09	7.68	5.86	6.14
NTHU	29.92	29.77	25.62	25.66
PKU	25.32	25.38	23.40	23.60
POST	30.88	31.01	27.54	27.99
RAC	26.68	26.88	22.83	23.15
SJTU	15.19	15.22	14.85	14.89
UFC	7.84	7.89	7.84	7.89
UMC	25.37	25.45	23.08	23.52

Table 4: Overall scores for each team in CoNLL-2014 using gold and auto references with both the M^2 scorer and our simpler edit comparison approach. All scores are in terms of $F_{0.5}$.

4.2 Comparison with the M^2 Scorer

Despite using the same metric, Table 4 also shows that the M^2 scorer tends to produce slightly higher $F_{0.5}$ scores than our own. This initially led us to believe that our scorer was underestimating performance, but we subsequently found that instead the M^2 scorer tends to overestimate performance (cf. Felice and Briscoe (2015) and Napoles et al. (2015)).

In particular, given a choice between matching [*have eating* → *has eaten*] from Annotator 1 or [*have* → *has*] and [*eating* → *eaten*] from Annotator 2, the M^2 scorer will always choose Annotator 2 because two true positives (TP) are worth more than one. Similarly, whenever the scorer encounters two false positives (FP) within a certain distance of each other,[7] it merges them and treats them as one false positive; e.g. [*is a cat* → *are a cats*] is selected over [*is* → *are*] and [*cat* → *cats*] even though these edits are best handled separately. In other words, the M^2 scorer exploits its dynamic edit boundary prediction to artificially maximise true positives and minimise false positives and hence produce slightly inflated scores.

[7] The distance is controlled by the *max_unchanged_words* parameter which is set to 2 by default.

	AMU			CAMB			CUUI			IITB		
Type	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
Missing	43.94	14.32	31.08	45.96	29.71	41.43	26.37	18.16	24.18	15.38	0.59	2.56
Replacement	37.22	26.92	34.57	37.53	28.12	35.18	45.90	22.98	38.27	29.85	1.49	6.22
Unnecessary	-	-	-	25.51	27.47	25.88	34.20	33.33	34.02	46.15	1.53	6.77

	IPN			NTHU			PKU			POST		
Type	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
Missing	2.86	0.29	1.04	34.33	11.39	24.47	33.33	4.37	14.34	31.14	13.13	24.44
Replacement	9.87	3.86	7.53	27.61	19.15	25.37	29.62	18.33	26.37	33.16	19.33	29.01
Unnecessary	0.00	0.00	0.00	34.76	15.97	28.14	0.00	0.00	0.00	26.32	32.84	27.40

	RAC			SJTU			UFC			UMC		
Type	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
Missing	1.52	0.27	0.79	62.50	4.44	17.28	-	-	-	40.08	23.57	35.16
Replacement	29.41	20.82	27.17	50.54	3.43	13.47	72.00	2.64	11.52	34.71	9.70	22.90
Unnecessary	0.00	0.00	0.00	17.65	11.36	15.89	-	-	-	16.86	17.17	16.92

Table 5: Precision, recall and $F_{0.5}$ for Missing, Unnecessary, and Replacement errors for each team. A dash indicates the team's system did not attempt to correct the given error type (TP+FP = 0).

5 CoNLL-2014 Shared Task Analysis

To demonstrate the value of ERRANT, we applied it to the data produced in the CoNLL-2014 shared task (Ng et al., 2014). Specifically, we automatically annotated all the system hypotheses and official reference files.[8] Although ERRANT can be applied to any dataset of parallel sentences, we chose to evaluate on CoNLL-2014 because it represents the largest collection of publicly available GEC system output. For more information about the systems in CoNLL-2014, we refer the reader to the shared task paper.

5.1 Edit Operation

In our first category experiment, we simply investigated the performance of each system in terms of Missing, Replacement and Unnecessary edits. The results are shown in Table 5 with additional information in Appendix B, Table 10.

The most surprising result is that five teams (AMU, IPN, PKU, RAC, UFC) failed to correct any unnecessary token errors at all. This is noteworthy because unnecessary token errors account for roughly 25% of all errors in the CoNLL-2014 test data and so failing to address them significantly limits a system's maximum performance. While the reason for this is clear in some cases, e.g. UFC's rule-based system was never designed to tackle unnecessary tokens (Gupta, 2014), it is less clear in others, e.g. there is no obvious reason why AMU's SMT system failed to learn when to delete tokens (Junczys-Dowmunt and Grundkiewicz, 2014). AMU's result is especially remarkable given that their system still came 3rd overall despite this limitation.

In contrast, CUUI's classifier approach (Rozovskaya et al., 2014) was the most successful at correcting not only unnecessary token errors, but also replacement token errors, while CAMB's hybrid MT approach (Felice et al., 2014) significantly outperformed all others in terms of missing token errors. It would hence make sense to combine these two approaches, and indeed recent research has shown this improves overall performance (Rozovskaya and Roth, 2016).

5.2 General Error Types

Table 6 shows precision, recall and $F_{0.5}$ for each of the error types in our proposed framework for each team in CoNLL-2014. As some error types are more common than others, we also provide the TP, FP and FN counts used to make this table in Appendix B, Table 11.

Overall, CAMB was the most successful team in terms of error types, achieving the highest F-score in 10 (out of 24) error categories, followed by AMU, who scored highest in 6 categories. All but 3 teams (IITB, IPN and POST) achieved the best score in at least 1 category, which suggests that different approaches to GEC complement different error types. Only CAMB attempted to correct at least 1 error from every category.

Other interesting observations we can make from this table include:

[8] http://www.comp.nus.edu.sg/~nlp/conll14st.html

		AMU	CAMB	CUUI	IITB	IPN	NTHU	PKU	POST	RAC	SJTU	UFC	UMC
ADJ	**P**	4.88	9.09	-	0.00	0.00	0.00	66.67	0.00	12.50	0.00	-	0.00
	R	6.67	13.89	-	0.00	0.00	0.00	7.14	0.00	3.57	0.00	-	0.00
	$F_{0.5}$	5.15	9.77	-	0.00	0.00	0.00	**25.00**	0.00	8.33	0.00	-	0.00
ADJ:FORM	**P**	55.56	75.00	100.00	100.00	0.00	33.33	100.00	50.00	8.00	-	-	100.00
	R	62.50	60.00	33.33	40.00	0.00	37.50	28.57	14.29	40.00	-	-	60.00
	$F_{0.5}$	56.82	71.43	71.43	76.92	0.00	34.09	66.67	33.33	9.52	-	-	**88.24**
ADV	**P**	6.67	11.54	0.00	0.00	0.00	0.00	0.00	-	0.00	4.76	-	8.77
	R	2.94	20.45	0.00	0.00	0.00	0.00	0.00	-	0.00	3.03	-	12.50
	$F_{0.5}$	5.32	**12.64**	0.00	0.00	0.00	0.00	0.00	-	0.00	4.27	-	9.33
CONJ	**P**	6.25	0.00	-	-	0.00	0.00	-	-	-	0.00	-	0.00
	R	7.69	0.00	-	-	0.00	0.00	-	-	-	0.00	-	0.00
	$F_{0.5}$	**6.49**	0.00	-	-	0.00	0.00	-	-	-	0.00	-	0.00
CONTR	**P**	29.17	40.00	46.15	-	0.00	-	-	33.33	0.00	66.67	-	28.57
	R	100.00	33.33	85.71	-	0.00	-	-	57.14	0.00	40.00	-	33.33
	$F_{0.5}$	33.98	38.46	50.85	-	0.00	-	-	36.36	0.00	**58.82**	-	29.41
DET	**P**	33.33	36.16	30.92	21.43	0.00	36.03	29.35	26.09	0.00	43.88	-	36.21
	R	14.09	43.03	51.91	0.92	0.00	28.46	7.85	49.41	0.00	12.54	-	23.66
	$F_{0.5}$	26.18	**37.35**	33.64	3.92	0.00	34.21	18.96	28.81	0.00	29.25	-	32.74
MORPH	**P**	55.56	59.15	55.88	28.57	1.16	27.87	20.80	27.78	32.69	100.00	40.00	43.75
	R	48.91	47.73	20.88	5.41	1.39	21.52	30.59	12.50	21.25	2.74	5.00	15.91
	$F_{0.5}$	54.09	**56.45**	41.85	15.38	1.20	26.32	22.22	22.32	29.51	12.35	16.67	32.41
NOUN	**P**	20.90	25.27	0.00	28.57	4.35	0.00	0.00	10.00	10.53	0.00	-	27.78
	R	12.39	19.49	0.00	2.20	2.17	0.00	0.00	1.92	1.92	0.00	-	9.90
	$F_{0.5}$	18.37	**23.86**	0.00	8.40	3.62	0.00	0.00	5.43	5.56	0.00	-	20.41
NOUN:INFL	**P**	60.00	60.00	50.00	-	25.00	100.00	62.50	66.67	66.67	0.00	-	-
	R	85.71	66.67	71.43	-	16.67	33.33	62.50	57.14	66.67	0.00	-	-
	$F_{0.5}$	63.83	61.22	53.19	-	22.73	**71.43**	62.50	64.52	66.67	0.00	-	-
NOUN:NUM	**P**	49.42	44.20	44.06	41.18	14.38	44.05	29.39	31.05	29.00	54.29	-	44.29
	R	56.14	53.74	59.49	3.87	11.28	47.62	42.54	56.20	36.45	10.27	-	16.94
	$F_{0.5}$	**50.63**	45.83	46.47	14.06	13.63	44.72	31.33	34.10	30.23	29.23	-	33.48
NOUN:POSS	**P**	20.00	66.67	-	-	-	-	14.29	0.00	0.00	25.00	-	50.00
	R	14.29	10.53	-	-	-	-	5.26	0.00	0.00	4.55	-	5.00
	$F_{0.5}$	18.52	**32.26**	-	-	-	-	10.64	0.00	0.00	13.16	-	17.86
ORTH	**P**	60.00	66.67	73.81	-	3.45	0.00	28.57	49.32	16.57	-	-	50.00
	R	11.11	40.00	59.62	-	4.55	0.00	6.90	64.29	49.12	-	-	17.24
	$F_{0.5}$	31.91	58.82	**70.45**	-	3.62	0.00	17.54	51.72	19.10	-	-	36.23
OTHER	**P**	20.34	23.60	10.34	0.00	2.33	1.37	14.29	10.00	0.00	0.00	-	11.58
	R	6.92	10.03	0.83	0.00	0.31	0.58	0.58	1.13	0.00	0.00	-	3.15
	$F_{0.5}$	14.65	**18.57**	3.14	0.00	1.01	1.07	2.49	3.90	0.00	0.00	-	7.54
PART	**P**	71.43	33.33	25.00	-	-	16.67	-	-	-	50.00	-	20.00
	R	20.83	15.38	4.76	-	-	21.74	-	-	-	9.52	-	11.11
	$F_{0.5}$	**48.08**	27.03	13.51	-	-	17.48	-	-	-	27.03	-	17.24
PREP	**P**	47.56	41.44	33.33	75.00	0.00	10.71	-	21.74	0.00	36.59	-	20.53
	R	16.05	35.66	13.49	1.44	0.00	12.35	-	2.17	0.00	7.18	-	13.36
	$F_{0.5}$	34.15	**40.14**	25.76	6.70	0.00	11.01	-	7.76	0.00	20.11	-	18.54
PRON	**P**	41.18	20.37	0.00	0.00	11.11	50.00	100.00	27.27	5.00	0.00	-	22.92
	R	9.72	13.41	0.00	0.00	1.69	2.82	1.54	4.62	1.52	0.00	-	13.92
	$F_{0.5}$	**25.00**	18.46	0.00	0.00	5.26	11.49	7.25	13.76	3.42	0.00	-	20.30
PUNCT	**P**	25.00	60.47	37.21	100.00	0.00	44.83	-	27.27	0.00	5.00	-	43.02
	R	3.52	15.48	10.60	1.85	0.00	8.97	-	6.34	0.00	0.96	-	23.13
	$F_{0.5}$	11.26	**38.24**	24.77	8.62	0.00	24.90	-	16.42	0.00	2.72	-	36.71
SPELL	**P**	76.92	77.55	0.00	0.00	25.00	0.00	44.17	68.63	73.98	-	-	100.00
	R	63.83	41.76	0.00	0.00	4.23	0.00	71.29	71.43	85.85	-	-	1.37
	$F_{0.5}$	73.89	66.20	0.00	0.00	12.61	0.00	47.81	69.17	**76.09**	-	-	6.49
VERB	**P**	18.84	15.12	-	0.00	7.69	0.00	14.29	0.00	0.00	0.00	-	16.33
	R	8.23	8.33	-	0.00	0.74	0.00	0.70	0.00	0.00	0.00	-	5.37
	$F_{0.5}$	**14.98**	13.00	-	0.00	2.66	0.00	2.94	0.00	0.00	0.00	-	11.59
VERB:FORM	**P**	34.92	36.36	68.75	0.00	8.77	35.11	30.77	25.00	34.41	28.57	-	31.11
	R	23.40	25.00	24.18	0.00	5.75	35.11	35.56	3.45	32.65	4.65	-	16.09
	$F_{0.5}$	31.79	33.33	**50.23**	0.00	7.94	35.11	31.62	11.11	34.04	14.08	-	26.22
VERB:INFL	**P**	100.00	100.00	-	-	100.00	100.00	50.00	100.00	100.00	-	0.00	-
	R	100.00	100.00	-	-	50.00	50.00	50.00	50.00	100.00	-	0.00	-
	$F_{0.5}$	**100.00**	**100.00**	-	-	83.33	83.33	50.00	83.33	**100.00**	-	0.00	-
VERB:SVA	**P**	49.09	44.05	54.80	50.00	24.56	50.56	56.25	32.69	35.56	59.09	81.58	60.00
	R	27.55	32.74	71.85	1.12	14.58	67.16	18.75	17.35	31.07	13.83	29.25	15.00
	$F_{0.5}$	42.45	41.20	57.53	5.15	21.60	53.19	40.18	27.78	34.56	35.71	**60.08**	37.50
VERB:TENSE	**P**	20.55	26.27	70.00	66.67	3.70	31.25	9.38	20.00	22.78	14.81	100.00	31.25
	R	8.72	17.51	4.12	1.25	0.61	2.98	3.66	2.31	20.57	2.45	0.63	12.05
	$F_{0.5}$	16.16	**23.88**	16.67	5.81	1.84	10.78	7.14	7.91	22.30	7.38	3.05	23.70
WO	**P**	-	38.89	0.00	66.67	-	-	-	0.00	0.00	-	-	41.18
	R	-	33.33	0.00	14.29	-	-	-	0.00	0.00	-	-	35.00
	$F_{0.5}$	-	37.63	0.00	38.46	-	-	-	0.00	0.00	-	-	**39.77**

Table 6: Precision, recall and $F_{0.5}$ for each team and error type. A dash indicates the team's system did not attempt to correct the given error type (TP+FP = 0). The highest F-score for each type is highlighted.

	CAMB		
Type	**P**	**R**	**$F_{0.5}$**
M:DET	43.20	51.77	44.68
R:DET	19.33	35.37	21.26
U:DET	43.75	39.90	42.92
DET	36.16	43.03	37.35

	CUUI		
Type	**P**	**R**	**$F_{0.5}$**
M:DET	23.86	45.00	26.34
R:DET	27.03	24.39	26.46
U:DET	36.19	66.37	39.81
DET	30.92	51.91	33.64

Table 7: Detailed breakdown of Determiner errors for two teams.

- Despite the prevalence of spell checkers nowadays, many teams did not seem to employ them; this would have been an easy way to boost overall performance.
- Although several teams built specialised classifiers for DET and PREP errors, CAMB's hybrid MT approach still outperformed them. This might be because the classifiers were trained using a different error type framework however.
- CUUI's classifiers significantly outperformed all other approaches at ORTH and VERB:FORM errors. This suggests classifiers are well-suited to these error types.
- Although UFC's rule-based approach was the best at VERB:SVA errors, CUUI's classifier was not very far behind.
- Only AMU managed to correct any CONJ errors.
- Content word errors (i.e. ADJ, ADV, NOUN and VERB) were unsurprisingly very difficult for all teams.

5.3 Detailed Error Types

In addition to analysing general error types, the modular design of our framework also allows us to evaluate error type performance at an even greater level of detail. For example, Table 7 shows the breakdown of Determiner errors for two teams using different approaches in terms of edit operation. Note that this is a representative example of detailed error type performance, as an analysis of all error type combinations for all teams would take up too much space.

Team	**P**	**R**	**$F_{0.5}$**
AMU	16.90	5.33	11.79
CAMB	27.22	17.06	24.32
CUUI	15.69	3.67	9.48
IITB	28.57	0.94	4.15
IPN	3.33	0.47	1.51
NTHU	0.00	0.00	0.00
PKU	25.00	1.40	5.73
POST	12.77	2.82	7.48
RAC	2.96	2.82	2.93
SJTU	10.00	0.47	1.99
UFC	-	-	-
UMC	19.82	9.82	16.47

Table 8: Each team's performance at correcting multi-token edits; i.e. there are at least two tokens on one side of the edit.

While CAMB's hybrid MT approach achieved a higher score than CUUI's classifier overall, our more detailed evaluation reveals that CUUI actually outperformed CAMB at Replacement Determiner errors. We also learn that CAMB scored twice as highly on M:DET and U:DET than it did on R:DET and that CUUI's significantly higher U:DET recall was offset by a lower precision. Ultimately, this shows that even though one approach might be better than another overall, different approaches may still have complementary strengths.

5.4 Multi Token Errors

Another benefit of explicitly annotating all hypothesis edits is that edit spans become fixed; this means we can evaluate system performance in terms of edit size. Table 8 hence shows the overall performance for each team at correcting multi-token edits, where a multi-token edit is an edit that has at least two tokens on either side. In the CoNLL-2014 test set, there are roughly 220 such edits (about 10% of all edits).

In general, teams did not do well at multi-token edits. In fact only three teams achieved scores greater than 10% $F_{0.5}$ and all of them used MT (AMU, CAMB, UMC). This is significant because recent work has suggested that the main goal of GEC should be to produce fluent-sounding, rather than just grammatical sentences, even though this often requires complex multi-token edits (Sakaguchi et al., 2016). If no system is particularly adept at correcting multi-token errors however, robust fluency correction will likely require more sophisticated methods than are currently available.

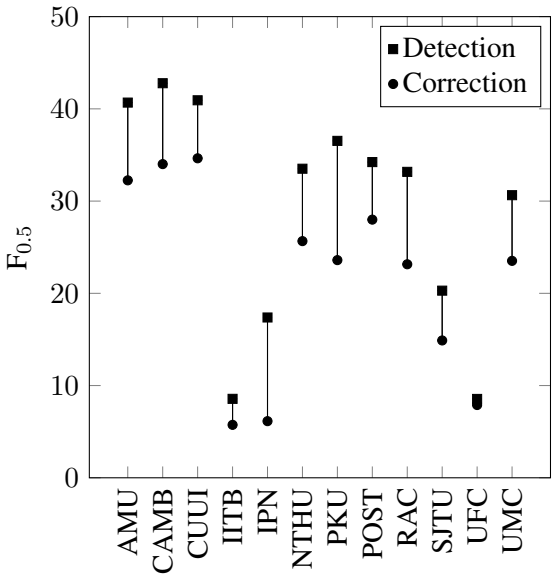


Figure 1: The difference between detection and correction scores for each team overall.

5.5 Detection vs. Correction

Another important aspect of GEC that is seldom reported in the literature is that of error detection; i.e. the extent to which a system can identify erroneous tokens in text. This can be calculated by comparing the edit overlap between the hypothesis and reference files *regardless of the proposed correction* in a manner similar to Recognition evaluation in the HOO shared tasks for GEC (Dale and Kilgarriff, 2011).

Figure 1 hence shows how each team's score for detection differed in relation to their score for correction. While CAMB scored highest for detection overall, it is interesting to note that CUUI ultimately performed slightly better than CAMB at correction. This suggests CUUI was more successful at correcting the errors they detected than CAMB. In contrast, IPN and PKU are notable for detecting significantly more errors than they were able to correct. Nevertheless, a system's ability to detect errors, even if it is unable to correct them, is still likely to be valuable information to a learner (Rei and Yannakoudakis, 2016).

Finally, although we do not do so here, our scorer is also capable of providing a detailed error type breakdown for detection.

6 Conclusion

In this paper, we described ERRANT, a grammatical ERRor ANnotation Toolkit designed to automatically annotate parallel error correction data with explicit edit spans and error type information. ERRANT can be used to not only facilitate a detailed error type evaluation in GEC, but also to standardise existing error correction corpora and reduce annotator workload. We release ERRANT with this paper.

Our approach makes use of previous work to align sentences based on linguistic intuition and then introduces a new rule-based framework to classify edits. This framework is entirely dataset independent, and relies only on automatically obtained information such as POS tags and lemmas. A small-scale evaluation of our classifier found that each rater considered >95% of the predicted error types as either "Good" (85%) or "Acceptable" (10%).

We demonstrated the value of ERRANT by carrying out a detailed evaluation of system error type performance for all teams in the CoNLL-2014 shared task on Grammatical Error Correction. We found that different systems had different strengths and weaknesses which we hope researchers can exploit to further improve general performance.

References

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 697–707. http://www.aclweb.org/anthology/P15-1068.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Montréal, Canada, pages 568–572. http://www.aclweb.org/anthology/N12-1067.

Robert Dale and Adam Kilgarriff. 2011. Helping Our Own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, Stroudsburg, PA, USA, ENLG '11, pages 242–249. http://dl.acm.org/citation.cfm?id=2187681.2187725.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York.

Mariano Felice and Ted Briscoe. 2015. Towards a standard evaluation method for grammatical error detection and correction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 578–587. http://www.aclweb.org/anthology/N15-1060.

Mariano Felice, Christopher Bryant, and Ted Briscoe. 2016. Automatic extraction of learner errors in ESL sentences using linguistically enhanced alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 825–835. http://aclweb.org/anthology/C16-1079.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Baltimore, Maryland, pages 15–24. http://www.aclweb.org/anthology/W14-1702.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Edward Gillian. 2015. Human evaluation of grammatical error correction systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 461–470. http://aclweb.org/anthology/D15-1052.

Anubhav Gupta. 2014. Grammatical error detection using tagger disagreement. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Baltimore, Maryland, pages 49–52. http://www.aclweb.org/anthology/W14-1706.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU system in the CoNLL-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Baltimore, Maryland, pages 25–33. http://www.aclweb.org/anthology/W14-1703.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 588–593. http://www.aclweb.org/anthology/P15-2097.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. ACL, Baltimore, Maryland, USA, pages 1–14. http://aclweb.org/anthology/W/W14/W14-1701.pdf.

Justus J. Randolph. 2005. Free-marginal multirater kappa: An alternative to Fleiss' fixed-marginal multirater kappa. *Joensuu University Learning and Instruction Symposium* http://files.eric.ed.gov/fulltext/ED490661.pdf.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1181–1191. http://www.aclweb.org/anthology/P16-1112.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014. The Illinois-Columbia system in the CoNLL-2014 shared task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Baltimore, Maryland, pages 34–42. http://www.aclweb.org/anthology/W14-1704.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 2205–2215. http://aclweb.org/anthology/P16-1208.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics* 4:169–182. https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/800.

Ben Swanson and Elif Yamangil. 2012. Correction detection and error type selection as an ESL educational aid. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Montréal, Canada, pages 357–361. http://www.aclweb.org/anthology/N12-1037.

Huichao Xue and Rebecca Hwa. 2014. Improved correction detection in revised ESL sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 599–604. http://www.aclweb.org/anthology/P14-2098.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 180–189. http://www.aclweb.org/anthology/P11-1019.

A Complete list of valid error code combinations

			Operation Tier		
		Type	**Missing**	**Unnecessary**	**Replacement**
Token Tier	**Part Of Speech**	Adjective	M:ADJ	U:ADJ	R:ADJ
		Adverb	M:ADV	U:ADV	R:ADV
		Conjunction	M:CONJ	U:CONJ	R:CONJ
		Determiner	M:DET	U:DET	R:DET
		Noun	M:NOUN	U:NOUN	R:NOUN
		Particle	M:PART	U:PART	R:PART
		Preposition	M:PREP	U:PREP	R:PREP
		Pronoun	M:PRON	U:PRON	R:PRON
		Punctuation	M:PUNCT	U:PUNCT	R:PUNCT
		Verb	M:VERB	U:VERB	R:VERB
	Other	Contraction	M:CONTR	U:CONTR	R:CONTR
		Morphology	-	-	R:MORPH
		Orthography	-	-	R:ORTH
		Other	M:OTHER	U:OTHER	R:OTHER
		Spelling	-	-	R:SPELL
		Word Order	-	-	R:WO
Morphology Tier		Adjective Form	-	-	R:ADJ:FORM
		Noun Inflection	-	-	R:NOUN:INFL
		Noun Number	-	-	R:NOUN:NUM
		Noun Possessive	M:NOUN:POSS	U:NOUN:POSS	R:NOUN:POSS
		Verb Form	M:VERB:FORM	U:VERB:FORM	R:VERB:FORM
		Verb Inflection	-	-	R:VERB:INFL
		Verb Agreement	-	-	R:VERB:SVA
		Verb Tense	M:VERB:TENSE	U:VERB:TENSE	R:VERB:TENSE

Table 9: There are 55 total possible error types. This table shows all of them except UNK, which indicates an uncorrected error. A dash indicates an impossible combination.

B TP, FP and FN counts for various CoNLL-2014 results

	AMU			**CAMB**			**CUUI**			**IITB**		
Type	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**
Missing	58	74	347	131	154	310	77	215	347	2	11	336
Replacement	428	722	1162	477	794	1219	381	449	1277	20	47	1320
Unnecessary	0	0	412	125	365	330	158	304	316	6	7	385

	IPN			**NTHU**			**PKU**			**POST**		
Type	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**
Missing	1	34	339	46	88	358	16	32	350	52	115	344
Replacement	53	484	1319	299	784	1262	279	663	1243	312	629	1302
Unnecessary	0	2	389	65	122	342	0	1	397	155	434	317

	RAC			**SJTU**			**UFC**			**UMC**		
Type	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**	**TP**	**FP**	**FN**
Missing	1	65	368	15	9	323	0	0	339	99	148	321
Replacement	325	780	1236	47	46	1325	36	14	1326	143	269	1331
Unnecessary	0	5	407	45	210	351	0	0	381	74	365	357

Table 10: True Positive, False Positive and False Negative counts for each team in terms of Missing, Replacement and Unnecessary edits. The total number of edits may vary for each system, as this depends on the individual references that are chosen during evaluation. These results were used to make Table 5.

		AMU	CAMB	CUUI	IITB	IPN	NTHU	PKU	POST	RAC	SJTU	UFC	UMC
ADJ	**TP**	2	5	0	0	0	0	2	0	1	0	0	0
	FP	39	50	0	3	2	3	1	4	7	8	0	21
	FN	28	31	30	23	23	25	26	33	27	20	20	26
ADJ:FORM	**TP**	5	6	3	2	0	3	2	1	2	0	0	3
	FP	4	2	0	0	1	6	0	1	23	0	0	0
	FN	3	4	6	3	5	5	5	6	3	5	5	2
ADV	**TP**	1	9	0	0	0	0	0	0	0	1	0	5
	FP	14	69	1	1	1	2	1	0	4	20	0	52
	FN	33	35	36	32	33	35	37	41	37	32	33	35
CONJ	**TP**	1	0	0	0	0	0	0	0	0	0	0	0
	FP	15	18	0	0	1	1	0	0	0	6	0	26
	FN	12	15	14	12	12	13	12	15	13	13	12	14
CONTR	**TP**	7	2	6	0	0	0	0	4	0	2	0	2
	FP	17	3	7	0	1	0	0	8	1	1	0	5
	FN	0	4	1	5	5	5	5	3	5	3	5	4
DET	**TP**	52	179	231	3	0	107	27	210	0	43	0	88
	FP	104	316	516	11	13	190	65	595	9	55	0	155
	FN	317	237	214	324	325	269	317	215	346	300	327	284
MORPH	**TP**	45	42	19	4	1	17	26	10	17	2	4	14
	FP	36	29	15	10	85	44	99	26	35	0	6	18
	FN	47	46	72	70	71	62	59	70	63	71	76	74
NOUN	**TP**	14	23	0	2	2	0	0	2	2	0	0	10
	FP	53	68	5	5	44	9	29	18	17	16	0	26
	FN	99	95	109	89	90	102	103	102	102	93	92	91
NOUN:INFL	**TP**	6	6	5	0	1	2	5	4	4	0	0	0
	FP	4	4	5	0	3	0	3	2	2	1	0	0
	FN	1	3	2	6	5	4	3	3	2	6	6	6
NOUN:NUM	**TP**	128	122	141	7	22	100	97	136	78	19	0	31
	FP	131	154	179	10	131	127	233	302	191	16	0	39
	FN	100	105	96	174	173	110	131	106	136	166	178	152
NOUN:POSS	**TP**	3	2	0	0	0	0	1	0	0	1	0	1
	FP	12	1	0	0	0	0	6	1	38	3	0	1
	FN	18	17	20	18	19	22	18	21	20	21	20	19
ORTH	**TP**	3	14	31	0	1	0	2	36	28	0	0	5
	FP	2	7	11	0	28	1	5	37	141	0	0	5
	FN	24	21	21	21	21	27	27	20	29	24	21	24
OTHER	**TP**	24	38	3	0	1	2	2	4	0	0	0	11
	FP	94	123	26	8	42	144	12	36	52	11	0	84
	FN	323	341	358	329	322	345	343	349	346	323	327	338
PART	**TP**	5	4	1	0	0	5	0	0	0	2	0	2
	FP	2	8	3	0	0	25	0	0	0	2	0	8
	FN	19	22	20	19	21	18	21	20	19	19	17	16
PREP	**TP**	39	92	34	3	0	30	0	5	0	15	0	31
	FP	43	130	68	1	2	250	0	18	3	26	0	120
	FN	204	166	218	205	207	213	219	225	215	194	206	201
PRON	**TP**	7	11	0	0	1	2	1	3	1	0	0	11
	FP	10	43	1	5	8	2	0	8	19	22	0	37
	FN	65	71	63	57	58	69	64	62	65	62	62	68
PUNCT	**TP**	5	26	16	2	0	13	0	9	0	1	0	37
	FP	15	17	27	0	16	16	0	24	29	19	0	49
	FN	137	142	135	106	114	132	123	133	129	103	109	123
SPELL	**TP**	60	38	0	0	3	0	72	70	91	0	0	1
	FP	18	11	1	1	9	2	91	32	32	0	0	0
	FN	34	53	74	68	68	74	29	28	15	70	70	72
VERB	**TP**	13	13	0	0	1	0	1	0	0	0	0	8
	FP	56	73	0	6	12	12	6	4	5	17	0	41
	FN	145	143	165	133	135	152	141	164	151	139	131	141
VERB:FORM	**TP**	22	24	22	0	5	33	32	3	32	4	0	14
	FP	41	42	10	1	52	61	72	9	61	10	0	31
	FN	72	72	69	87	82	61	58	84	66	82	82	73
VERB:INFL	**TP**	2	2	0	0	1	1	1	1	2	0	0	0
	FP	0	0	0	0	0	0	1	0	0	0	1	0
	FN	0	0	2	2	1	1	1	1	0	2	2	2
VERB:SVA	**TP**	27	37	97	1	14	90	18	17	32	13	31	15
	FP	28	47	80	1	43	88	14	35	58	9	7	10
	FN	71	76	38	88	82	44	78	81	71	81	75	85
VERB:TENSE	**TP**	15	31	7	2	1	5	6	4	36	4	1	20
	FP	58	87	3	1	26	11	58	16	122	23	0	44
	FN	157	146	163	158	163	163	158	169	139	159	159	146
WO	**TP**	0	7	0	2	0	0	0	0	0	0	0	7
	FP	0	11	10	1	0	0	0	2	1	0	0	10
	FN	12	14	14	12	12	11	12	12	12	11	11	13

Table 11: True Positive, False Positive and False Negative counts for each error type for each team. These results were used to make Table 6.

Evaluation Metrics for Machine Reading Comprehension: Prerequisite Skills and Readability

Saku Sugawara[♠], **Yusuke Kido**[♠], **Hikaru Yokono**[♣], and **Akiko Aizawa**[♢♠]
[♠]The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
[♣]Fujitsu Laboratories Ltd., 4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki, Japan
[♢]Natural Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
sakus@is.s.u-tokyo.ac.jp mail@yusuk.eki.do
yokono.hikaru@jp.fujitsu.com aizawa@nii.ac.jp

Abstract

Knowing the quality of reading comprehension (RC) datasets is important for the development of natural-language understanding systems. In this study, two classes of metrics were adopted for evaluating RC datasets: prerequisite skills and readability. We applied these classes to six existing datasets, including MCTest and SQuAD, and highlighted the characteristics of the datasets according to each metric and the correlation between the two classes. Our dataset analysis suggests that the readability of RC datasets does not directly affect the question difficulty and that it is possible to create an RC dataset that is easy to read but difficult to answer.

1 Introduction

A major goal of natural language processing (NLP) is to develop agents that can understand natural language. Such an ability can be tested with a reading comprehension (RC) task that requires the agent to read open-domain documents and answer questions about them. Constructing systems with RC competence is challenging because RC comprises multiple processes including parsing, understanding cohesion, and inference with linguistic and general knowledge.

Clarifying what a system achieves is important in the development of RC systems. To achieve robust improvement, systems should be measured according to a variety of metrics beyond simple accuracy. However, a current problem is that most RC datasets are presented only with superficial categories, such as question types (e.g., what, where, and who) and answer types (e.g., numeric, location, and person). In addition, Chen et al. (2016) noted that some questions in datasets may not be suited to the testing of RC systems. In such situations, it is difficult to obtain an accurate assessment of the RC system.

ID: SQuAD, United_Methodist_Church
Context: The United Methodist Church (UMC) practices infant and adult baptism. Baptized Members are those who have been baptized as an infant or child, but who have not subsequently professed their own faith.
Question: What are members who have been baptized as an infant or child but who have not subsequently professed their own faith?
Answer: Baptized Members

ID: MCTest, mc160.dev.8
Context: Sara wanted to play on a baseball team. She had never tried to swing a bat and hit a baseball before. Her Dad gave her a bat and together they went to the park to practice.
Question: Why was Sara practicing?
Answer: She wanted to play on a team

Figure 1: Examples of RC questions from SQuAD (Rajpurkar et al., 2016) and MCTest (Richardson et al., 2013) (the Contexts are excerpts).

Norvig (1989) argued that questions that are easy for humans to answer often turn out to be difficult for machines. For example, consider the two RC questions in Figure 1. The first example is from SQuAD (Rajpurkar et al., 2016), although the document is taken from a Wikipedia article and was therefore written for adults. The question is answerable simply by noticing one sentence, without needing to fully understand the content of the text. On the other hand, consider the second example from MCTest (Richardson et al., 2013), which was written for children and is easy to read. Here, answering the question involves gathering information from multiple sentences and utilizing a combination of several skills, such as understanding causal relations (*Sara wanted...* $\rightarrow$ *they went to...*), coreference resolution (*Sara* and *Her Dad* = *they*), and complementing ellipsis (*baseball team* = *team*). These two examples show that the readability of the text does not necessarily correlate with the difficulty of answering questions about it.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 806–817
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1075

Furthermore, the accompanying categories of existing RC datasets cannot help with the analysis of this issue.

In this study, our goal is to investigate how these two types of difficulty, namely "answering questions" and "reading text," are correlated in RC. Corresponding to each type, we formalize two classes of evaluation metrics, *prerequisite skills* and *readability*, and analyze existing RC datasets. Our intention is to provide the basis of an evaluation methodology of RC systems to help their robust development.

Our two classes of metrics are inspired by the analysis in McNamara and Magliano (2009) of human text comprehension in psychology. They considered two aspects of text comprehension, namely "strategic/skilled comprehension" and "text ease of processing."

Our first class defines metrics for "strategic/skilled comprehension," namely the difficulty of comprehending the context when answering questions. We adopted the set of prerequisite skills that Sugawara et al. (2017) proposed for the fine-grained analysis of RC capability. Their study also presented an important observation of the relation between the difficulty of an RC task and prerequisite skills: the more skills that are required to answer a question, the more difficult is the question. Based on this observation, in this work, we assume that the number of skills required to answer a question is a reasonable indication of the difficulty of the question. This is because each skill corresponds to one of the functions of an NLP system, which has to be capable of that functionality.

Our second class defines metrics for "text ease of processing," namely the difficulty of reading the text. We regard it as readability of the text in terms of syntactic and lexical complexity. From among readability studies in NLP, we adopt a wide range of linguistic features proposed by Vajjala and Meurers (2012), which can be used for texts with no available annotations.

The contributions of this paper are as follows.

1. We adopt two classes of evaluation metrics to show the qualitative features of RC datasets. Through analyses of RC datasets, we demonstrate that there is only a weak correlation between the difficulty of questions and the readability of context texts in RC datasets.

2. We revise a previous classification of prerequisite skills for RC. Specifically, skills of knowledge reasoning are organized by using insights of entailment phenomena in NLP and human text comprehension in psychology.

3. We annotate six existing RC datasets, compared to the two datasets considered in Sugawara and Aizawa (2016), with our organized metrics being used in the comparison. We have made the results publicly available[1] and report on the characteristics of the datasets and the differences between them.

We should note that, in this study, RC datasets with different task formulations were annotated with prerequisite skills under the same conditions. Annotators first saw a context, a question, and its answer. They selected the sentences required to provide the answer, and then annotated them with appropriate prerequisite skills. That is, the datasets were annotated from the point of view of whether the context entailed the hypothesis constructed from the pair of the question and answer. This means that our methodology cannot quantify the systems' competence in searching the context for necessary sentences and answer candidates. In other words, our methodology can be only used to evaluate the competence of understanding RC questions as *contextual entailments*.

The remainder of this paper is divided into the following sections. First, we discuss related work in Section 2. Next, we specify our two classes of metrics in Section 3. In Section 4, we annotate existing RC datasets with the prerequisite skills. Section 5 gives the results of our dataset analysis and Section 6 discusses their implications. Section 7 presents our conclusions.

2 Related Work

2.1 Reading Comprehension Datasets

In this section, we present a short history of RC datasets. To our knowledge, Hirschman et al. (1999) were the first to use NLP methods for RC. Their dataset comprised reading materials for grades 3–6 with simple 5W (*wh*-) questions. Subsequent investigations into questions of natural language understanding focused on other formulations, such as question answering (Yang et al., 2015; Wang et al., 2007; Voorhees et al., 1999) and

[1] http://www-al.nii.ac.jp/rc_dataset_analysis

textual entailment (Bentivogli et al., 2010; Sammons et al., 2010; Dagan et al., 2006). One of the RC tasks of the time was QA4MRE (Sutcliffe et al., 2013). The highest accuracy achieved for this task was 59% and the size of the dataset was very limited: there were only 224 gold-standard questions, which is insufficient for machine learning methods.

This means that an important issue for designing RC datasets is their scalability. Richardson et al. (2013) presented MCTest, which is an open-domain narrative dataset for gauging comprehension at a child's level. This dataset was created by crowdsourcing and was based on a scalable methodology. Since then, additional large-scale datasets have been proposed with the development of machine learning methods in NLP. For example, the CNN/Daily Mail dataset (Hermann et al., 2015) and CBTest (Hill et al., 2016) have approximately 1.4M and 688K passages, respectively. These context texts and questions were automatically curated and generated from large corpora. However, Chen et al. (2016) indicated that approximately 25% of the questions in the CNN/Daily Mail dataset are either unsolvable or nonsensical. This dataset-quality issue highlights the demand for more stable and robust sourcing methods.

Several additional RC datasets were presented in the last half of 2016, involving large documents and sensible queries that were guaranteed by crowdsourcing or other human testing. They were intended to provide large and high-quality content for machine learning models. Nonetheless, as shown in the examples of Figure 1, they were not offered with metrics that could evaluate NLP systems adequately with respect to the difficulty of questions and the surface features of texts.

2.2 Reading Comprehension in Psychology

In psychology, there is a rich tradition of research on human text comprehension. The construction–integration (C–I) model (Kintsch, 1988) is one of the most basic and influential theories. This model assumes a connectional and computational architecture for text comprehension. It assumes that comprehension is the processing of information based on the following two steps.[2]

1. *Construction:* read sentences or clauses as inputs; form and elaborate concepts and propositions corresponding to the inputs.
2. *Integration:* associate the contents to understand them consistently (e.g., coreference, discourse, and coherence).

During these steps, three levels of representation are constructed (van Dijk and Kintsch, 1983): the *surface code* (i.e., wording and syntax), the *textbase* (i.e., text propositions with cohesion), and the *situation model* (i.e., mental representation). Based on these assumptions, McNamara and Magliano (2009) proposed two aspects of text comprehension, namely "strategic/skilled comprehension" and "text ease of processing." We adopted these assumptions as the basis of our two classes of evaluation metrics (Section 3).

In an alternative approach, Kintsch (1993) proposed two dichotomies for the classification of human inferences, including the knowledge-based inference assumed in the C–I model. The first dichotomy is between inferences that are *automatic* and those that are *controlled*. However, Graesser et al. (1994) indicated that this distinction is ambiguous, because there is a continuum between the two states that depends on individuals. Therefore, this dichotomy is unsuited to empirical evaluation, which is our focus. The second dichotomy is between inferences that are *retrieved* and those that are *generated*. *Retrieved* means that the information used for inference is retrieved entirely from the context. In contrast, when inferences are *generated*, the reader uses external knowledge that goes beyond the context.

A similar distinction was proposed by McNamara and Magliano (2009), namely that between *bridging* and *elaboration*. A bridging inference connects current information to other information that has been encountered previously. Elaboration connects current information to external knowledge that is not included in the context. We use these two types of inference in the classification of knowledge reasoning.

3 Evaluation Metrics for Datasets

Following the depiction of text comprehension by McNamara and Magliano (2009), we adopted two classes for the evaluation of RC datasets: *prerequisite skills* and *readability*.

For the prerequisite skills class (Section 3.1), we refined RC skills that were proposed by Sugawara et al. (2017) and Sugawara and Aizawa (2016). However, a problem in these studies is that their categorization of knowledge reasoning

[2]Note that this is a very simplified overview.

was provisional and with a weak theoretical background.

Therefore, in this study, we reorganized the category of knowledge reasoning in terms of textual entailment in NLP and human text comprehension in psychology. In research on textual entailment, several methodologies have been proposed for the precise analysis of entailment phenomena (Dagan et al., 2013; LoBue and Yates, 2011). In psychology research, as described in Section 2.2, McNamara and Magliano (2009) proposed a similar distinction for inferences: *bridging* versus *elaboration*. We utilized these insights in developing a comprehensive but not overly specific classification of knowledge reasoning.

Our prerequisite skills class includes the *textbase* and *situation model* (van Dijk and Kintsch, 1983). In our terminology, this means understanding each fact and associating multiple facts in a text, such as the relations of events, characters, or the topic of a story. The skills also involve knowledge reasoning, which is divided into several metrics according to the distinctions of human inferences. This point is discussed by Kintsch (1993) and McNamara and Magliano (2009). It also accords with the classification of entailment phenomena by Dagan et al. (2013) and LoBue and Yates (2011).

Readability metrics (Section 3.2) are quantitative measures used to assess the difficulty of reading, with respect to vocabulary and the complexity of texts. In this study, they measure the competence in understanding the first basic representation of a text, called the *surface code* (van Dijk and Kintsch, 1983).

3.1 Prerequisite Skills

Based on the 10 RC skills in Sugawara et al. (2017), we identified 13 prerequisite skills, which are presented below. (We use * and † to indicate skills that have been modified/elaborated from the original definition or have been newly introduced in this study, respectively.)

1. Object tracking*: jointly tracking or grasping of multiple objects, including sets or memberships (Clark, 1975). This skill is a version of the *list/enumeration* used in the original classification, renamed to emphasize its scope with respect to multiple objects.

2. Mathematical reasoning*: we merged statistical and quantitative reasoning with mathematical reasoning. This skill is a renamed version of *mathematical operations*.

3. Coreference resolution*: this skill has a small modification to include an anaphora (Dagan et al., 2013). It is similar to *direct reference* (Clark, 1975).

4. Logical reasoning*: we identified this skill as the understanding of predicate logic, e.g., conditionals, quantifiers, negation, and transitivity. Note that this skill, together with *mathematical reasoning*, is intended to align with the offline skills described by Graesser et al. (1994).

5. Analogy*: understanding of metaphors including metonymy and synecdoche (see LoBue and Yates (2011) for examples of synecdoche.)

6. Causal relation: understanding of causality that is represented by explicit expressions such as "why," "because," and "the reason for" (only if they exist).

7. Spatiotemporal relation: understanding of spatial and/or temporal relationships between multiple entities, events, and states.

In addition, we propose the following four categories by refining the "commonsense reasoning" category proposed originally in Sugawara et al. (2017).

8. Ellipsis†: recognizing implicit/omitted information (argument, predicate, quantifier, time, or place). This skill is inspired by Dagan et al. (2013) and the discussion in Sugawara et al. (2017).

9. Bridging†: inference supported by grammatical and lexical knowledge (e.g., synonymy, hypernymy, thematic role, part of events, idioms, and apposition). This skill is inspired by the concept of *indirect reference* in the literature (Clark, 1975). Note that we exclude *direct reference* because it is covered by *coreference resolution* (pronominalization) and *elaboration* (epithets).

10. Elaboration†: inference using known facts, general knowledge (e.g., kinship, exchange, typical event sequence, and naming), and implicit relations (e.g., noun compounds and possessives) (see Dagan et al. (2013) for details). *Bridging* and *elaboration* are distinguished by the knowledge used in inferences being grammatical/lexical or general/commonsense, respectively.

11. Meta-knowledge†: using knowledge that includes a reader, writer, or text genre (e.g., narratives and expository documents) from meta-viewpoints (e.g., *Who are the principal characters of the story?* or *What is the main subject of*

this article?). Although this skill can be regarded as part of *elaboration*, we defined it as an independent skill because this knowledge is specific to RC. We were motivated by the discussion in Smith et al. (2015).

Whereas the above 11 skills involve multiple items, the final pair of skills involve only a single sentence.

12. Schematic clause relation: understanding of complex sentences that have coordination or subordination, including relative clauses.

13. Punctuation*: understanding of punctuation marks (e.g., parenthesis, dash, quotation, colon, or semicolon). This skill is a renamed version of *special sentence structure*. Concerning the original definition, we regarded "scheme" in figures of speech as ambiguous and excluded it. We defined *ellipsis* as a independent skill, and apposition was merged into *bridging*. Similarly, understanding of constructions was merged into the idioms in *bridging*.

Note that we did not construct this classification to be dependent on particular RC systems in NLP. This was because our methodology is intended to be general and applicable to many kinds of architectures. For example, we did not consider the dichotomy between *automatic* and *controlled* inferences because the usage of knowledge is not necessarily the same for all RC systems.

3.2 Readability Metrics

In this study, we evaluated the readability of texts based on metrics in NLP. Several studies have examined readability in various applications, such as second-language learning (Razon and Barnden, 2015) and text simplification (Aluisio et al., 2010), and from various aspects, such as development measures in second-language acquisition (Vajjala and Meurers, 2012) and discourse relations (Pitler and Nenkova, 2008).

Of these, we adopted the classification of linguistic features proposed by Vajjala and Meurers (2012). This was because they presented a comparison of a wide range of linguistic features focusing on second-language acquisition and their method can be applied to plain text.[3]

We list the readability metrics in Table 1, which were reported by Vajjala and Meurers (2012) as the top 10 features that affect human readability. To classify these metrics, we can identify three classes: lexical features (*NumChar*, *NumSyll*, *AWL*, *AdvVar*, and *ModVar*), syntactic features (*MLS*, *CoOrd*, *DC/C*, and *CN/C*), and traditional features (*Coleman*). We applied these metrics only to sentences that needed to be read in answering questions.

- Ave. no. of characters per word (*NumChar*)
- Ave. no. of syllables per word (*NumSyll*)
- Ave. sentence length in words (*MLS*)
- Proportion of words in AWL (*AWL*)
- Modifier variation (*ModVar*)
- No. of coordinate phrases per sentence (*CoOrd*)
- Coleman–Liau index (*Coleman*)
- Dependent clause-to-clause ratio (*DC/C*)
- Complex nominals per clause (*CN/C*)
- Adverb variation (*AdvVar*)

Table 1: Readability metrics. *AWL* refers to the Academic Word List.[4]

However, because these metrics were proposed for human readability, they do not necessarily correlate with those used in RC systems. Therefore, in any system analysis, ideally we would have to consult a variety of features.

4 Annotation of Reading Comprehension Datasets

We annotated six existing RC datasets with the prerequisite skills. We explain the annotation procedure in Section 4.1 and the annotated RC datasets in Section 4.2.

4.1 Annotation Procedure

We prepared annotation guidelines according to Sugawara et al. (2017). The guidelines include the definitions and examples of the skills and annotation instructions.

Four annotators were asked to simulate the process of answering questions in RC datasets, using only the prerequisite skills, and to annotate questions with one or more skills required in answering. For each task in the datasets, the annotators saw simultaneously the context, question, and its answer. When a dataset contained multiple-choice questions, we showed all candidate answers and labeled the correct one with an asterisk. The an-

[3]The classification in Pitler and Nenkova (2008) is more suited to measuring text quality. However, we could not use their results because we could not use discourse annotations.

[4]http://en.wikipedia.org/wiki/Academic_Word_List

RC dataset	Genre	Query sourcing	Task formulation
QA4MRE (2013)	Technical documents	Handcrafted by experts	Multiple choice
MCTest (2013)	Narratives by crowd workers	Crowdsourced	Multiple choice
SQuAD (2016)	Wikipedia articles	Crowdsourced	Text span selection
Who-did-What (2016)	News articles	Automated	Cloze
MS MARCO (2016)	Segmented web pages	Search engine queries	Description
NewsQA (2016)	News articles	Crowdsourced	Text span selection

Table 2: Analyzed RC datasets, their genres, query sourcing methods, and task formulations.

notators then selected the sentences that needed to be read to be able to answer the question and decided on the set of prerequisite skills required.

The annotators were allowed to select *nonsense* for unsolvable or unanswerable questions (e.g., the "coreference error" and "ambiguous" questions described in Chen et al. (2016)) to distinguish them from any solvable questions that required no skills.

4.2 Datasets

As summarized in Table 2, the annotation was performed on six existing RC datasets: QA4MRE (Sutcliffe et al., 2013), MCTest (Richardson et al., 2013), SQuAD (Rajpurkar et al., 2016), Who-did-What (Onishi et al., 2016), MS MARCO (Nguyen et al., 2016), and NewsQA (Trischler et al., 2016). We selected these datasets to enable coverage of a variety of genres, query sourcing methods, and task formulations. From each dataset, we randomly selected 100 questions. This number was considered sufficient for the degree of analysis of RC datasets performed by Chen et al. (2016). The questions were sampled from the gold-standard dataset of QA4MRE and the development sets of the other RC datasets. (We explain the method of choosing questions for the annotation in Appendix A.)

For a variety of reasons, there were other datasets we did not annotate in this study. CNN/Daily Mail (Hermann et al., 2015) is anonymized and contains errors, according to Chen et al. (2016), making it unsuitable for annotation. We considered CBTest (Hill et al., 2016) to be devised as language-modeling tasks rather than RC-related tasks. LAMBADA (Paperno et al., 2016) texts are formatted for machine reading, with all tokens in lower case, which would seem to disallow inferences based on proper nouns and render them unsuitable for human reading and annotation.

Skills	QA4MRE	MCTest	SQuAD	WDW	MARCO	NewsQA
1. Tracking	**11.0**	6.0	3.0	8.0	6.0	2.0
2. Math.	**4.0**	**4.0**	0.0	3.0	0.0	1.0
3. Coref. resol.	32.0	**49.0**	13.0	19.0	15.0	24.0
4. Logical rsng.	**15.0**	2.0	0.0	8.0	1.0	2.0
5. Analogy	**7.0**	0.0	0.0	**7.0**	0.0	3.0
6. Causal rel.	1.0	**6.0**	0.0	2.0	0.0	4.0
7. Sptemp rel.	**26.0**	9.0	2.0	2.0	0.0	3.0
8. Ellipsis	13.0	4.0	3.0	**16.0**	2.0	15.0
9. Bridging	**69.0**	26.0	42.0	59.0	36.0	50.0
10. Elaboration	**60.0**	8.0	13.0	57.0	18.0	36.0
11. Meta	**1.0**	**1.0**	0.0	0.0	0.0	0.0
12. Clause rel.	**52.0**	40.0	28.0	42.0	27.0	34.0
13. Punctuation	**34.0**	1.0	24.0	20.0	14.0	25.0
Nonsense	10.0	**1.0**	3.0	27.0	14.0	**1.0**

Table 3: Frequencies (%) of prerequisite skills needed for the RC datasets.

#Skills	QA4MRE	MCTest	SQuAD	WDW	MARCO	NewsQA
0	2.0	18.0	27.0	2.0	15.0	13.0
1	13.0	36.0	33.0	5.0	35.0	26.0
2	13.0	24.0	24.0	14.0	29.0	23.0
3	20.0	15.0	6.0	22.0	6.0	25.0
4	14.0	4.0	6.0	16.0	2.0	9.0
5	13.0	1.0	1.0	6.0	0.0	2.0
6	10.0	1.0	0.0	6.0	0.0	1.0
7	1.0	0.0	0.0	2.0	0.0	0.0
8	1.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0
10	3.0	0.0	0.0	0.0	0.0	0.0
Ave.	**3.25**	1.56	1.28	2.43	1.19	1.99

Table 4: Frequencies (%) of the number of required prerequisite skills for the RC datasets.

5 Results of the Dataset Analysis

We now present the results of evaluating the RC datasets according to the two classes of metrics. In the annotation of prerequisite skills, the inter-annotator agreement was 90.1% for 62 randomly sampled questions. The evaluation was performed with respect to the following four aspects: (i) frequencies of prerequisite skills required for each RC dataset; (ii) number of prerequisite skills required per question; (iii) readability metrics for each RC dataset; and (iv) correlation between readability metrics and the number of required prerequisite skills.

(i) Frequencies of prerequisite skills (see Table 3): QA4MRE had the highest scores for frequencies among the datasets. This seems to reflect

Metrics	QA4MRE	MCTest	SQuAD	WDW	MARCO	NewsQA
NumChar	5.026	3.892	**5.378**	4.988	5.016	5.017
NumSyll	1.663	1.250	**1.791**	1.657	1.698	1.635
MLS	28.488	11.858	23.479	**29.146**	19.634	22.933
AWL	0.067	0.003	**0.071**	0.033	0.047	0.038
ModVar	0.174	0.114	**0.188**	0.150	0.186	0.138
CoOrd	**0.922**	0.309	0.722	0.467	0.651	0.507
Coleman	12.553	4.333	**14.095**	12.398	11.836	12.138
DC/C	**0.343**	0.223	0.243	0.254	0.220	0.264
CN/C	1.948	0.614	1.887	**2.310**	1.935	1.702
AdvVar	**0.038**	0.035	0.032	0.019	0.022	0.019
F–K	14.953	3.607	14.678	**15.304**	12.065	12.624
Words	**1545.7**	174.1	130.4	253.7	70.7	638.4

Table 5: Results of readability metrics for the RC datasets. *F–K* is the Flesch–Kincaid grade level (Kincaid et al., 1975). *Words* is the average word count of the context for each question.

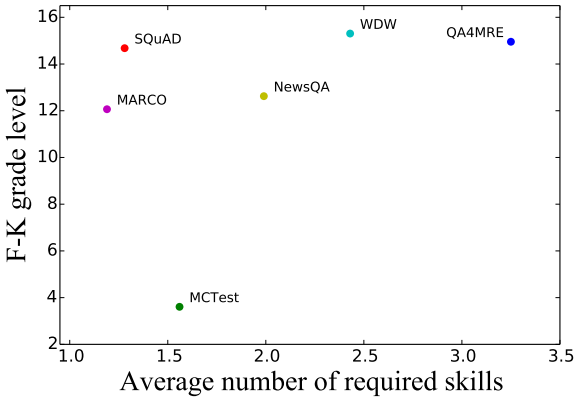


Figure 2: Flesch–Kincaid grade levels and average number of required prerequisite skills for the RC datasets.

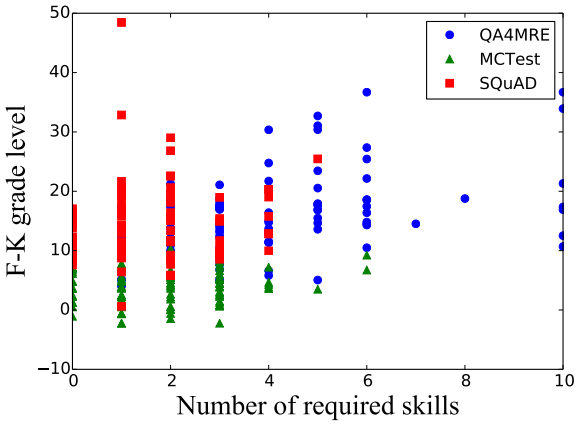


Figure 3: Flesch–Kincaid grade levels and number of required prerequisite skills for all questions in the selected RC datasets.

the fact that QA4MRE involves technical documents that contain a wide range of knowledge, multiple clauses, and punctuation. Moreover, the questions are devised by experts.

MCTest achieved a high score for several skills (best for *causal relation* and *meta-knowledge* and second-best for *coreference resolution* and *spatiotemporal relation*), but a low score for *punctuation*. These scores seem to be because the MCTest dataset consists of narratives.

Another dataset that achieved notable scores is Who-did-What. This dataset achieved the highest score for *ellipsis*. This is because the questions of Who-did-What are automatically generated from articles not used as context. This methodology tends to avoid textual overlap between a question and its context, thereby requiring frequently the skills of *ellipsis*, *bridging*, and *elaboration*.

With regard to *nonsense*, MS MARCO and Who-did-What received relatively high scores. This appears to have been caused by the automated sourcing methods, which may generate a separation between the contents of the context and question (i.e., web segments and a search query in MS MARCO, and a context article and question article in Who-did-What). In contrast, NewsQA had no nonsense questions. Although this result was affected by our filtering (described in Appendix A), it is important to note that the NewsQA dataset includes annotations of meta-information whether or not a question makes sense (*is_question_bad*).

(ii) Number of required prerequisite skills (see Table 4): QA4MRE had the highest score. On average, each question required 3.25 skills. There were few questions in QA4MRE that required zero or one skill, whereas such questions were contained more frequently in other datasets. Table 4 also indicates that more than 90% of the MS MARCO questions required fewer than three skills according to the annotation.

(iii) Readability metrics for each dataset (see Table 5): SQuAD and QA4MRE achieved the highest scores for most metrics. This reflects the fact that Wikipedia articles and technical documents usually require a high-grade level of understanding. In contrast, MCTest had the lowest scores, with its dataset consisting of narratives for children.

(iv) Correlation between numbers of required prerequisite skills and readability metrics (see Figures 2 and 3, and Table 6): our main interest was in the correlation between prerequisite skills and readability. To investigate this, we examined the relation between the number of required prerequisite skills and readability metrics.

Metrics	r	p	Metrics	r	p
NumChar	0.068	0.095	CoOrd	0.166	0.000
NumSyll	0.057	0.161	Coleman	0.140	0.001
MLS	0.416	0.000	DC/C	0.188	0.000
AWL	0.114	0.005	CN/C	0.131	0.001
ModVar	0.025	0.545	AdvVar	0.026	0.515
F–K	0.343	0.000	Words	0.355	0.000

Table 6: Pearson's correlation coefficients (r) with the p-values (p) for the readability metrics and number of required prerequisite skills for all questions in the RC datasets.

We used the Flesch–Kincaid grade level (Kincaid et al., 1975) as an intuitive reference for readability. This value represents the typical number of years of education required to understand texts based on counts of syllables, words, and sentences.

Figures 2 and 3 show the relation between two values for each dataset and for each question, respectively. Figure 2 shows the trends of the datasets. QA4MRE was relatively difficult both to read and to answer, whereas SQuAD was difficult to read but easy to answer. For further investigation, we selected three datasets (QA4MRE, MCTest, and SQuAD) and plotted all of their questions in Figure 3. Three separate domains can be seen.

Table 6 presents Pearson's correlation coefficients between the number of required prerequisite skills and each readability metric for all questions in the RC datasets. Although there are weak correlations, from 0.025 to 0.416, these results demonstrate that there is not necessarily a strong correlation between the two values. This leads to the following two insights. First, the readability of RC datasets does not directly affect the difficulty of their questions. That is, RC datasets that are difficult to read are not necessarily difficult to answer. Second, it is possible to create difficult questions from the context that are easy to read. MCTest is a good example. The context texts in the MCTest dataset are easy to read, but the difficulty of its questions compares to that for the other datasets.

To summarize our results in terms of each RC dataset, we can make the following observations.

- **QA4MRE** is difficult both to read and to answer among the datasets analyzed. This would seem to follow its questions being devised by experts.
- **MCTest** is a good example of an RC dataset that is easy to read but difficult to answer. We presume that this is because the corpus genre (i.e., narrative) reflects the trend in required skills for the questions.
- **SQuAD** is difficult to read, along with QA4MRE, but relatively easy to answer compared with the other datasets.
- **Who-did-What** performs well in terms of its query-sourcing method. Although its questions are created automatically, they are sophisticated in terms of knowledge reasoning. However, the automated sourcing method must be improved to exclude nonsense questions.
- **MS MARCO** is a relatively easy dataset in terms of prerequisite skills. However, one problem is that the dataset contained nonsense questions.
- **NewsQA** is advantageous in that it provides meta-information on the reliability of the questions. Such information enabled us to avoid using nonsense questions, as for the training of machine learning models.

6 Discussion

In this section, we discuss several issues regarding the construction of RC datasets and the development of RC systems using our methodology.

How to utilize the two classes of metrics for system development: one possible scenario for developing an RC system is that it is first built to solve an easy-to-read and easy-to-answer dataset. The next step would be to improve the system so that it can solve an easy-to-read but difficult-to-answer dataset (or its converse). Finally, only after it can solve such datasets should the system be applied to difficult-to-read and difficult-to-answer datasets. The metrics of this study may be useful in preparing appropriate datasets for each step by measuring their properties. The datasets can then be ordered according to the grades of the metrics and applied to each step of the development, as in curriculum learning (Bengio et al., 2009) and transfer learning (Pan and Yang, 2010).

Corpus genre: attention should be paid to the genre of the corpus used to construct a dataset. Expository documents such as news articles tend to require factorial understanding. Most existing RC datasets use such texts because of their availability. On the other hand, narrative texts may have a

closer correspondence to our everyday experience, involving the emotions and intentions of characters (Graesser et al., 1994). To build agents that work in the real world, RC datasets may have to be constructed from narratives.

Question type: in contrast to factorial understanding, comprehensive understanding of natural language texts needs a better grasp of *global* coherence (e.g., the main point or moral of the text, the goal of a story, or the intention of characters) from the broad context (Graesser et al., 1994). Most questions in current use require only *local* coherence (e.g., referential relations and thematic roles) within a narrow context. An example of a question based on global coherence would be to give a summary of the text, as used in Hermann et al. (2015). It could be generated automatically by techniques of abstractive text summarization (Rush et al., 2015; Ganesan et al., 2010).

Annotation issues: we found questions for which there were disagreements regarding *non-sense* decisions. For example, some questions can be solved by external knowledge without even seeing their context. Therefore, we should clarify what constitutes a "solvable" or "reasonable" question for RC. In addition, annotators reported that the prerequisite skills did not easily treat questions whose answer was "none of the above" in QA4MRE. We considered these "no answer" questions difficult, in that systems have to decide not to select any of the candidate answers, and our methodology failed to specify them.

Competence in selecting necessary sentences: as mentioned in Section 1, our methodology cannot evaluate competence in selecting sentences that need to be read to answer questions. In a brief analysis, we further investigated sentences in the context of the datasets that were selected in the annotation. Analyses were performed in two ways. For each question, we counted the number of required sentences and their distance apart.[4] The first row of Table 7 gives the average number of required sentences per question for each RC dataset. Although the scores are reasonably close, MCTest required multiple sentences to be read most frequently. The second row gives the average distance apart of the required sentences. QA4MRE required the longest distance because readers had to look for clues in the long context texts. In contrast, SQuAD and MS MARCO had lower scores. Most of their questions seemed to be answered by reading only a single sentence. Of course, the scores for distances will depend on the length of the context texts.

Sentence	QA4MRE	MCTest	SQuAD	WDW	MARCO	NewsQA
Number	1.120	**1.180**	1.040	1.110	1.080	1.170
Distance	**1.880**	0.930	0.090	0.730	0.280	0.540

Table 7: Average number and distance apart of sentences that need to be read to answer a question in the RC datasets.

Metrics of RC for machines: our underlying assumption in this study is that, in the development of interactive agents such as dialogue systems, it is important to make the systems behave in a human-like way. This has also become a distinguishing feature of recent RC task design, and one that has never been explicitly considered in conventional NLP tasks. To date, the difference between human and machine RC has not attracted much research attention. We believe that our human-based evaluation metrics and analysis will help researchers to develop a method for the step-by-step construction of better RC datasets and improved RC systems.

7 Conclusion

In this study, we adopted evaluation metrics that comprise two classes, namely refined *prerequisite skills* and *readability*, for analyzing the quality of RC datasets. We applied these classes to six existing datasets and highlighted their characteristics according to each metric. Our dataset analysis suggests that the readability of RC datasets does not directly affect the difficulty of the questions and that it is possible to create an RC dataset that is easy to read but difficult to answer. In future work, we plan to use the analysis from the present study in constructing a system that can be applied to multiple datasets.

Acknowledgments

We would like to thank anonymous reviewers for their insightful comments. This work was supported by JSPS KAKENHI Grant Numbers 15H02754 and 16K16120.

[4]The distance of sentences was calculated as follows. If a question required only one sentence to be read, its distance was zero. If a question required two adjacent sentences to be read, its distance was one. If a question required more than two sentences to be read, its distance was the sum of the distances of any two sentences.

References

Sandra Aluisio, Lucia Specia, Caroline Gasperin, and Carolina Scarton. 2010. Readability assessment for text simplification. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, pages 1–9. https://aclweb.org/anthology/W10-1001.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. ACM, pages 41–48. https://doi.org/10.1145/1553374.1553380.

Luisa Bentivogli, Elena Cabrio, Ido Dagan, Danilo Giampiccolo, Medea Lo Leggio, and Bernardo Magnini. 2010. Building textual entailment specialized data sets: a methodology for isolating linguistic phenomena relevant to inference. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*. Citeseer.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 2358–2367. https://aclweb.org/anthology/P16-1223.

Herbert H Clark. 1975. Bridging. In *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*. Association for Computational Linguistics, pages 169–174. https://doi.org/10.3115/980190.980237.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, Springer, pages 177–190. https://doi.org/10.1007/11736790_9.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies* 6(4):1–220.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*. Association for Computational Linguistics, pages 340–348.

Arthur C Graesser, Murray Singer, and Tom Trabasso. 1994. Constructing inferences during narrative text comprehension. *Psychological review* 101(3):371. https://doi.org/10.1037/0033-295X.101.3.371.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the International Conference on Learning Representations*.

Lynette Hirschman, Marc Light, Eric Breck, and John D Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 325–332. https://doi.org/10.3115/1034678.1034731.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Chief of Naval Technical Training, Research Branch Report 8-75.

Walter Kintsch. 1988. The role of knowledge in discourse comprehension: A construction-integration model. *Psychological review* 95(2):163. https://doi.org/10.1037/0033-295X.95.2.163.

Walter Kintsch. 1993. Information accretion and reduction in text processing: Inferences. *Discourse processes* 16(1-2):193–202.

Peter LoBue and Alexander Yates. 2011. Types of common-sense knowledge needed for recognizing textual entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 329–334. https://aclweb.org/anthology/P11-2057.

Danielle S McNamara and Joe Magliano. 2009. Toward a comprehensive model of comprehension. *Psychology of learning and motivation* 51:297–384. https://doi.org/10.1016/S0079-7421(09)51009-2.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR* abs/1611.09268.

Peter Norvig. 1989. Marker passing as a weak method for text inferencing. *Cognitive Science* 13(4):569–620. https://doi.org/10.1207/s15516709cog1304_4.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did What: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2230–2235. https://aclweb.org/anthology/D16-1241.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10):1345–1359. https://doi.org/10.1109/TKDE.2009.191.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1525–1534. https://aclweb.org/anthology/P16-1144.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 186–195. https://aclweb.org/anthology/D08-1020.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Abigail Razon and John Barnden. 2015. A new approach to automated text readability classification based on concept indexing with integrated part-of-speech n-gram features. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*. pages 521–528. https://aclweb.org/anthology/R15-1068.

Matthew Richardson, J.C. Christopher Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 193–203. http://aclweb.org/anthology/D13-1020.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 379–389. https://aclweb.org/anthology/D15-1044.

Mark Sammons, V.G.Vinod Vydiswaran, and Dan Roth. 2010. "ask not what textual entailment can do for you...". In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1199–1208. https://aclweb.org/anthology/P10-1122.

Ellery Smith, Nicola Greco, Matko Bosnjak, and Andreas Vlachos. 2015. A strong lexical matching method for the machine comprehension test. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1693–1698.

Saku Sugawara and Akiko Aizawa. 2016. An analysis of prerequisite skills for reading comprehension. In *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*. Association for Computational Linguistics, pages 1–5. https://aclweb.org/anthology/W16-6001.

Saku Sugawara, Hikaru Yokono, and Akiko Aizawa. 2017. Prerequisite skills for reading comprehension: Multi-perspective analysis of mctest datasets and systems. In *AAAI Conference on Artificial Intelligence*. pages 3089–3096.

Richard Sutcliffe, Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Corina Forascu, Yassine Benajiba, and Petya Osenova. 2013. Overview of QA4MRE main task at CLEF 2013. *Working Notes, CLEF* .

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. NewsQA: A machine comprehension dataset. *CoRR* abs/1611.09830.

Sowmya Vajjala and Detmar Meurers. 2012. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics, pages 163–173. https://aclweb.org/anthology/W12-2019.

Teun Adrianus van Dijk and Walter Kintsch. 1983. *Strategies of discourse comprehension*. Citeseer.

Ellen M Voorhees et al. 1999. The TREC-8 question answering track report. In *TREC*. volume 99, pages 77–82.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 22–32. https://aclweb.org/anthology/D/D07/D07-1003.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2013–2018. https://aclweb.org/anthology/D15-1237.

A Sampling Methods for Questions

In this appendix, we explain the method of choosing questions for annotation.

QA4MRE (Sutcliffe et al., 2013): the gold-standard dataset comprised four different topics and four documents for each topic. We randomly selected 100 main and auxiliary questions so that at least one question for each document was included.

MCTest (Richardson et al., 2013): this dataset comprised two sets: MC160 and MC500. Their development sets had 80 tasks in total, with each containing context texts and four questions. We randomly chose 25 tasks (100 questions) from the development sets.

SQuAD (Rajpurkar et al., 2016): this dataset included Wikipedia articles involving various topics, with the articles being divided into paragraphs. We randomly chose 100 paragraphs from 15 articles and used only one question from each paragraph for the annotation.

Who-did-What (WDW) (Onishi et al., 2016): this dataset was constructed from the English Gigaword newswire corpus (v5). Its questions were automatically created using a different article from that used for context. In addition, questions that could be solved by a simple baseline method were excluded from the dataset.

MS MARCO (MARCO) (Nguyen et al., 2016): each task in this dataset comprised several segments, one question, and its answer. We randomly chose 100 tasks (100 questions) and only used segments whose attribute was *is_selected* = 1 as context.

NewsQA (Trischler et al., 2016): we randomly chose questions that satisfied the following conditions: *is_answer_absent* = 0, *is_question_bad* = 0, and *validated_answers* do not include *bad_question* or *none*.

A Minimal Span-Based Neural Constituency Parser

Mitchell Stern **Jacob Andreas** **Dan Klein**
Computer Science Division
University of California, Berkeley
{mitchell,jda,klein}@cs.berkeley.edu

Abstract

In this work, we present a minimal neural model for constituency parsing based on independent scoring of labels and spans. We show that this model is not only compatible with classical dynamic programming techniques, but also admits a novel greedy top-down inference algorithm based on recursive partitioning of the input. We demonstrate empirically that both prediction schemes are competitive with recent work, and when combined with basic extensions to the scoring model are capable of achieving state-of-the-art single-model performance on the Penn Treebank (91.79 F1) and strong performance on the French Treebank (82.23 F1).

1 Introduction

This paper presents a minimal but surprisingly effective span-based neural model for constituency parsing. Recent years have seen a great deal of interest in parsing architectures that make use of recurrent neural network (RNN) representations of input sentences (Vinyals et al., 2015). Despite evidence that linear RNN decoders are implicitly able to respect some nontrivial well-formedness constraints on structured outputs (Graves, 2013), researchers have consistently found that the best performance is achieved by systems that explicitly require the decoder to generate well-formed tree structures (Chen and Manning, 2014).

There are two general approaches to ensuring this structural consistency. The most common is to encode the output as a sequence of operations within a transition system which constructs trees incrementally. This transforms the parsing problem back into a sequence-to-sequence problem, while making it easy to force the decoder to take only actions guaranteed to produce well-formed outputs. However, transition-based models do not admit fast dynamic programs and require careful feature engineering to support exact search-based inference (Thang et al., 2015). Moreover, models with recurrent state require complex training procedures to benefit from anything other than greedy decoding (Wiseman and Rush, 2016).

An alternative line of work focuses on *chart parsers*, which use log-linear or neural scoring potentials to parameterize a tree-structured dynamic program for maximization or marginalization (Finkel et al., 2008; Durrett and Klein, 2015). These models enjoy a number of appealing formal properties, including support for exact inference and structured loss functions. However, previous chart-based approaches have required considerable scaffolding beyond a simple well-formedness potential, e.g. pre-specification of a complete context-free grammar for generating output structures and initial pruning of the output space with a weaker model (Hall et al., 2014). Additionally, we are unaware of any recent chart-based models that achieve results competitive with the best transition-based models.

In this work, we present an extremely simple chart-based neural parser based on independent scoring of labels and spans, and show how this model can be adapted to support a greedy top-down decoding procedure. Our goal is to preserve the basic algorithmic properties of span-oriented (rather than transition-oriented) parse representations, while exploring the extent to which neural representational machinery can replace the additional structure required by existing chart parsers. On the Penn Treebank, our approach outperforms a number of recent models for chart-based and transition-based parsing—including the state-of-the-art models of Cross and Huang (2016) and Liu and Zhang (2016)—achieving an F1 score of 91.79. We additionally obtain a strong F1 score of 82.23 on the French Treebank.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 818–827
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1076

2 Model

A constituency tree can be regarded as a collection of labeled spans over a sentence. Taking this view as a guiding principle, we propose a model with two components, one which assigns scores to span labels and one which assigns scores directly to span existence. The former is used to determine the labeling of the output, and the latter provides its structure.

At the core of both of these components is the issue of span representation. Given that a span's correct label and its quality as a constituent depend heavily on the context in which it appears, we naturally turn to recurrent neural networks as a starting point, since they have previously been shown to capture contextual information suitable for use in a variety of natural language applications (Bahdanau et al., 2014; Wang et al., 2015)

In particular, we run a bidirectional LSTM over the input to obtain context-sensitive forward and backward encodings for each position i, denoted by $\mathbf{f}_i$ and $\mathbf{b}_i$, respectively. Our representation of the span (i, j) is then the concatenatation the vector differences $\mathbf{f}_j - \mathbf{f}_i$ and $\mathbf{b}_i - \mathbf{b}_j$. This corresponds to a bidirectional version of the LSTM-Minus features first proposed by Wang and Chang (2016).

On top of this base, our label and span scoring functions are implemented as one-layer feedforward networks, taking as input the concatenated span difference and producing as output either a vector of label scores or a single span score. More formally, letting $\mathbf{s}_{ij}$ denote the vector representation of span (i, j), we define

$$s_{\text{labels}}(i, j) = \mathbf{V}_\ell g(\mathbf{W}_\ell \mathbf{s}_{ij} + \mathbf{b}_\ell),$$
$$s_{\text{span}}(i, j) = \mathbf{v}_s^\top g(\mathbf{W}_s \mathbf{s}_{ij} + \mathbf{b}_s),$$

where g denotes an elementwise nonlinearity. For notational convenience, we also let the score of an individual label ℓ be denoted by

$$s_{\text{label}}(i, j, \ell) = [s_{\text{labels}}(i, j)]_\ell,$$

where the right-hand side is the corresponding element of the label score vector.

One potential issue is the existence of unary chains, corresponding to nested labeled spans with the same endpoints. We take the common approach of treating these as additional atomic labels alongside all elementary nonterminals. To accommodate n-ary trees, our inventory additionally includes a special empty label $\varnothing$ used for spans that are not themselves full constituents but arise during the course of implicit binarization.

Our model shares several features in common with that of Cross and Huang (2016). In particular, our representation of spans and the form of our label scoring function were directly inspired by their work, as were our handling of unary chains and our use of an empty label. However, our approach differs in its treatment of structural decisions, and consequently, the inference algorithms we describe below diverge significantly from their transition-based framework.

3 Chart Parsing

Our basic model is compatible with traditional chart-based dynamic programming. Representing a constituency tree T by its labeled spans,

$$T := \{(\ell_t, (i_t, j_t)) : t = 1, \ldots, |T|\},$$

we define the score of a tree to be the sum of its constituent label and span scores,

$$s_{\text{tree}}(T) = \sum_{(\ell,(i,j))\in T} \left[s_{\text{label}}(i, j, \ell) + s_{\text{span}}(i, j)\right].$$

To find the tree with the highest score for a given sentence, we use a modified CKY recursion. As with classical chart parsing, the running time of our procedure is $O(n^3)$ for a sentence of length n.

3.1 Dynamic Program for Inference

The base case is a span $(i, i + 1)$ consisting of a single word. Since every valid tree must include all singleton spans, possibly with an empty label, we need not consider the span score in this case and perform only a single maximization over the choice of label:

$$s_{\text{best}}(i, i + 1) = \max_\ell \left[s_{\text{label}}(i, i + 1, \ell)\right].$$

For a general span (i, j), we define the score of the split (i, k, j) as the sum of its subspan scores,

$$s_{\text{split}}(i, k, j) = s_{\text{span}}(i, k) + s_{\text{span}}(k, j). \quad (1)$$

For convenience, we also define an augmented split score incorporating the scores of the corresponding subtrees,

$$\tilde{s}_{\text{split}}(i, k, j) = s_{\text{split}}(i, k, j) + s_{\text{best}}(i, k) + s_{\text{best}}(k, j).$$

Using these quantities, we can then write the general joint label and split decision as

$$s_{\text{best}}(i,j) = \max_{\ell,k} \left[s_{\text{label}}(i,j,\ell) + \tilde{s}_{\text{split}}(i,k,j) \right]. \tag{2}$$

Because our model assigns independent scores to labels and spans, this maximization decomposes into two disjoint subproblems, greatly reducing the size of the state space:

$$s_{\text{best}}(i,j) = \max_{\ell} \left[s_{\text{label}}(i,j,\ell) \right] + \max_{k} \left[\tilde{s}_{\text{split}}(i,k,j) \right].$$

We also note that the span scores $s_{\text{span}}(i,j)$ for each span (i,j) in the sentence can be computed once at the beginning of the procedure and shared across different subproblems with common left or right endpoints, allowing for a quadratic rather than cubic number of span score computations.

3.2 Margin Training

Training the model under this inference scheme is accomplished using a margin-based approach. When presented with an example sentence and its corresponding parse tree T^*, we compute the best prediction under the current model using the above dynamic program,

$$\widehat{T} = \operatorname*{argmax}_{T} \left[s_{\text{tree}}(T) \right].$$

If $\widehat{T} = T^*$, then our prediction was correct and no changes need to be made. Otherwise, we incur a hinge penalty of the form

$$\max\left(0,\ 1 - s_{\text{tree}}(T^*) + s_{\text{tree}}(\widehat{T})\right)$$

to encourage the model to keep a margin of at least 1 between the gold tree and the best alternative. The loss to be minimized is then the sum of penalties across all training examples.

Prior work has found that it can be beneficial in a variety of applications to incorporate a structured loss function into this margin objective, replacing the hinge penalty above with one of the form

$$\max\left(0,\ \Delta(\widehat{T}, T^*) - s_{\text{tree}}(T^*) + s_{\text{tree}}(\widehat{T})\right)$$

for a loss function Δ that measures the similarity between the prediction $\widehat{T}$ and the reference T^*. Here we take Δ to be a Hamming loss on labeled spans. To incorporate this loss into the training objective, we modify the dynamic program of Section 3.1 to support loss-augmented decoding (Taskar et al., 2005). Since the label decisions are isolated from the structural decisions, it suffices to replace every occurrence of the label scoring function $s_{\text{label}}(i,j,\ell)$ by

$$s_{\text{label}}(i,j,\ell) + \mathbf{1}(\ell \neq \ell^*_{ij}),$$

where ℓ^*_{ij} is the label of span (i,j) in the gold tree T^*. This has the effect of requiring larger margins between the gold tree and predictions that contain more mistakes, offering a greater degree of robustness and better generalization.

4 Top-Down Parsing

While we have so far motivated our model from the perspective of classical chart parsing, it also allows for a novel inference algorithm in which trees are constructed greedily from the top down. At a high level, given a span, we independently assign it a label and pick a split point, then repeat this process for the left and right subspans; the recursion bottoms out with length-one spans that can no longer be split. Figure 1 gives an illustration of the process, which we describe in more detail below.

The base case is again a singleton span $(i, i+1)$, and follows the same form as the base case for the chart parser. In particular, we select the label $\widehat{\ell}$ that satisfies

$$\widehat{\ell} = \operatorname*{argmax}_{\ell} \left[s_{\text{label}}(i, i+1, \ell) \right],$$

omitting span scores from consideration since singleton spans cannot be split.

To construct a tree over a general span (i,j), we aim to solve the maximization problem

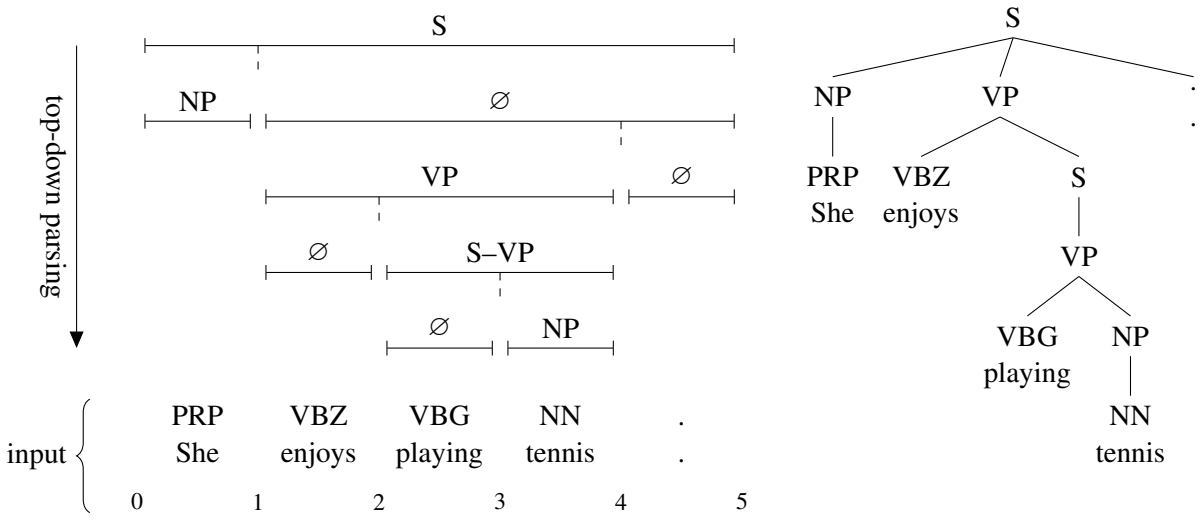
$$(\widehat{\ell}, \widehat{k}) = \operatorname*{argmax}_{\ell,k} \left[s_{\text{label}}(i,j,\ell) + s_{\text{split}}(i,k,j) \right],$$

where $s_{\text{split}}(i,k,j)$ is defined as in Equation (1). The independence of our label and span scoring functions again yields the decomposed form

$$\begin{aligned} \widehat{\ell} &= \operatorname*{argmax}_{\ell} \left[s_{\text{label}}(i,j,\ell) \right], \\ \widehat{k} &= \operatorname*{argmax}_{k} \left[s_{\text{split}}(i,k,j) \right], \end{aligned} \tag{3}$$

leading to a significant reduction in the size of the state space.

To generate a tree for the whole sentence, we call this procedure on the full sentence span $(0,n)$ and return the result. As there are $O(n)$ spans each



(a) Execution of the top-down parsing algorithm.

(b) Output parse tree.

Figure 1: An execution of our top-down parsing algorithm (a) and the resulting parse tree (b) for the sentence "She enjoys playing tennis." Part-of-speech tags, shown here together with the words, are predicted externally and are included as part of the input to our system. Beginning with the full sentence span $(0, 5)$, the label S and the split point 1 are predicted, and recursive calls are made on the child spans $(0, 1)$ and $(1, 5)$. The left child span $(0, 1)$ is assigned the label NP, and with no further splits to make, recursion terminates on this branch. The right child span $(1, 5)$ is assigned the empty label $\varnothing$, indicating that it does not represent a constituent in the tree. A split point of 4 is selected, and further recursive calls are made on the grandchild spans $(1, 4)$ and $(4, 5)$. This process of labeling and splitting continues until every branch of recursion bottoms out in singleton spans, at which point the full parse tree can be returned. Note that the unary chain S–VP is produced in a single labeling step.

requiring one label evaluation and at most $n - 1$ split point evaluations, the running time of the procedure is $O(n^2)$.

The algorithm outlined here bears a strong resemblance to the chart parsing dynamic program discussed in Section 3, but differs in one key aspect. When performing inference from the bottom up, we have already computed the scores of all of the subtrees below the current span, and we can take this knowledge into consideration when selecting a split point. In contrast, when producing a tree from the top down, we can only select a split point based on top-level evaluations of span quality, without knowing anything about the subtrees that will be generated below them. This difference is manifested in the augmented split score $\tilde{s}_{\text{split}}$ used in the definition of s_{best} in Equation (2), where the scores of the subtrees associated with a split point are included in the chart recursion but necessarily excluded from the top-down recursion.

While this apparent deficiency may be a cause for concern, we demonstrate the surprising empirical result in Section 6 that there is no loss in performance when moving from the globally-optimal chart parser to the greedy top-down procedure.

4.1 Margin Training

As with the chart parsing formulation, we also use a margin-based method for learning under the top-down model. However, rather than requiring separation between the scores of full trees, we instead enforce a local margin at every decision point.

For a span (i, j) occurring in the gold tree, let ℓ^* and k^* represent the correct label and split point, and let $\widehat{\ell}$ and $\widehat{k}$ be the predictions made by computing the maximizations in Equation (3). If $\widehat{\ell} \neq \ell^*$, meaning the prediction is incorrect, we incur a hinge penalty of the form

$$\max\left(0, 1 - s_{\text{label}}(i, j, \ell^*) + s_{\text{label}}(i, j, \widehat{\ell})\right).$$

Similarly, if $\widehat{k} \neq k^*$, we incur a hinge penalty of the form

$$\max\left(0, 1 - s_{\text{split}}(i, k^*, j) + s_{\text{split}}(i, \widehat{k}, j)\right).$$

To obtain the loss for a given training example, we trace out the actions corresponding to the gold tree and accumulate the above penalties over all decision points. As before, the total loss to be minimized is the sum of losses across all training examples.

Loss augmentation is also beneficial for the local decisions made by the top-down model, and can be implemented in a manner akin to the one discussed in Section 3.2.

4.2 Training with Exploration

The hinge penalties given above are only defined for spans (i, j) that appear in the example tree. The model must therefore be constrained at training time to follow decisions that exactly reproduce the gold tree, since supervision cannot be provided otherwise. As a result, the model is never exposed to its mistakes, which can lead to a lack of calibration and poor performance at test time.

To circumvent this issue, a *dynamic oracle* can be defined to inform the model about correct behavior even after it has deviated from the gold tree. Cross and Huang (2016) propose such an oracle for a related transition-based parsing system, and prove its optimality for the F1 metric on labeled spans. We adapt their result here to obtain a dynamic oracle for the present model with similar guarantees.

The oracle for labeling decisions carries over without modification: the correct label for a span is the label assigned to that span if it is part of the gold tree, or the empty label $\varnothing$ otherwise.

For split point decisions, the oracle can be broken down into two cases. If a span (i, j) appears as a constituent in the gold tree T, we let $b(i, j)$ denote the collection of its interior boundary points. For example, if the constituent over $(1, 7)$ has children spanning $(1, 3)$, $(3, 6)$, and $(6, 7)$, then we would have the two interior boundary points, $b(1, 7) = \{3, 6\}$. The oracle for a span appearing in the gold tree is then precisely the output of this function. Otherwise, for spans (i, j) not corresponding to gold constituents, we must instead identify the smallest enclosing gold constituent:

$$(i^*, j^*) = \min\{(i', j') \in T : i' \leq i < j \leq j'\},$$

where the minimum is taken with respect to the partial ordering induced by span length. The output of the oracle is then the set of interior boundary points of this enclosing span that also lie inside the original, $\{k \in b(i^*, j^*) : i < k < j\}$. The proof of correctness is similar to the proof in Cross and Huang (2016); we refer to the Dynamic Oracle section in their paper for a more detailed discussion.

As presented, the dynamic oracle for split point decisions returns a collection of one or more splits rather than a single correct answer. Any of these is a valid choice, with different splits corresponding to different binarizations of the original n-ary tree. We choose to use the leftmost split point for consistency in our implementation, but remark that the oracle split with the highest score could also be chosen at training time to allow for additional flexibility.

Having defined the dynamic oracle for our system, we note that training with exploration can be implemented by a single modification to the procedure described in Section 4.1. Local penalties are accumulated as before, but instead of tracing out the decisions required to produce the gold tree, we instead follow the decisions predicted by the model. In this way, supervision is provided at states within the prediction procedure that are more likely to arise at test time when greedy inference is performed.

5 Scoring and Loss Alternatives

The model presented in Section 2 is designed to be as simple as possible. However, there are many variations of the label and span scoring functions that could be explored; we discuss some of the options here.

5.1 Top-Middle-Bottom Label Scoring

Our basic model treats the empty label, elementary nonterminals, and unary chains each as atomic units, obscuring similarities between unary chains and their component nonterminals or between different unary chains with common prefixes or suffixes. To address this lack of structure, we consider an alternative scoring scheme in which labels are predicted in three parts: a top nonterminal, a middle unary chain, and a bottom nonterminal (each of which is possibly empty).[1] This not only allows for parameter sharing across labels with common subcomponents, but also has the added benefit of allowing the model to produce novel unary chains at test time.

[1] In more detail, $\varnothing$ decomposes as $(\varnothing, \varnothing, \varnothing)$, X decomposes as $(X, \varnothing, \varnothing)$, X–Y decomposes as $(X, \varnothing, Y)$, and X–Z_1–$\cdots$–Z_k–Y decomposes as $(X, Z_1$–$\cdots$–$Z_k, Y)$.

More precisely, we introduce the decomposition

$$s_{\text{label}}(i, j, (\ell_t, \ell_m, \ell_b)) = s_{\text{top}}(i, j, \ell_t) + s_{\text{middle}}(i, j, \ell_m) + s_{\text{bottom}}(i, j, \ell_b),$$

where s_{top}, s_{middle}, and s_{bottom} are independent one-layer feedforward networks of the same form as s_{label} that output vectors of scores for all label tops, label middle chains, and label bottoms encountered in the training corpus, respectively. The best label for a span (i, j) is then computed by solving the maximization problem

$$\max_{\ell_t, \ell_m, \ell_b} \left[s_{\text{label}}(i, j, (\ell_t, \ell_m, \ell_b)) \right],$$

which decomposes into three independent subproblems corresponding to the three label components. The final label is obtained by concatenating ℓ_t, ℓ_m, and ℓ_b, with empty components being omitted from the concatenation.

5.2 Left and Right Span Scoring

The basic model uses the same span scoring function s_{span} to assign a score to the left and right subspans of a given span. One simple extension is to replace this by a pair of distinct left and right feedforward networks of the same form, giving the decomposition

$$s_{\text{split}}(i, k, j) = s_{\text{left}}(i, k) + s_{\text{right}}(k, j).$$

5.3 Span Concatenation Scoring

Since span scores are only used to score splits in our model, we also consider directly scoring a split by feeding the concatenation of the span representations of the left and right subspans through a single feedforward network, giving

$$s_{\text{split}}(i, k, j) = \mathbf{v}_s^\top g\left(\mathbf{W}_s[\mathbf{s}_{ik}; \mathbf{s}_{kj}] + \mathbf{b}_s\right).$$

This is similar to the structural scoring function used by Cross and Huang (2016), although whereas they additionally include features for the outside spans $(0, i)$ and (j, n) in their concatenation, we omit these from our implementation, finding that they do not improve performance.

5.4 Deep Biaffine Span Scoring

Inspired by the success of deep biaffine scoring in recent work by Dozat and Manning (2016) for dependency parsing, we also consider a split scoring function of a similar form for our model. Specifically, we let $\mathbf{h}_{ik} = f_{\text{left}}(\mathbf{s}_{ik})$ and $\mathbf{h}_{kj} = f_{\text{right}}(\mathbf{s}_{kj})$ be deep left and right span representations obtained by passing the child vectors through corresponding left and right feedforward networks. We then define the biaffine split scoring function

$$s_{\text{split}}(i, k, j) = \mathbf{h}_{ik}^\top \mathbf{W}_s \mathbf{h}_{kj} + \mathbf{v}_{\text{left}}^\top \mathbf{h}_{ik} + \mathbf{v}_{\text{right}}^\top \mathbf{h}_{kj},$$

which consists of the sum of a bilinear form between the two hidden representations together with two inner products.

5.5 Structured Label Loss

The three-way label scoring scheme described in Section 5.1 offers one path towards the incorporation of label structure into the model. We additionally consider a structured Hamming loss on labels. More specifically, given two labels ℓ_1 and ℓ_2 consisting of zero or more nonterminals, we define the loss as $|\ell_1 \setminus \ell_2| + |\ell_2 \setminus \ell_1|$, treating each label as a multiset of nonterminals. This structured loss can be incorporated into the training process using the methods described in Sections 3.2 and 4.1.

6 Experiments

We first describe the general setup used for our experiments. We use the Penn Treebank (Marcus et al., 1993) for our English experiments, with standard splits of sections 2-21 for training, section 22 for development, and section 23 for testing. We use the French Treebank from the SPMRL 2014 shared task (Seddah et al., 2014) with its provided splits for our French experiments. No token preprocessing is performed, and only a single <UNK> token is used for unknown words at test time. The inputs to our system are concatenations of 100-dimensional word embeddings and 50-dimensional part-of-speech embeddings. In the case of the French Treebank, we also include 50-dimensional embeddings of each morphological tag. We use automatically predicted tags for training and testing, obtaining predicted part-of-speech tags for the Penn Treebank using the Stanford tagger (Toutanova et al., 2003) with 10-way jackknifing, and using the provided predicted part-of-speech and morphological tags for the French Treebank. Words are replaced by <UNK> with probability $1/(1+\text{freq}(w))$ during training, where $\text{freq}(w)$ is the frequency of w in the training data.

We use a two-layer bidirectional LSTM for our base span features. Dropout with a ratio selected from $\{0.2, 0.3, 0.4\}$ is applied to all non-recurrent

WSJ Dev, Atomic Labels, Basic 0-1 Label Loss

Parser	Minimal	Left-Right	Concat.	Biaffine
Chart	91.95	92.09	92.15	91.96
Top-Down	92.16	92.25	92.24	92.14

(a)

WSJ Dev, Atomic Labels, Structured Label Loss

Parser	Minimal	Left-Right	Concat.	Biaffine
Chart	91.86	92.12	92.09	91.95
Top-Down	92.12	92.31	92.26	92.20

(b)

WSJ Dev, 3-Part Labels, Basic 0-1 Label Loss

Parser	Minimal	Left-Right	Concat.	Biaffine
Chart	92.08	92.05	91.94	91.79
Top-Down	92.12	92.18	92.14	92.02

(c)

WSJ Dev, 3-Part Labels, Structured Label Loss

Parser	Minimal	Left-Right	Concat.	Biaffine
Chart	91.92	91.96	91.97	91.78
Top-Down	91.98	92.27	92.17	92.06

(d)

Table 1: Development F1 scores on the Penn Treebank. Each table corresponds to a particular choice of label loss (either the basic 0-1 loss or the structured Hamming label loss of Section 5.5) and labeling scheme (either the basic atomic scheme or the top-middle-bottom labeling scheme of Section 5.1). The columns within each table correspond to different split scoring schemes: basic minimal scoring, the left-right scoring of Section 5.2, the concatenation scoring of Section 5.3, and the deep biaffine scoring of Section 5.4.

connections of the LSTM, including its inputs and outputs. We tie the hidden dimension of the LSTM and all feedforward networks, selecting a size from $\{150, 200, 250\}$. All parameters (including word and tag embeddings) are randomly initialized using Glorot initialization (Glorot and Bengio, 2010), and are tuned on development set performance. We use the Adam optimizer (Kingma and Ba, 2014) with its default settings for optimization, with a batch size of 10. Our system is implemented in C++ using the DyNet neural network library (Neubig et al., 2017).

We begin by training the minimal version of our proposed chart and top-down parsers on the Penn Treebank. Out of the box, we obtain test F1 scores of 91.69 for the chart parser and 91.58 for the top-down parser. The higher of these matches the recent state-of-the-art score of 91.7 reported by Liu and Zhang (2016), demonstrating that our simple neural parsing system is already capable of achieving strong results.

Building on this, we explore the effects of different split scoring functions when using either the basic 0-1 label loss or the structured label loss discussed in Section 5.5. Our results are presented in Tables 1a and 1b.

We observe that regardless of the label loss, the minimal and deep biaffine split scoring schemes perform a notch below the left-right and concatenation scoring schemes. That the minimal scoring scheme performs worse than the left-right scheme is unsurprising, since the latter is a strict generalization of the former. It is evident, however, that joint scoring of left and right subspans is not required for strong results—in fact, the left-right scheme which scores child subspans in isolation slightly outperforms the concatenation scheme in all but one case, and is stronger than the deep biaffine scoring function across the board.

Comparing results across the choice of label loss, however, we find that fewer trends are apparent. The scores obtained by training with a 0-1 loss are all within 0.1 of those obtained using a structured Hamming loss, being slightly higher in four out of eight cases and slightly lower in the other half. This leads us to conclude that the more elementary approach is sufficient when selecting atomic labels from a fixed inventory.

We also perform the same set of experiments under the setting where the top-middle-bottom label scoring function described in Section 5.1 is used in place of an atomic label scoring function. These results are shown in Tables 1c and 1d.

A priori, we might expect that exposing additional structure would allow the model to make better predictions, but on the whole we find that the scores in this set of experiments are worse than those in the previous set. Trends similar to before hold across the different choices of scoring functions, though in this case the minimal setting has scores closer to those of the left-right setting, even exceeding its performance in the case of a chart parser with a 0-1 label loss.

Our final test results are given in Table 2, along with the results of other recent single-model parsers trained without external parse data. We

Final Parsing Results on Penn Treebank

Parser	LR	LP	F1
Durrett and Klein (2015)	–	–	91.1
Vinyals et al. (2015)	–	–	88.3
Dyer et al. (2016)	–	–	89.8
Cross and Huang (2016)	90.5	92.1	91.3
Liu and Zhang (2016)	91.3	92.1	91.7
Best Chart Parser	90.63	92.98	91.79
Best Top-Down Parser	90.35	93.23	91.77

Table 2: Comparison of final test F1 scores on the Penn Treebank. Here we only include scores from single-model parsers trained without external parse data.

Final Parsing Results on French Treebank

Parser	LR	LP	F1
Björkelund et al. (2014)	–	–	82.53
Durrett and Klein (2015)	–	–	81.25
Cross and Huang (2016)	81.90	84.77	83.11
Best Chart Parser	80.26	84.12	82.14
Best Top-Down Parser	79.60	85.05	82.23

Table 3: Comparison of final test F1 scores on the French Treebank.

achieve a new state-of-the-art F1 score of 91.79 with our best model. Interestingly, we observe that our parsers have a noticeably higher gap between precision and recall than do other top parsers, likely owing to the structured label loss which penalizes mismatching nonterminals more heavily than it does a nonterminal and empty label mismatch. In addition, there is little difference between the best top-down model and the best chart model, indicating that global normalization is not required to achieve strong results. Processing one sentence at a time on a `c4.4xlarge` Amazon EC2 instance, our best chart and top-down parsers operate at speeds of 20.3 sentences per second and 75.5 sentences per second, respectively, as measured on the test set.

We additionally train parsers on the French Treebank using the same settings from our English experiments, selecting the best model of each type based on development performance. We list our test results along with those of several other recent papers in Table 3. Although we fall short of the scores obtained by Cross and Huang (2016), we achieve competitive performance relative to the neural CRF parser of Durrett and Klein (2015).

7 Related Work

Many early successful approaches to constituency parsing focused on rich modeling of correlations in the *output space*, typically by engineering proabilistic context-free grammars with state spaces enriched to capture long-distance dependencies and lexical phenomena (Collins, 2003; Klein and Manning, 2003; Petrov and Klein, 2007). By contrast, the approach we have described here continues a recent line of work on direct modeling of correlations in the *input space*, by using rich feature representations to parameterize local potentials that interact with a comparatively unconstrained structured decoder. As noted in the introduction, this class of feature-based tree scoring functions can be implemented with either a linear transition system (Chen and Manning, 2014) or a global decoder (Finkel et al., 2008). Kiperwasser and Goldberg (2016) describe an approach closely related to ours but targeted at dependency formalisms, and which easily accommodates both sparse log-linear scoring models (Hall et al., 2014) and deep neural potentials (Henderson, 2004; Ballesteros et al., 2016).

The best-performing constituency parsers in the last two years have largely been transition-based rather than global; examples include the models of Dyer et al. (2016), Cross and Huang (2016) and Liu and Zhang (2016). The present work takes many of the insights developed in these models (e.g. the recurrent representation of spans (Kiperwasser and Goldberg, 2016), and the use of a dynamic oracle and exploration policy during training (Goldberg and Nivre, 2013)) and extends these insights to span-oriented models, which support a wider range of decoding procedures. Our approach differs from other recent chart-based neural models (e.g. Durrett and Klein (2015)) in the use of a recurrent input representation, structured loss function, and comparatively simple parameterization of the scoring function. In addition to the globally optimal decoding procedures for which these models were designed, and in contrast to the left-to-right decoder typically employed by transition-based models, our model admits an additional greedy top-to-bottom inference procedure.

8 Conclusion

We have presented a minimal span-oriented parser that uses a recurrent input representation to score

trees with a sum of independent potentials on their constituent spans and labels. Our model supports both exact chart-based decoding and a novel top-down inference procedure. Both approaches achieve state-of-the-art performance on the Penn Treebank, and our best model achieves competitive performance on the French Treebank. Our experiments show that many of the key insights from recent neural transition-based approaches to parsing can be easily ported to the chart parsing setting, resulting in a pair of extremely simple models that nonetheless achieve excellent performance.

Acknowledgments

We would like to thank Nick Altieri and the anonymous reviewers for their valuable comments and suggestions. MS is supported by an NSF Graduate Research Fellowship. JA is supported by a Facebook graduate fellowship and a Berkeley AI / Huawei fellowship.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A Smith. 2016. Training with exploration improves a greedy stack-lstm parser. *arXiv preprint arXiv:1603.03793* .

Anders Björkelund, Ozlem Cetinoglu, Agnieszka Falenska, Richárd Farkas, Thomas Müller, Wolfgang Seeker, and Zsolt Szántó. 2014. The ims-wrocław-szeged-cis entry at the spmrl 2014 shared task: Reranking and morphosyntax meet unlabeled data. *Notes of the SPMRL* .

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

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics* 29(4):589–637.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *EMNLP*.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR* abs/1611.01734. http://arxiv.org/abs/1611.01734.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. *arXiv preprint arXiv:1507.03641* .

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776* .

Jenny Rose Finkel, Alex Kleeman, and Christopher D Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL*. volume 46, pages 959–967.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*.

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

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

David Leo Wright Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *ACL (1)*. pages 228–237.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv:1603.04351* .

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 423–430.

Jiangming Liu and Yue Zhang. 2016. Shift-reduce constituent parsing with neural lookahead features. *CoRR* abs/1612.00567. http://arxiv.org/abs/1612.00567.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.* 19(2):313–330. http://dl.acm.org/citation.cfm?id=972470.972475.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji,

Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. Assocation for Computational Linguistics.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*. Dublin City University, Dublin, Ireland, pages 103–109. http://www.aclweb.org/anthology/W14-6111.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22Nd International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '05, pages 896–903. https://doi.org/10.1145/1102351.1102464.

Le Quang Thang, Hiroshi Noji, and Yusuke Miyao. 2015. Optimal shift-reduce constituent parsing with structured perceptron. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. volume 1, pages 1534–1544.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '03, pages 173–180. https://doi.org/10.3115/1073445.1073478.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. A unified tagging solution: Bidirectional LSTM recurrent neural network with word embedding. *CoRR* abs/1511.00215. http://arxiv.org/abs/1511.00215.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1218.pdf.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* .

Semantic Dependency Parsing via Book Embedding

Weiwei Sun, Junjie Cao and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{ws,junjie.cao,wanxiaojun}@pku.edu.cn

Abstract

We model a dependency graph as a book, a particular kind of topological space, for semantic dependency parsing. The spine of the book is made up of a sequence of words, and each page contains a subset of noncrossing arcs. To build a semantic graph for a given sentence, we design new Maximum Subgraph algorithms to generate noncrossing graphs on each page, and a Lagrangian Relaxation-based algorithm to combine pages into a book. Experiments demonstrate the effectiveness of the book embedding framework across a wide range of conditions. Our parser obtains comparable results with a state-of-the-art transition-based parser.

1 Introduction

Dependency analysis provides a lightweight and effective way to encode syntactic and semantic information of natural language sentences. One of its branches, syntactic dependency parsing (Kübler et al., 2009) has been an extremely active research area, with high-performance parsers being built and applied for practical use of NLP. Semantic dependency parsing, however, has only been addressed in the literature recently (Oepen et al., 2014, 2015; Du et al., 2015; Zhang et al., 2016; Cao et al., 2017).

Semantic dependency parsing employs a graph-structured semantic representation. On the one hand, it is flexible enough to provide analysis for various semantic phenomena (Ivanova et al., 2012). This very flexibility, on the other hand, brings along new challenges for designing parsing algorithms. For graph-based parsing, no previously defined Maximum Subgraph algorithm has simultaneously a high coverage and a polynomial complexity to low degrees. For transition-based parsing, no principled decoding algorithms, e.g. dynamic programming (DP), has been developed for existing transition systems.

In this paper, we borrow the idea of book embedding from graph theory, and propose a novel framework to build parsers for flexible dependency representations. In graph theory, a *book* is a kind of topological space that consists of a spine and a collection of one or more half-planes. In our "book model" of semantic dependency graph, the spine is made up of a sequence of words, and each half-plane contains a subset of dependency arcs. In particular, the arcs on each page compose a noncrossing dependency graph, a.k.a. planar graph. Though a dependency graph in general is very flexible, its subgraph on each page is rather regular. Under the new perspective, semantic dependency parsing can be cast as a two-step task: Each page is first analyzed separately, and then all the pages are bound coherently.

Our work is motivated by the extant low-degree polynomial time algorithm for first-order Maximum Subgraph parsing for noncrossing dependency graphs (Kuhlmann and Jonsson, 2015). We enhance existing work with new exact second- and approximate higher-order algorithms. Our algorithms facilitate building with high accuracy the partial semantic dependency graphs on each page. To produce a full semantic analysis, we also need to integrate partial graphs on all pages into one coherent book. To this end, we formulate the problem as a combinatorial optimization problem, and propose a Lagrangian Relaxation-based algorithm for solutions.

We implement a practical parser in the new framework with a statistical disambiguation model. We evaluate this parser on four data sets: those used in SemEval 2014 Task 8 (Oepen et al., 2014), and the dependency graphs extracted from

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 828–838
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1077

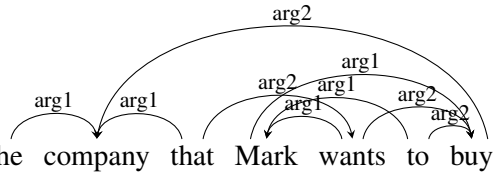


Figure 1: A fragment of a semantic dependency graph.

CCGbank (Hockenmaier and Steedman, 2007). On all data sets, we find that our higher-order parsing models are more accurate than the first-order baseline. Experiments also demonstrate the effectiveness of our page binding algorithm. Our new parser can be taken as a graph-based parser extended for more general dependency graphs. It parallels the state-of-the-art transition-based system of Zhang et al. (2016) in performance.

The implementation of our parser is available at `http://www.icst.pku.edu.cn/lcwm/grass`.

2 Background

2.1 Semantic Dependency Graphs

A dependency graph $G = (V, A)$ is a labeled directed graph for a sentence $s = w_1, \ldots, w_n$. The vertex set V consists of n vertices, each of which corresponds to a word and is indexed by an integer. The arc set A represents the labeled dependency relations of the particular analysis G. Specifically, an arc, viz. $a_{(i,j,l)}$, represents a dependency relation l from head w_i to dependent w_j.

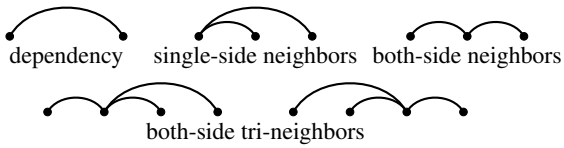
Semantic dependency parsing is the task of mapping a natural language sentence into a formal meaning representation in the form of a dependency graph. Figure 1 shows a graph fragment of a noun phrase. This semantic graph is grounded on Combinatory Categorial Grammar (CCG; Steedman, 2000), and can be taken as a proxy for predicate–argument structure. The graph includes most semantically relevant non-anaphoric local (e.g. from "wants" to "Mark") and long-distance (e.g. from "buy" to "company") dependencies.

2.2 Maximum Subgraph Parsing

Usually, syntactic dependency analysis employs *tree*-shaped representations. Dependency parsing, thus, can be formulated as the search for a maximum spanning tree (MST) of an arc-weighted graph. For semantic dependency parsing, where the target representations are not necessarily trees, Kuhlmann and Jonsson (2015) proposed to generalize the MST model to other types of subgraphs. In general, dependency parsing is formulated as the search for Maximum Subgraph for graph class $\mathcal{G}$: Given a graph $G = (V, A)$, find a subset $A' \subseteq A$ with maximum total weight such that the induced subgraph $G' = (V, A')$ belongs to $\mathcal{G}$. Formally, we have the following optimization problem:

$$G'(s) = \arg \max_{H \in \mathcal{G}(s,G)} \sum_{p \in H} \textsc{ScorePart}(s, p)$$

Here, $\mathcal{G}(s, G)$ is the set of all graphs that belong to $\mathcal{G}$ and are compatible with s and G. For parsing, G is usually a complete graph. $\textsc{ScorePart}(s, p)$ evaluates the event that a small subgraph p of a candidate graph H is good. We define the *order* of a part according to the number of dependencies it contains, in analogy with tree parsing in terminology. Previous work only discussed the first-order case for Maximum Subgraph parsing (Kuhlmann and Jonsson, 2015). In this paper, we are also interested in higher-order parsing, with a special focus on factorizations utilizing the following parts:



If $\mathcal{G}$ is the set of projective trees or non-crossing graphs the first-order Maximum Subgraph problem can be solved in cubic-time (Eisner, 1996; Kuhlmann and Jonsson, 2015). Unfortunately, these two graph classes are not expressive enough to encode semantic dependency graphs. Moreover, this problem for several well-motivated graph classes, including acyclic or 2-planar graphs, is NP-hard, even if one only considers first-order factorization. The lack of appropriate decoding algorithms results in one major challenge for semantic dependency parsing.

2.3 Book Embedding

This section introduces the basic idea about book embedding from a graph theoretical point of view.

Definition 1. *A book is a kind of topological space that consists of a line, called the spine, together with a collection of one or more half-planes, called the pages, each having the spine as its boundary.*

Definition 2. *A book embedding of a finite graph G onto a book B satisfies the following conditions.*

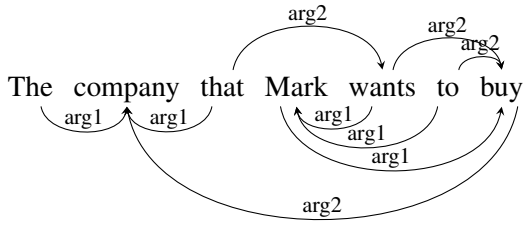


Figure 2: Book embedding for the graph in Figure 1. Arcs are assigned to two pages.

1. *Every vertex of G is depicted as a point on the spine of B.*
2. *Every edge of G is depicted as a curve that lies within a single page of B.*
3. *Every page of B does not have any edge crossings.*

A book embedding separates a graph into several subgraphs, each of which contains all vertices, but only a subset of arcs that are not crossed with each other. This kind of graph is named noncrossing dependency graph by Kuhlmann and Jonsson (2015) and planar by Titov et al. (2009), Gómez-Rodríguez and Nivre (2010) and many others.

We can formalize a semantic dependency graph as a *book*. Take the graph in Figure 1 for example. We can separate the edges into two sets and take each set as a single page, as shown in Figure 2.

Empirically, a semantic dependency graph is sparse enough that it can be that it can be usually embedded onto a very *thin* book. To measure the thickness, we can use pagenumber that is defined as follows.

Definition 3. *The book pagenumber of G is the minimum number of pages required for a book embedding of G.*

We look into the pagenumber of graphs on four linguistic graph banks (as defined in Section 5). These corpora are also used for training and evaluating our data-driven parsers. The pagenumbers are calculated using sentences in the training sets. Table 1 lists the percentages of complete graphs that can be accounted with books of different thickness. The percentages of noncrossing graphs, i.e. graphs that have pagenumber 1, vary between 48.23% and 78.26%. The practical usefulness of the algorithms for computing maximum noncrossing graphs will be limited by the relatively low coverage.

The class of graphs with pagenumber no more than two has a considerably satisfactory coverage.

PN	DM	PAS	CCD	PSD
1	69.83%	60.07%	48.23%	78.26%
2	29.85%	39.46%	49.86%	20.12%
3	0.31%	0.46%	1.71%	1.39%
4	0	0.02%	0.18%	0.21%
5	0	0	0.02%	0.02%
6	0	0	0	0.01%

Table 1: Coverage in terms of complete graphs with respect to different pagenumbers ("PN" for short). "DM," "PAS," "CCD" and "PSD" are short for DeepBank, Enju HPSGBank, CCGBank and Prague Dependency Treebank.

It can account for more than 98% of the graphs and sometimes close to 100% in each data set. Unfortunately, the power of Maximum Subgraph parsing is limited given that finding the maximum acyclic subgraph when pagenumber is at most k is NP-hard for $k \geq 2$ (Kuhlmann and Jonsson, 2015). As an alternative, we propose to model a semantic graph as a book, in which the spine is made up of a sequence of words, and each half-plane contains a subset of dependency arcs. To build a semantic graph for a given sentence, we design new parsing algorithms to generate noncrossing graphs on each page (Section 3), and a Lagrangian Relaxation-based algorithm to integrate pages into a book (Section 4).

3 Maximum Subgraph for Noncrossing Graphs

We introduce several DP algorithms for calculating the maximum noncrossing dependency graphs. Each algorithm visits all the spans from bottom to top, finding the best combination of smaller structures to form a new structure, according to the scores of first- or higher-order features. For sake of conciseness, we focus on undirected graphs and treat direction of linguistic dependencies as edge labels[1]. We will use $e_{(i,j,l)}(i < j)$ or simply $e_{(i,j)}$ to indicate an edge in either direction

[1] The *single-head* property does not hold. We currently do not consider other constraints of directions. So prediction of the direction of one edge does not affect prediction of other edges as well as their directions. The directions can be assigned *locally*, and our parser builds directed rather than undirected graphs in this way. Undirected graphs are only used to conveniently illustrate our algorithms. All experimental results in Section 5 consider directed dependencies in a standard way. We use the official evaluation tool provided by SDP2014 shared task. The numberic results reported in this paper are directly comparable to results in other papers.

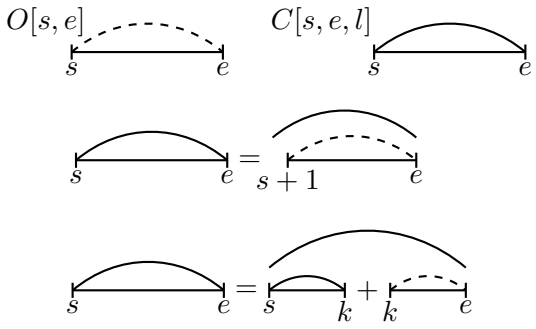


Figure 3: The sub-problems of first-order factorization and the decomposition for $C[s, e, l]$.

between i and j.

For sake of formal concision, we introduce the algorithm of which the goal is to calculate the maximum score of a subgraph. Extracting corresponding optimal graphs can be done in a number of ways. For example, we can maintain an auxiliary arc table which is populated parallel to the procedure of obtaining maximum scores. We define two score functions: (1) $\mathrm{s}_{\mathrm{fst}}(s, e, l)$ assigns a score to an individual edge $e_{(s,e,l)}$ and (2) $\mathrm{s}_{\mathrm{scd}}(s, e_1, e_2, l_1, l_2)$ assigns a score to a pair of neighboring edges $e_{(s,e_1,l_1)}$ and $e_{(s,e_2,l_2)}$.

3.1 First-Order Factorization

Given a sentence, we define two DP tables, namely $O[s, e]$ and $C[s, e, l]$ which represents the value of the highest scoring noncrossing graphs that spans sequences of words of a sentence. The two tables are related to two sub-problems, as graphically shown in Figure 3. The following is their explaination.

Open $O[s, e]$ is intended to represent the highest weighted subgraph spanning w_s to w_e. The subgraphs related to $O[s, e]$ may or may not contain $e_{(s,e)}$.

Closed $C[s, e, l]$ represents the highest weighted subgraph spanning w_s to w_e too. But the subgraphs related to $C[s, e, l]$ must contain $e_{(s,e,l)}$.

$O[s, e]$ can be obtained by one of the following combinations:

- $C[s, e, l](l \in L)$, if there is an edge between s and e with label l.
- $C[s, k, l] + O[k, e](l \in L, s < k < e)$, if $e_{(s,e)}$ does not exist and there is an edge with label l between s and some node in this span. k is the farthest node linked to s.
- $O[s + 1, e]$, if $e_{(s,e)}$ does not exist and there is no edge to its right in this span.

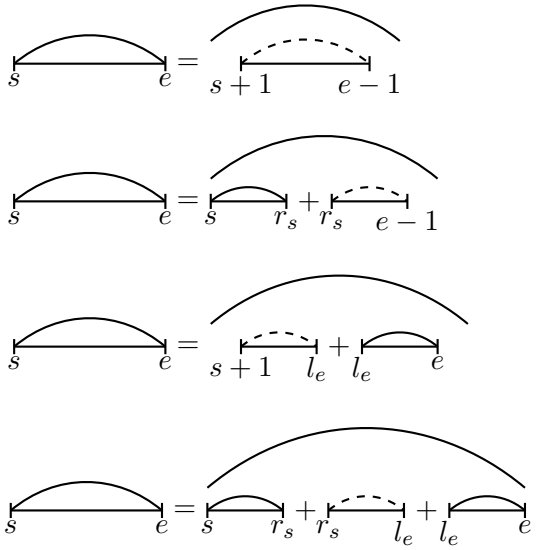


Figure 4: The decomposition for $C[s, e, l]$ in exact single-side second-order factorization.

$C[s, e, l]$ can be obtained by one of the following combinations:

- $O[s + 1, e] + \mathrm{s}_{\mathrm{fst}}(s, e, l)$, if s has no edge to its right;
- $C[s, k, l'] + O[k, e] + \mathrm{s}_{\mathrm{fst}}(s, e, l)(l' \in L, s < k < e)$, if there is an edge from s to some node in the span.

For each edge, there are two directions for the edge, we encode the directions into the label l, and treat it as undirected edge. We need to search for a best split and a best label for every span, so the time complexity of the algorithm is $O(n^3|L|)$ where n is the length of the sentence and L is the set of labels.

3.2 Second-Order Single Side Factorization

We propose a new algorithm concerning single-side second-order factorization. The DP tables, as well as the decomposition for the open problem, are the same as in the first order factorization. The decomposition of $C[s, e, l]$ is very different. In order to score second-order features from adjacent edges in the same side, which is similar to *sibling* features for tree parsing (McDonald and Pereira,

2006), we need to find the rightmost node adjacent to s, denoted as r_s, and the leftmost node adjacent to e, denoted as l_e, and here we have $s < r_s \leq l_e < e$. And, sometimes, we split $C[s, e, l]$ into three parts to capture the neighbor factors on both endpoints. In summary, $C[s, e, l]$ can be obtained by one of the following combination (as graphically shown in Figure 4):

- $O[s + 1, e - 1] + \mathrm{s}_{\mathrm{fst}}(s, e, l) + \mathrm{s}_{\mathrm{scd}}(s, nil, e, nil, l) + \mathrm{s}_{\mathrm{scd}}(e, nil, s, nil, l)$, if there is no edge from s/e to any node in the span.
- $C[s, r_s, l'] + O[r_s, e - 1] + \mathrm{s}_{\mathrm{fst}}(s, e, l) + \mathrm{s}_{\mathrm{scd}}(s, r_s, e, l', l) + \mathrm{s}_{\mathrm{scd}}(e, nil, s, nil, l)$ $(s < r_s < e)$, if there is no edge from e to any node in the span.
- $O[s + 1, l_e] + C[l_e, e, l'] + \mathrm{s}_{\mathrm{fst}}(s, e, l) + \mathrm{s}_{\mathrm{scd}}(e, l_e, s, l', l) + \mathrm{s}_{\mathrm{scd}}(s, nil, e, nil, l)$ $(s < l_e < e)$, if there is no edge from s to any node in the span.
- $C[s, r_s, l'] + O[r_s, l_e] + C[l_e, e, l''] + \mathrm{s}_{\mathrm{fst}}(s, e, l) + \mathrm{s}_{\mathrm{scd}}(s, r_s, e, l', l) + \mathrm{s}_{\mathrm{scd}}(e, l_e, s, l'', l)$ $(s < r_s \leq l_e < e)$, otherwise.

For the last combination, we need to search for two best separating words, namely s_r and l_e, and two best labels, namely l' and l', so the time complexity of this second-order algorithm is $O(n^4|L|^2)$.

3.3 Generalized Higher-Order Parsing

Both of the above two algorithms are exact decoding algorithms. Solutions allow for exact decoding with higher-order features typically at a high cost in terms of efficiency. A trade-off between rich features and exact decoding benefit tree parsing (McDonald and Nivre, 2011). In particular, Zhang and McDonald (2012) proposed a generalized higher-order model that abandons exact search in graph-based parsing in favor of freedom in feature scope. They kept intact Eisner's algorithm for first-order parsing problems, while enhanced the scoring function in an approximate way by introducing higher-order features.

We borrow Zhang and McDonald's idea and develop a generalized parsing model for noncrossing dependency representations. The sub-problems and their decomposition are much like the first-order algorithm. The difference is that we expand

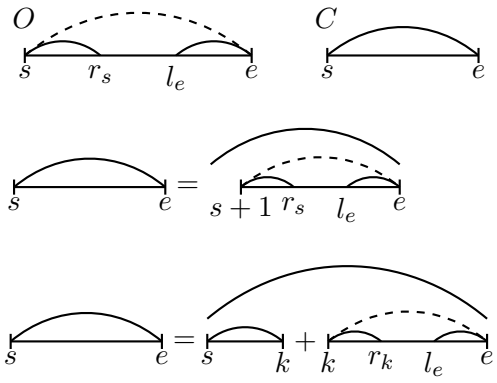


Figure 5: Sub-problems of generalized higher-order factorization and some of the combinations.

the signature of each structure to include all the larger context required to compute higher-order features. For example, we can record the leftmost and the rightmost edges in the open structure to get the tri-neighbor features. The time complexity is thus always $O(n^3B^2)$, no matter how complicatedly higher-order features are incorporated.

We focus on five factors introduced in Section 2.2. Still consider single-side second-order factorization. We keep the closed structure the same but modify the open one to $O[s, e; r_s, l_e, l_{s,r_s}, l_{l_e,e}]$. During parsing, we only record the top-B combinations of label concerning $e_{(s,e)}$ and related r_s, l_e, l_{s,r_s} and $l_{l_e,e}$. The split of a structure is similar to the first-order algorithm, shown in Figure 5. Note that r_s may be e and l_e may be s. In this way, we know exactly whether or not there is an edge from s to e in a refined open structure. This is different from the intuition of the design of the open structure when we consider first-order factorization.

4 Finding and Binding Pages

Statistics presented in Table 1 indicate that the coverage of noncrossing dependency graphs is relatively low. If we treat semantic dependency parsing as Maximum Subgraph parsing, the practical usefulness of the algorithms introduced above is rather limited accordingly. To deal with this problem, we model a semantic graph as a book, and view semantic dependency parsing as finding a book with coherent optimal pages. Given the considerably high coverage of pagenumber at most 2, we only consider 2-page books.

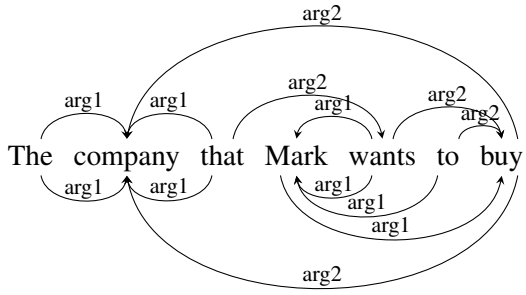


Figure 6: Every non-crossing arc is repeatedly assigned to every page.

4.1 Finding Pages via Coloring

In general, finding the pagenumber of a graph is NP-hard (Gómez-Rodríguez and Nivre, 2010). However, it is easy to figure out that the problem is solvable if the pagenumber is at most 2. Fortunately, a semantic dependency graph is not so dense that it can be usually embedded onto a very *thin* book with only 2 pages. For a structured prediction problem, the structural information of the output produced by a parser is very important. The density of semantic dependency graphs therefore results in a defect: The output's structural information is limited because only a half of arcs on average are included in one page. To enrich the structural information, we put into each page the arcs that do not cross with any other arcs. See Figure 6 for example.

We utilize an algorithm based on coloring to decompose a graph $G = (V, A)$ into two noncrossing subgraphs $G_A = (V, A_B)$ and $G_B = (V, A_B)$. A detailed description is included in the supplementary note. The key idea of our algorithm is to color each crossing arc in two colors using depth-first search. When we color an arc e_x, we examine all arcs crossing with e_x. If one of them, say e_y, has not been examined and can be colored in the other color (no crossing arc of e_y has the same color with e_y), we color e_y and then recursively process e_y. Otherwise, e_y is marked as a bad arc and dropped from both A_A and A_B. After coloring all the crossing arcs, we add every arc in different color to different subgraphs. Specially, all noncrossing arcs are assigned to both A_A and A_B.

4.2 Binding Pages via Lagrangian Relaxation

Applying the above algorithm, we can obtain two corpora to train two noncrossing dependency parsing models. In other words, we can learn two score functions f_A and f_B to score noncrossing dependency graphs. Given the trained models and a sentence, we can find two optimal noncrossing graphs, i.e. find the solutions for $\arg\max_{\boldsymbol{g}} f_A(\boldsymbol{g})$ and $\arg\max_{\boldsymbol{g}} f_B(\boldsymbol{g})$, respectively.

We can put all the arcs contained in $\boldsymbol{g}_A = \arg\max_{\boldsymbol{g}} f_A(\boldsymbol{g})$ and $\boldsymbol{g}_B = \arg\max_{\boldsymbol{g}} f_B(\boldsymbol{g})$ together as our parse for the sentence. This naive combination always gives a graph with a recall much higher than the precision. The problem is that a naive combination does not take the agreements of the graphs on the two pages into consideration, and thus loses some information. To combine the two pages in a principled way, we must do *joint* decoding to find two graphs $\boldsymbol{g}_A$ and $\boldsymbol{g}_B$ to maximize the score $f_A(\boldsymbol{g}_A) + f_B(\boldsymbol{g}_B)$, under the following constraints.

$$\boldsymbol{g}_A(i,j) \leq \sum_{\text{cross}((i,j),(i',j'))} \boldsymbol{g}_B(i',j') + \boldsymbol{g}_B(i,j)$$
$$\boldsymbol{g}_B(i,j) \leq \sum_{\text{cross}((i,j),(i',j'))} \boldsymbol{g}_A(i',j') + \boldsymbol{g}_A(i,j)$$
$$\forall i, j$$

The functionality of *cross* is to figure out whether $e_{(i,j)}$ and $e_{(i',j')}$ cross. The meaning of the first constraint is: When there is an arc $e_{(i,j)}$ in the first graph, $e_{(i,j)}$ is also in the second graph, or there is an arc $e_{(i',j')}$ in the second graph which cross with $e_{(i,j)}$. So is the second one. All constraints are linear and can be written in a simplified way as,

$$A\boldsymbol{g}_A + B\boldsymbol{g}_B \leq \mathbf{0}$$

where A and B are matrices that can be constructed by checking all possible crossing arc pairs. In summary, we have the following constrained optimization problem,

$$\begin{array}{ll} \text{min.} & -f_A(\boldsymbol{g}_A) - f_B(\boldsymbol{g}_B) \\ \text{s.t.} & \boldsymbol{g}_A, \boldsymbol{g}_B \text{ are noncrossing graphs} \\ & A\boldsymbol{g}_A + B\boldsymbol{g}_B \leq \mathbf{0} \end{array}$$

The Lagrangian of the optimization problem is

$$\begin{aligned} &\mathcal{L}(\boldsymbol{g}_A, \boldsymbol{g}_B; u) \\ &\quad = -f_g(\boldsymbol{g}_A) - f_t(\boldsymbol{g}_B) + u^\top (A\boldsymbol{g}_A + B\boldsymbol{g}_B) \end{aligned}$$

where u is the Lagrangian multiplier. Then the dual is

$$\begin{aligned} \mathcal{L}(u) &= \min_{\boldsymbol{g}_A, \boldsymbol{g}_B} \mathcal{L}(\boldsymbol{g}_A, \boldsymbol{g}_B; u) \\ &= \max_{\boldsymbol{g}_A} (f_g(\boldsymbol{g}_A) - u^\top A\boldsymbol{g}_A) \\ &\quad + \max_{\boldsymbol{g}_B} (f_y(\boldsymbol{g}_B) - u^\top B\boldsymbol{g}_B) \end{aligned}$$

BINDTWOPAGES($\boldsymbol{g}_A, \boldsymbol{g}_B$)
1 $u^{(0)} \leftarrow 0$
2 **for** $k \leftarrow 0..T$ **do**
3 $\quad \boldsymbol{g}_A \leftarrow \arg\max_{\boldsymbol{g}} f_A(\boldsymbol{g}) - u^{(k)\top} A\boldsymbol{g}$
4 $\quad \boldsymbol{g}_B \leftarrow \arg\max_{\boldsymbol{g}} f_B(\boldsymbol{g}) - u^{(k)\top} B\boldsymbol{g}$
5 $\quad$ **if** $A\boldsymbol{g}_A + B\boldsymbol{g}_B \leq \mathbf{0}$ **then**
6 $\quad\quad$ **return** $\boldsymbol{g}_A, \boldsymbol{g}_B$
7 $\quad$ **else**
8 $\quad\quad u^{(k+1)} \leftarrow u^{(k)} + \alpha^{(k)}(A\boldsymbol{g}_A + B\boldsymbol{g}_B)$
9 **return** $\boldsymbol{g}_A, \boldsymbol{g}_B$

Figure 7: The page binding algorithm.

We instead try to find the solution for $\max_u \mathcal{L}(u)$. By using a subgradient method to calculate $\max_u \mathcal{L}(u)$, we have an algorithm for joint decoding (see Figure 7). $\mathcal{L}(u)$ is divided into two optimization problems which can be decoded easily. Each sub-problem is still a parsing problem for noncrossing graphs. Only the scores of factors are modified (see Line 3 and 4). Specifically, to modify the first order weights of edges, we take a subtraction of $u^\top A$ in the first model and a substraction of $u^\top B$ in the second one. In each iteration, after obtaining two new parsing results, we check whether the constraints are satisfied. If the answer is "yes," we stop and return the merged graph. Otherwise, we update u in a way to increase $\mathcal{L}(u)$ (see Line 8).

5 Experiments

5.1 Data Sets

To evaluate the effectiveness of book embedding in practice, we conduct experiments on unlabeled parsing using four corpora: CCGBank (Hockenmaier and Steedman, 2007), DeepBank (Flickinger et al., 2012), Enju HPSGBank (EnjuBank; Miyao et al., 2004) and Prague Dependency TreeBank (PCEDT; Hajic et al., 2012), We use "standard" training, validation, and test splits to facilitate comparisons. Following previous experimental setup for CCG parsing, we use section 02-21 as training data, section 00 as the development data, and section 23 for testing. The other three data sets are from SemEval 2014 Task 8 (Oepen et al., 2014), and the data splitting policy follows the shared task. All the four data sets are publicly available from LDC (Oepen et al., 2016).

Experiments for CCG analysis were performed using automatically assigned POS-tags generated by a symbol-refined HMM tagger (Huang et al., 2010). For the other three data sets we use POS-tags provided by the shared task. We also use features extracted from trees. We consider two types of trees: (1) syntactic trees provided as a companion analysis by the shared task and CCGBank, (2) pseudo trees (Zhang et al., 2016) automatically extracted from semantic dependency annotations. We utilize the Mate parser (Bohnet, 2010) to generate pseudo trees for all data sets and also syntactic trees for CCG analysis, and use the companion syntactic analysis provided by the shared task for the other three data sets.

5.2 Statistical Disambiguation

Our parsing algorithms can be applied to scores originated from any source, but in our experiments we chose to use the framework of global linear models, deriving our scores as:

$$\text{SCOREPART}(s, p) = \mathbf{w}^\top \phi(s, p)$$

ϕ is a feature-vector mapping and $\mathbf{w}$ is a parameter vector. p may refer to a single arc, a pair of neighboring arcs, or a general tuple of arcs, according to the definition of a parsing model. For details we refer to the source code. We chose the averaged structured perceptron (Collins, 2002) for parameter estimation.

5.3 Results of Practical Parsing

We evaluate five decoding algorithms:

M1 first-order exact algorithm,

M2 second-order exact algorithm with single-side factorization,

M3 second-order approximate algorithm[2] with single-side factorization,

M4 second-order approximate algorithm with single- and both-side factorization,

M5 third-order approximate algorithm with single- and both-side factorization.

5.3.1 Effectiveness of Higher-Order Features

Table 2 lists the accuracy of Maximum Subgraph parsing. The output of our parser was evaluated against each dependency in the corpus. We report unlabeled precision (UP), recall (UR) and f-score (UF). We can see that the first-order model obtains a considerably good precision, with rich features.

[2]The beam size is set to 4 for all approximate algorithms.

			DeepBank			EnjuBank			CCGBank			PCEDT		
			UP	UR	UF	UP	UR	UF	UP	UR	UF	UP	UR	UF
Syntax Tree	M1	MS	90.97	86.11	88.47	92.92	89.71	91.29	94.21	88.70	91.37	91.49	86.39	88.87
	M2		91.04	87.47	89.22	93.03	90.48	91.74	93.95	88.96	91.39	91.11	87.56	89.30
	M3		90.94	87.65	89.27	93.27	90.62	91.93	93.93	89.11	91.46	91.25	87.66	89.42
	M4		91.02	87.78	89.37	93.18	90.65	91.90	94.02	89.14	91.51	91.43	87.98	89.67
	M5		90.91	87.51	89.18	93.15	90.57	91.84	93.91	89.19	91.49	91.29	87.96	89.59
	M4	NC	88.17	90.46	89.30	91.42	93.42	92.41	92.36	93.10	92.73	89.25	90.34	89.79
		LR	90.72	88.80	89.75	92.75	92.49	92.62	93.50	92.48	92.98	90.98	89.04	90.00
Pseudo Tree	M1	MS	90.75	86.13	88.38	93.38	90.20	91.76	94.21	88.55	91.29	90.62	85.69	88.08
	M2		90.13	87.01	88.54	93.18	90.63	91.89	93.96	88.54	91.17	89.92	86.55	88.20
	M3		90.39	87.20	88.77	93.20	90.64	91.90	93.90	88.98	91.37	90.07	86.69	88.35
	M4		90.31	87.25	88.76	93.18	90.67	91.91	94.01	89.04	91.46	90.03	86.84	88.40
	M5		90.17	87.11	88.61	93.13	90.62	91.86	93.87	89.00	91.37	90.21	86.93	88.54
	M4	NC	88.39	89.85	89.11	91.63	93.24	92.43	92.83	92.97	92.90	88.51	88.97	88.74
		LR	90.01	88.55	89.27	92.79	92.59	92.69	93.78	92.28	93.02	90.04	87.92	88.97

Table 2: Parsing accuracy evaluated on the development sets. "MS" is short for Maximum Subgraph parsing. "NC" and "LR" are short for naive combination and Lagrangian Relaxation.

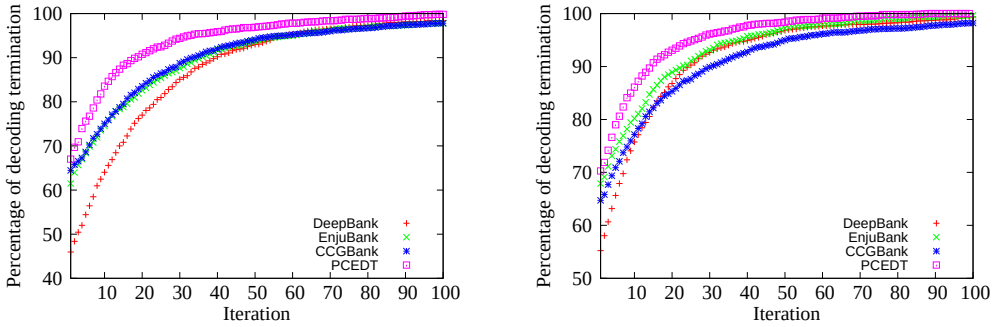


Figure 8: The termination rate of page binding. The left and right diagrams show the results obtained when applying syntactic and pseudo tree features respectively.

But due to the low coverage of the noncrossing dependency graphs, a set of dependencies can not be built. This property has a great impact on recall. Furthermore, we can see that the introduction of higher-order features improves parsing substantially for all data sets, as expected. When pseudo trees are utilized, the improvement is marginal. We think the reason is that we have already included many higher-order features at the stage of pseudo tree parsing.

5.3.2 Effectiveness of Approximate Parsing

Perhaps surprisingly approximate parsing with single-side second order features and cube pruning is even slightly better than exact parsing. This result demonstrates the effectiveness of generalized dependency parsing. Further including third-order features does not improve parsing accuracy.

5.3.3 Effectiveness of Page Binding

When arcs are assigned to two sets, we can separately train two parsers for producing two types of noncrossing dependency graphs. These two parsers can be integrated using a naive merger or a LR-based merger. Table 2 also shows the accuracy obtained by the second-order model M4. The effectivenss of the Lagrangian Relaxation-based algorithm for binding pages is confirmed.

5.3.4 Termination Rate of Page Binding

Figure 8 presents the termination rate with respective to the number of iterations. Here we apply M4 with syntax and pseudo tree features. In practice the Lagrangian Relaxation-based algorithm finds solutions in a few iterations for a majority of sentences. This suggests that even though the joint decoding is an iterative procedure, satisfactory efficiency is still available.

		DeepBank			EnjuBank			CCGBank			PCEDT		
		UP	UR	UF	UP	UR	UF	UP	UR	UF	UP	UR	UF
M4-LR	Syn	89.99	87.77	88.87	92.87	92.04	92.46	93.45	92.51	92.98	89.58	87.73	88.65
	Pse	90.01	88.16	89.08	93.17	92.48	92.83	93.66	92.06	92.85	89.27	87.37	88.31
ZDSW	Pse	89.04	88.85	88.95	92.92	92.83	92.87	92.49	92.30	92.40	- -	- -	- -
Peking		91.72	89.92	90.81	94.46	91.61	93.02	- -	- -	- -	91.79	86.02	88.81

Table 3: Parsing accuracy evaluated on the test sets.

5.4 Comparison with Other Parsers

We show the parsing results on the test data together with some relevant results from related work. We compare our parser with two other systems: (1) ZDSW (Zhang et al., 2016) is a transition-based system that obtains state-of-the-art accuracy; we present the results of their best single parsing model; (2) Peking (Du et al., 2014) is the best-performing system in the shared task; it is a hybrid system that integrate more than ten submodels to achieve high accuracy. Our parser can be taken as a graph-based parser. It reaches state-of-the-art performance produced by the transition-based system. On DeepBank and EnjuBank, the accuracy of our parser is equivalent to ZDSW, while on CCGBank, our parser is significantly better.

There is still a gap between our single parsing model and Peking hybrid model. For a majority of NLP tasks, e.g. parsing (Surdeanu and Manning, 2010), semantic role labeling (Koomen et al., 2005), hybrid systems that combines complementary strength of heterogeneous models perform better. But good individual system is the cornerstone of hybrid systems. Better design of single system almost always benefits system ensemble.

6 Conclusion

We propose a new data-driven parsing framework, namely book embedding, for semantic dependency analysis, viz. mapping from natural language sentences to bilexical semantic dependency graphs. Our work includes two contributions:

1. new algorithms for maximum noncrossing dependency parsing.

2. a Lagrangian Relaxation based algorithm to combine noncrossing dependency subgraphs.

Experiments demonstrate the effectiveness of the book embedding framework across a wide range of conditions. Our graph-based parser obtains state-of-the-art accuracy.

Acknowledgments

This work was supported by 863 Program of China (2015AA015403), NSFC (61331011), and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). Xiaojun Wan is the corresponding author.

References

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, Beijing, China, pages 89–97. http://www.aclweb.org/anthology/C10-1011.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017. Parsing to 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1–8. https://doi.org/10.3115/1118693.1118694.

Yantao Du, Weiwei Sun, and Xiaojun Wan. 2015. A data-driven, factorization parser for CCG dependency structures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1545–1555. http://www.aclweb.org/anthology/P15-1149.

Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 459–464. http://www.aclweb.org/anthology/S14-2080.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 340–345.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*. pages 85–96.

Carlos Gómez-Rodríguez and Joakim Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1492–1501. http://www.aclweb.org/anthology/P10-1151.

Jan Hajic, Eva Hajicová, Jarmila Panevová, Petr Sgall, Ondej Bojar, Silvie Cinková, Eva Fucíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jirí Semecký, Jana Sindlerová, Jan Stepánek, Josef Toman, Zdenka Uresová, and Zdenek Zabokrtský. 2012. Announcing prague czech-english dependency treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*. Istanbul, Turkey.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics* 33(3):355–396.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 12–22. http://www.aclweb.org/anthology/D10-1002.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*. Jeju, Republic of Korea, pages 2–11.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 181–184.

Sandra Kübler, Ryan T. McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics* 3:559–570.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006))*. volume 6, pages 81–88.

Ryan T. McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics* 37(1):197–230.

Yusuke Miyao, Takashi Ninomiya, and Jun ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. In *IJCNLP*. pages 684–693.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Zdeňka Urešová. 2016. Semantic Dependency Parsing (SDP) graph banks release 1.0 LDC2016T10. Web Download.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresová. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 63–72. http://www.aclweb.org/anthology/S14-2008.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 649–652. http://www.aclweb.org/anthology/N10-1091.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the 21st international jont conference on Artifical intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pages 1562–1567. http://dl.acm.org/citation.cfm?id=1661445.1661696.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational

Linguistics, Jeju Island, Korea, pages 320–331. http://www.aclweb.org/anthology/D12-1030.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389. http://aclweb.org/anthology/J16-3001.

Neural Word Segmentation with Rich Pretraining

Jie Yang* and **Yue Zhang*** and **Fei Dong**
Singapore University of Technology and Design
{jie_yang, fei_dong}@mymail.sutd.edu.sg
yue_zhang@sutd.edu.sg

Abstract

Neural word segmentation research has benefited from large-scale raw texts by leveraging them for pretraining character and word embeddings. On the other hand, statistical segmentation research has exploited richer sources of external information, such as punctuation, automatic segmentation and POS. We investigate the effectiveness of a range of external training sources for neural word segmentation by building a modular segmentation model, pretraining the most important submodule using rich external sources. Results show that such pretraining significantly improves the model, leading to accuracies competitive to the best methods on six benchmarks.

1 Introduction

There has been a recent shift of research attention in the word segmentation literature from statistical methods to deep learning (Zheng et al., 2013; Pei et al., 2014; Morita et al., 2015; Chen et al., 2015b; Cai and Zhao, 2016; Zhang et al., 2016b). Neural network models have been exploited due to their strength in *non-sparse representation learning* and *non-linear power* in feature combination, which have led to advances in many NLP tasks. So far, neural word segmentors have given comparable accuracies to the best statictical models.

With respect to *non-sparse representation*, character embeddings have been exploited as a foundation of neural word segmentors. They serve to reduce sparsity of character ngrams, allowing, for example, "猫(cat) 躺(lie) 在(in) 墙角(corner)" to be connected with "狗(dog) 蹲(sit) 在(in) 墙角(corner)" (Zheng et al., 2013), which is infeasible by using sparse one-hot character features. In addition to character embeddings, distributed representations of character bigrams (Mansur et al., 2013; Pei et al., 2014) and words (Morita et al., 2015; Zhang et al., 2016b) have also been shown to improve segmentation accuracies.

With respect to *non-linear modeling power*, various network structures have been exploited to represent contexts for segmentation disambiguation, including multi-layer perceptrons on five-character windows (Zheng et al., 2013; Mansur et al., 2013; Pei et al., 2014; Chen et al., 2015a), as well as LSTMs on characters (Chen et al., 2015b; Xu and Sun, 2016) and words (Morita et al., 2015; Cai and Zhao, 2016; Zhang et al., 2016b). For structured learning and inference, CRF has been used for character sequence labelling models (Pei et al., 2014; Chen et al., 2015b) and structural beam search has been used for word-based segmentors (Cai and Zhao, 2016; Zhang et al., 2016b).

Previous research has shown that segmentation accuracies can be improved by pretraining character and word embeddings over large Chinese texts, which is consistent with findings on other NLP tasks, such as parsing (Andor et al., 2016). Pretraining can be regarded as one way of leveraging external resources to improve accuracies, which is practically highly useful and has become a standard practice in neural NLP. On the other hand, statistical segmentation research has exploited raw texts for semi-supervised learning, by collecting clues from raw texts more thoroughly such as mutual information and punctuation (Li and Sun, 2009; Sun and Xu, 2011), and making use of self-predictions (Wang et al., 2011; Liu and Zhang, 2012). It has also utilised heterogenous annotations such as POS (Ng and Low, 2004; Zhang and Clark, 2008) and segmentation under different

* Equal contribution.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 839–849
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1078

State	Recognized words	Partial word	Incoming chars	Next Action
state0	[ ]	ϕ	[我去过火车站那边]	SEP
state1	[ ]	我	[去过火车站那边]	SEP
state2	[我]	去	[过火车站那边]	SEP
state3	[我,去]	过	[火车站那边]	SEP
state4	[我,去,过]	火	[车站那边]	APP
state5	[我,去,过]	火车	[站那边]	APP
state6	[我,去,过]	火车站	[那边]	SEP
state7	[我,去,过,火车站]	那	[边]	APP
state8	[我,去,过,火车站]	那边	[ ]	FIN
state9	[我,去,过,火车站,那边]	ϕ	[ ]	- -

Table 1: A transition based word segmentation example.

standards (Jiang et al., 2009). To our knowledge, such rich external information has not been systematically investigated for neural segmentation.

We fill this gap by investigating rich external pretraining for neural segmentation. Following Cai and Zhao (2016) and Zhang et al. (2016b), we adopt a globally optimised beam-search framework for neural structured prediction (Andor et al., 2016; Zhou et al., 2015; Wiseman and Rush, 2016), which allows word information to be modelled explicitly. Different from previous work, we make our model conceptually simple and modular, so that the most important sub module, namely a five-character window context, can be pretrained using external data. We adopt a multi-task learning strategy (Collobert et al., 2011), casting each external source of information as a auxiliary classification task, sharing a five-character window network. After pretraining, the character window network is used to initialize the corresponding module in our segmentor.

Results on 6 different benchmarks show that our method outperforms the best statistical and neural segmentation models consistently, giving the best reported results on 5 datasets in different domains and genres. Our implementation is based on *LibN3L*[1] (Zhang et al., 2016a). Code and models can be downloaded from `http://gitHub.com/jiesutd/RichWordSegmentor`

2 Related Work

Work on *statistical* word segmentation dates back to the 1990s (Sproat et al., 1996). State-of-the-art approaches include *character* sequence labeling models (Xue et al., 2003) using CRFs (Peng et al., 2004; Zhao et al., 2006) and max-margin structured models leveraging *word* features (Zhang and Clark, 2007; Sun et al., 2009; Sun, 2010). Semi-supervised methods have been applied to both character-based and word-based models, exploring external training data for better segmentation (Sun and Xu, 2011; Wang et al., 2011; Liu and Zhang, 2012; Zhang et al., 2013). Our work belongs to recent *neural* word segmentation.

To our knowledge, there has been no work in the literature systematically investigating rich external resources for neural word segmentation training. Closest in spirit to our work, Sun and Xu (2011) empirically studied the use of various external resources for enhancing a statistical segmentor, including character mutual information, access variety information, punctuation and other statistical information. Their baseline is similar to ours in the sense that both character and word contexts are considered. On the other hand, their model is statistical while ours is neural. Consequently, they integrate external knowledge as features, while we integrate it by shared network parameters. Our results show a similar degree of error reduction compared to theirs by using external data.

Our model inherits from previous findings on context representations, such as character windows (Mansur et al., 2013; Pei et al., 2014; Chen et al., 2015a) and LSTMs (Chen et al., 2015b; Xu and Sun, 2016). Similar to Zhang et al. (2016b) and Cai and Zhao (2016), we use word context on top of character context. However, words play a relatively less important role in our model, and we find that word LSTM, which has been used by all previous neural segmentation work, is unnecessary for our model. Our model is conceptually simpler and more modularised compared with

[1] `https://github.com/SUTDNLP/LibN3L`

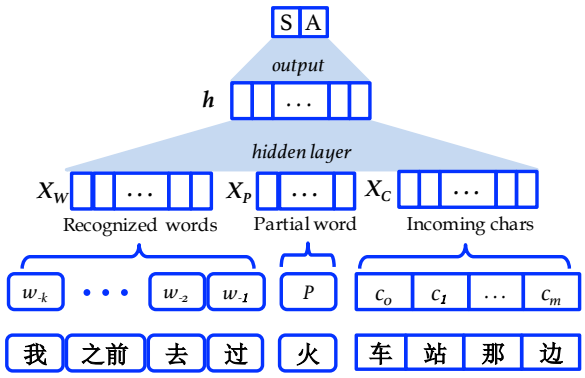


Figure 1: Overall model.

Zhang et al. (2016b) and Cai and Zhao (2016), allowing a central sub module, namely a five-character context window, to be pretrained.

3 Model

Our segmentor works incrementally from left to right, as the example shown in Table 1. At each step, the state consists of a sequence of words that have been fully recognized, denoted as $W = [w_{-k}, w_{-k+1}, ..., w_{-1}]$, a current partially recognized word P, and a sequence of next incoming characters, denoted as $C = [c_0, c_1, ..., c_m]$, as shown in Figure 1. Given an input sentence, W and P are initialized to $[\,]$ and ϕ, respectively, and C contains all the input characters. At each step, a decision is made on c_0, either appending it as a part of P, or seperating it as the beginning of a new word. The incremental process repeats until C is empty and P is null again ($C = [\,]$, $P = \phi$). Formally, the process can be regarded as a state-transition process, where a state is a tuple $S = \langle W, P, C\rangle$, and the transition actions include SEP (*seperate*) and APP (*append*), as shown by the deduction system in Figure 2[2].

In the figure, V denotes the score of a state, given by a neural network model. The score of the initial state (i.e. axiom) is 0, and the score of a non-axiom state is the sum of scores of all incremental decisions resulting in the state. Similar to Zhang et al. (2016b) and Cai and Zhao (2016), our model is a global structural model, using the overall score to disambiguate states, which correspond to sequences of inter-dependent transition actions.

Different from previous work, the structure of

$$\text{Axiom:}\quad S = \langle [\,], \phi, C\rangle,\ V = 0$$
$$\text{Goal:}\quad S = \langle W, \phi, [\,]\rangle,\ V = V_{final}$$

$$\text{SEP:}\quad \frac{S = \langle W, P, c_0|C\rangle,\ V}{S' = \langle W|P, c_0, C\rangle,\ V' = V + Score(S, \text{SEP})}$$

$$\text{APP:}\quad \frac{S = \langle W, P, c_0|C\rangle,\ V}{S' = \langle W, P \oplus c_0, C\rangle,\ V' = V + Score(S, \text{APP})}$$

Figure 2: Deduction system, where $\oplus$ denotes string concatenation.

our scoring network is shown in Figure 1. It consists of three main layers. On the bottom is a *representation layer*, which derives dense representations X_W, X_P and X_C for W, P and C, respectively. We compare various distributed representations and neural network structures for learning X_W, X_P and X_C, detailed in Section 3.1. On top of the representation layer, we use a *hidden layer* to merge X_W, X_P and X_C into a single vector

$$h = tanh(W_{hW} \cdot X_W + W_{hP} \cdot X_P + W_{hC} \cdot X_C + b_h) \quad (1)$$

The hidden feature vector h is used to represent the state $S = \langle W, P, C\rangle$, for calculating the scores of the next action. In particular, a linear *output layer* with two nodes is employed:

$$o = W_o \cdot h + b_o \quad (2)$$

The first and second node of o represent the scores of SEP and APP given S, namely $Score(S, \text{SEP})$, $Score(S, \text{APP})$ respectively.

3.1 Representation Learning

Characters. We investigate two different approaches to encode incoming characters, namely a *window approach* and an *LSTM approach*. For the former, we follow prior methods (Xue et al., 2003; Pei et al., 2014), using five-character window $[c_{-2}, c_{-1}, c_0, c_1, c_2]$ to represent incoming characters. Shown in Figure 3, a multi-layer perceptron (MLP) is employed to derive a five-character window vector D_C from single-character vector representations $V_{c_{-2}}, V_{c_{-1}}, V_{c_0}, V_{c_1}, V_{c_2}$.

$$D_C = MLP([V_{c_{-2}}; V_{c_{-1}}; V_{c_0}; V_{c_1}; V_{c_2}]) \quad (3)$$

For the latter, we follow recent work (Chen et al., 2015b; Zhang et al., 2016b), using a bi-directional LSTM to encode input character sequence.[3] In particular, the bi-directional LSTM

[2] An end of sentence symbol $\langle/s\rangle$ is added to the input so that the last partial word can be put onto W as a full word before segmentation finishes.

[3] The LSTM variation with coupled input and forget gate but without peephole connections is applied (Gers and Schmidhuber, 2000)

hidden vector $[\overleftarrow{h_C}(c_0); \overrightarrow{h_C}(c_0)]$ of the next incoming character c_0 is used to represent the coming characters $[c_0, c_1, ...]$ given a state. Intuitively, a five-character window provides a local context from which the meaning of the middle character can be better disambiguated. LSTM, on the other hand, captures larger contexts, which can contain more useful clues for dismbiguation but also irrelevant information. It is therefore interesting to investigate a combination of their strengths, by first deriving a locally-disambiguated version of c_0, and then feed it to LSTM for a globally disambiguated representation.

Now with regard to the single-character vector representation $V_{c_i} (i \in [-2, 2])$, we follow previous work and consider both character embedding $e^c(c_i)$ and character-bigram embedding $e^b(c_i, c_{i+1})$, investigating the effect of each on the accuracies. When both $e^c(c_i)$ and $e^b(c_i, c_{i+1})$ are utilized, the concatenated vector is taken as V_{c_i}.

Partial Word. We take a very simple approach to representing the partial word P, by using the embedding vectors of its first and last characters, as well as the embedding of its length. Length embeddings are randomly initialized and then tuned in model training. X_P has relatively less influence on the empirical segmentation accuracies.

$$X_P = [e^c(P[0]); e^c(P[-1]); e^l(\text{LEN}(P))] \quad (4)$$

Word. Similar to the character case, we investigate two different approaches to encoding incoming characters, namely a *window approach* and an *LSTM approach*. For the former, we follow prior methods (Zhang and Clark, 2007; Sun, 2010), using the two-word window $[w_{-2}, w_{-1}]$ to represent recognized words. A hidden layer is employed to derive a two-word vector X_W from single word embeddings $e^w(w_{-2})$ and $e^w(w_{-1})$.

$$X_W = tanh(W_w[e^w(w_{-2}); e^w(w_{-1})] + b_w) \quad (5)$$

For the latter, we follow Zhang et al. (2016b) and Cai and Zhao (2016), using an uni-directional LSTM on words that have been recognized.

3.2 Pretraining

Neural network models for NLP benefit from pretraining of word/character embeddings, learning distributed sementic information from large raw texts for reducing sparsity. The three basic elements in our neural segmentor, namely characters, character bigrams and words, can all be pretrained

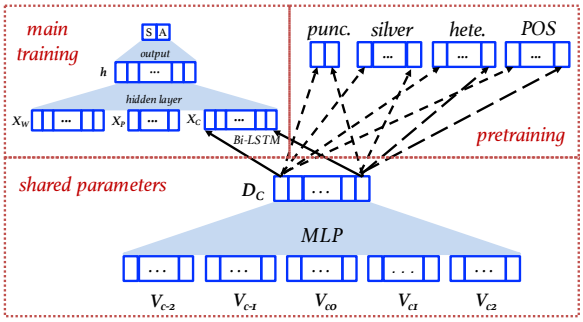


Figure 3: Shared character representation.

over large unsegmented data. We pretrain the five-character window network in Figure 3 as an unit, learning the MLP parameter together with character and bigram embeddings. We consider four types of commonly explored external data to this end, all of which have been studied for statistical word segmentation, but not for neural network segmentors.

Raw Text. Although raw texts do not contain explicit word boundary information, statistics such as mutual information between consecutive characters can be useful features for guiding segmentation (Sun and Xu, 2011). For neural segmentation, these distributional statistics can be implicitly learned by pretraining character embeddings. We therefore consider a more explicit clue for pretraining our character window network, namely punctuations (Li and Sun, 2009).

Punctuation can serve as a type of explicit markup (Spitkovsky et al., 2010), indicating that the two characters on its left and right belong to two different words. We leverage this source of information by extracting character five-grams excluding punctuation from raw sentences, using them as inputs to classify whether there is punctuation before middle character. Denoting the resulting five character window as $[c_{-2}, c_{-1}, c_0, c_1, c_2]$, the MLP in Figure 3 is used to derive its representation D_C, which is then fed to a softmax layer for binary classification:

$$P(punc) = softmax(W_{punc} \cdot D_C + b_{punc}) \quad (6)$$

Here $P(punc)$ indicates the probability of a punctuation mark existing before c_0. Standard back-propagation training of the MLP in Figure 3 can be done jointly with the training of W_{punc} and b_{punc}. After such training, the embedding V_{ci} and MLP values can be used to initialize the corresponding parameters for D_C in the main segmentor, before

its training.

Automatically Segmented Text. Large texts automatically segmented by a baseline segmentor can be used for self-training (Liu and Zhang, 2012) or deriving statistical features (Wang et al., 2011). We adopt a simple strategy, taking automatically segmented text as silver data to pretrain the five-character window network. Given $[c_{-2}, c_{-1}, c_0, c_1.c_2]$, D_C is derived using the MLP in Figure 3, and then used to classify the segmentation of c_0 into B(begining)/M(middle)/E(end)/S(single character word) labels.

$$P(silver) = softmax(W_{silv} \cdot D_C + b_{silv}) \quad (7)$$

Here W_{silv} and b_{silv} are model parameters. Training can be done in the same way as training with punctuation.

Heterogenous Training Data. Multiple segmentation corpora exist for Chinese, with different segmentation granularities. There has been investigation on leveraging two corpora under different annotation standards to improve statistical segmentation (Jiang et al., 2009). We try to utilize heterogenous treebanks by taking an external treebank as labeled data, training a B/M/E/S classifier for the character windows network.

$$P(hete) = softmax(W_{hete} \cdot D_C + b_{hete}) \quad (8)$$

POS Data. Previous research has shown that POS information is closely related to segmentation (Ng and Low, 2004; Zhang and Clark, 2008). We verify the utility of POS information for our segmentor by pretraining a classifier that predicts the POS on each character, according to the character window representation D_C. In particular, given $[c_{-2}, c_{-1}, c_0, c_1, c_2]$, the POS of the word that c_0 belongs to is used as the output.

$$P(pos) = softmax(W_{pos} \cdot D_C + b_{pos}) \quad (9)$$

Multitask Learning. While each type of external training data can offer one source of segmentation information, different external data can be complimentary to each other. We aim to inject all sources of information into the character window representation D_C by using it as a shared representation for different classification tasks. Neural model have been shown capable of doing multi-task learning via parameter sharing (Collobert et al., 2011). Shown in Figure 3, in our case, the output layer for each task is independent, but the hidden layer D_C and all layers below D_C are shared.

For training with all sources above, we randomly sample sentences from the Punc./Auto-seg/Heter./POS sources with the ratio of 10/1/1/1, for each sentence in punctuation corpus we take only 2 characters (character before and after the punctuation) as input instances.

Algorithm 1: Training

Input : (x^i, y^i)
Parameters: Θ
Process:
agenda $\leftarrow (S = \langle [\,], \phi, X^i \rangle, V = 0)$
for *j* ***in*** *[0:*LEN*(X^i)]* **do**
  beam = []
  for $\hat{y}$ ***in*** *agenda* **do**
    $\hat{y}'$ = ACTION($\hat{y}$, SEP)
    ADD($\hat{y}'$, *beam*)
    $\hat{y}'$ = ACTION($\hat{y}$, APP)
    ADD($\hat{y}'$, *beam*)
  end
  agenda $\leftarrow$ TOP(*beam*, *B*)
  if $y^i_j \notin$ *agenda* **then**
    $\hat{y}_j$ = BESTIN(*agenda*)
    UPDATE(y^i_j, $\hat{y}_j$,Θ)
    return
  end
end
$\hat{y}$ = BESTIN(*agenda*)
UPDATE(y^i, $\hat{y}$,Θ)
return

4 Decoding and Training

To train the main segmentor, we adopt the global transition-based learning and beam-search strategy of Zhang and Clark (2011). For decoding, standard beam search is used, where the *B* best partial output hypotheses at each step are maintained in an *agenda*. Initially, the agenda contains only the start state. At each step, all hypotheses in the agenda are expanded, by applying all possible actions and *B* highest scored resulting hypotheses are used as the agenda for the next step.

For training, the same decoding process is applied to each training example (x^i, y^i). At step j, if the gold-standard sequence of transition actions y^i_j falls out of the agenda, max-margin update is performed by taking the current best hypothesis $\hat{y}_j$ in the beam as a negative example, and y^i_j as

Paramater	Value	Paramater	Value
α	0.01	$size(e^c)$	50
λ	10^{-8}	$size(e^b)$	50
p	0.2	$size(e^w)$	50
η	0.2	$size(e^l)$	20
MLP layer	2	$size(X_C)$	150
beam B	8	$size(X_P)$	50
$size(h)$	200	$size(X_W)$	100

Table 2: Hyper-parameter values.

a positive example. The loss function is

$$l(\hat{y}_j, y^i_j) = max((score(\hat{y}_j) + \eta \cdot \delta(\hat{y}_j, y^i_j) - score(y^i_j)), 0), \quad (10)$$

where $\delta(\hat{y}_j, y^i_j)$ is the number of incorrect local decisions in $\hat{y}_j$, and η controls the score margin.

The strategy above is *early-update* (Collins and Roark, 2004). On the other hand, if the gold-standard hypothesis does not fall out of the agenda until the full sentence has been segmented, a final update is made between the highest scored hypothesis $\hat{y}$ (non-gold standard) in the agenda and the gold-standard y^i, using exactly the same loss function. Pseudocode for the online learning algorithm is shown in Algorithm 1.

We use *Adagrad* (Duchi et al., 2011) to optimize model parameters, with an initial learning rate α. $L2$ regularization and dropout (Srivastava et al., 2014) on input are used to reduce overfitting, with a $L2$ weight λ and a dropout rate p. All the parameters in our model are randomly initialized to a value $(-r, r)$, where $r = \sqrt{\frac{6.0}{fan_{in}+fan_{out}}}$ (Bengio, 2012). We fine-tune character and character bigram embeddings, but not word embeddings, according to Zhang et al. (2016b).

5 Experiments

5.1 Experimental Settings

Data. We use Chinese Treebank 6.0 (CTB6) (Xue et al., 2005) as our main dataset. Training, development and test set splits follow previous work (Zhang et al., 2014). In order to verify the robustness of our model, we additionally use SIGHAN 2005 bake-off (Emerson, 2005) and NLPCC 2016 shared task for Weibo segmentation (Qiu et al., 2016) as test datasets, where the standard splits are used. For pretraining embedding of words, characters and character bigrams, we use Chinese Gigaword (simplified Chinese sections)[4], automatically segmented using ZPar 0.6 off-the-shelf (Zhang and Clark, 2007), the statictics of which are shown in Table 3.

	Source	#Chars	#Words	#Sents
Raw data	Gigaword	116.5m	–	–
Auto seg	Gigaword	398.2m	238.6m	12.04m
Hete.	People's Daily	10.14m	6.17m	104k
POS	People's Daily	10.14m	6.17m	104k

Table 3: Statistics of external data.

For pretraining character representations, we extract punctuation classification data from the Gigaword corpus, and use the word-based ZPar and a standard character-based CRF model (Tseng et al., 2005) to obtain automatic segmentation results. We compare pretraining using ZPar results only and using results that both segmentors agree on. For heterogenous segmentation corpus and POS data, we use a People's Daily corpus of 5 months[5]. Statistics are listed in Table 3.

Evaluation. The standard word precision, recall and F1 measure (Emerson, 2005) are used to evaluate segmentation performances.

Hyper-parameter Values. We adopt commonly used values for most hyperparameters, but tuned the sizes of hidden layers on the development set. The values are summarized in Table 2.

5.2 Development Experiments

We perform development experiments to verify the usefulness of various context representations, network configurations and different pretraining methods, respectively.

5.2.1 Context Representations

The influence of character and word context representations are empirically studied by varying the network structures for X_C and X_W in Figure 1, respectively. All the experiments in this section are performed using a beam size of 8.

Character Context. We fix the word representation X_W to a 2-word window and compare different character context representations. The results are shown in Table 4, where "no char" represents our model without X_C, "5-char window" represents a five-character window context, "char LSTM" represents character LSTM context and

[4]https://catalog.ldc.upenn.edu/LDC2011T13
[5]http://www.icl.pku.edu.cn/icl_res

Character	P	R	F
No char	82.19	87.20	84.62
5-char window	95.33	95.50	95.41
char LSTM	95.21	95.82	95.51
5-char window+LSTM	95.77	95.95	**95.86**
-char emb	95.20	95.19	95.20
-bichar emb	93.87	94.67	94.27

Table 4: Influence of character contexts.

Word	P	R	F
No word	95.50	95.83	95.66
1-word window	95.70	95.85	95.78
2-word window	95.77	95.95	**95.86**
3-word window	95.80	95.85	95.83
word LSTM	95.71	95.97	95.84
2-word window+LSTM	95.74	95.95	95.84

Table 5: Influence of word contexts.

"5-char window + LSTM" represents a combination, detailed in Section 3.1. "-char emb" and "-bichar emb" represent the combined window and LSTM context without character and character-bigram information, respectively.

As can be seen from the table, without character information, the F-score is 84.62%, demonstrating the necessity of character contexts. Using window and LSTM representations, the F-scores increase to 95.41% and 95.51%, respectively. A combination of the two lead to further improvement, showing that local and global character contexts are indeed complementary, as hypothesized in Section 3.1. Finally, by removing character and character-bigram embeddings, the F-score decreases to 95.20% and 94.27%, respectively, which suggests that character bigrams are more useful compared to character unigrams. This is likely because they contain more distinct tokens and hence offer a larger parameter space.

Word Context. The influence of various word contexts are shown in Table 5. Without using word information, our segmentor gives an F-score of 95.66% on the development data. Using a context of only w_{-1} (1-word window), the F-measure increases to 95.78%. This shows that word contexts are far less important in our model compared to character contexts, and also compared to word contexts in previous word-based segmentors (Zhang et al., 2016b; Cai and Zhao, 2016). This is likely due to the difference in our neural network structures, and that we fine-tune both character and character bigram embeddings, which significantly enlarges the adjustable parameter space as compared with Zhang et al. (2016b). The fact that word contexts can contribute relatively less than characters in a word is also not surprising in the sense that word-based neural segmentors do not outperform the best character-based models by large margins. Given that character context is what we pretrain, our model relies more heavily on them.

With both w_{-2} and w_{-1} being used for the context, the F-score further increases to 95.86%, showing that a 2-word window is useful by offering more contextual information. On the other hand, when w_{-3} is also considered, the F-score does not improve further. This is consistent with previous findings of statistical word segmentation (Zhang and Clark, 2007), which adopt a 2-word context. Interestingly, using a word LSTM does not bring further improvements, even when it is combined with a window context. This suggests that global word contexts may not offer crucial additional information compared with local word contexts. Intuitively, words are significantly less polysemous compared with characters, and hence can serve as effective contexts even if used locally, to supplement a more crucial character context.

5.2.2 Stuctured Learning and Inference

We verify the effectiveness of structured learning and inference by measuring the influence of beam size on the baseline segmentor. Figure 4 shows the F-scores against different numbers of training iterations with beam size 1,2,4,8 and 16, respectively. When the beam size is 1, the inference is local and greedy. As the size of the beam increases, more global structural ambiguities can be resolved since learning is designed to guide search. A contrast between beam sizes 1 and 2 demonstrates the usefulness of structured learning and inference. As the beam size increases, the gain by doubling the beam size decreases. We choose a beam size of 8 for the remaining experiments for a tradeoff between speed and accuracy.

5.2.3 Pretraining Results

Table 6 shows the effectiveness of rich pretraining of D_c on the development set. In particular, by using punctuation information, the F-score increases from 95.86% to 96.25%, with a relative error reduction of 9.4%. This is consistent with

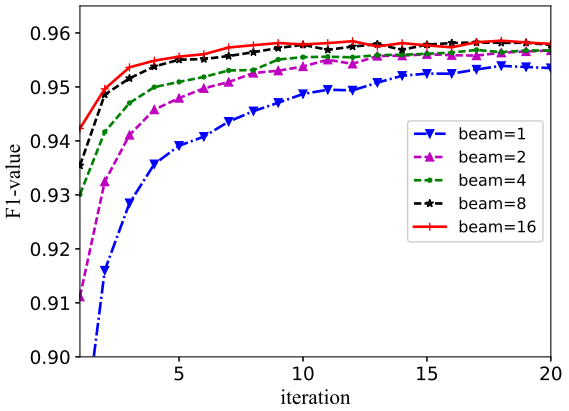


Figure 4: F1 measure against the training epoch.

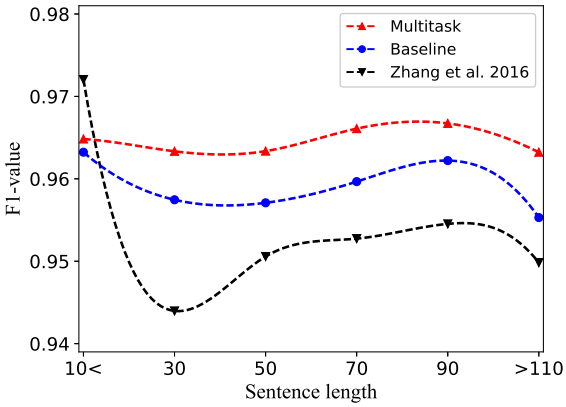


Figure 5: F1 measure against the sentence length.

Pretrain	P	R	F	ER%
Baseline	95.77	95.95	95.86	0
+Punc. pretrain	96.36	96.13	96.25	-9.4
+Auto-seg pretrain	96.23	96.29	96.26	-9.7
+Heter-seg pretrain	96.28	96.27	96.27	-9.9
+POS pretrain	96.16	96.28	96.22	-8.7
+Multitask pretrain	96.54	96.42	96.48	-15.0

Table 6: Influence of pretraining.

the observation of Sun and Xu (2011), who show that punctuation is more effective compared with mutual information and access variety as semi-supervised data for a statistical word segmentation model. With automatically-segmented data[6], heterogenous segmentation and POS information, the F-score increases to 96.26%, 96.27% and 96.22%, respectively, showing the relevance of all information sources to neural segmentation, which is consistent with observations made for statistical word segmentation (Jiang et al., 2009; Wang et al., 2011; Zhang et al., 2013). Finally, by integrating all above information via multi-task learning, the F-score is further improved to 96.48%, with a 15.0% relative error reduction.

5.2.4 Comparision with Zhang et al. (2016b)

Both our model and Zhang et al. (2016b) use global learning and beam search, but our network is different. Zhang et al. (2016b) utilizes the action history with LSTM encoder, while we use partial word rather than action information. Besides, the character and character bigram embeddings are fine-tuned in our model while Zhang et al. (2016b) set the embeddings fixed during training.

[6]By using ZPar alone, the auto-segmented result is 96.02%, less than using results by matching ZPar and the CRF segmentor outputs.

We study the F-measure distribution with respect to sentence length on our baseline model, multi-task pretraining model and Zhang et al. (2016b). In particular, we cluster the sentences in the development dataset into 6 categories based on their length and evaluate their F1-values, respectively. As shown in Figure 5, the models give different error distributions, with our models being more robust to the sentence length compared with Zhang et al. (2016b). Their model is better on very short sentences, but worse on all other cases. This shows the relative advantages of our model.

5.3 Final Results

Our final results on CTB6 are shown in Table 7, which lists the results of several current state-of-the-art methods. Without multitask pretraining, our model gives an F-score of 95.44%, which is higher than the neural segmentor of Zhang et al. (2016b), which gives the best accuracies among pure neural segments on this dataset. By using multitask pretraining, the result increases to 96.21%, with a relative error reduction of 16.9%. In comparison, Sun and Xu (2011) investigated heterogenous semi-supervised learning on a state-of-the-art *statistical* model, obtaining a relative error reduction of 13.8%. Our findings show that external data can be as useful for *neural* segmentation as for statistical segmentation.

Our final results compare favourably to the best statistical models, including those using semi-supervised learning (Sun and Xu, 2011; Wang et al., 2011), and those leveraging joint POS and syntactic information (Zhang et al., 2014). In addition, it also outperforms the best neural models, in particular Zhang et al. (2016b)*, which is a hybrid neural and statistical model, integrating man-

Models	P	R	F
Baseline	95.3	95.5	95.4
Punc. pretrain	96.0	95.6	95.8
Auto-seg pretrain	95.8	95.6	95.7
Multitask pretrain	**96.4**	**96.0**	**96.2**
Sun and Xu (2011) baseline	95.2	94.9	95.1
Sun and Xu (2011) multi-source semi	95.9	95.6	95.7
Zhang et al. (2016b) neural	95.3	94.7	95.0
Zhang et al. (2016b)* hybrid	96.1	95.8	96.0
Chen et al. (2015a) window	95.7	95.8	95.8
Chen et al. (2015b) char LSTM	96.2	95.8	96.0
Zhang et al. (2014) POS and syntax	–	–	95.7
Wang et al. (2011) statistical semi	95.8	95.8	95.8
Zhang and Clark (2011) statistical	95.5	94.8	95.1

Table 7: Main results on CTB6.

F1 measure	PKU	MSR	AS	CityU	Weibo
Multitask pretrain	**96.3**	97.5	**95.7**	**96.9**	**95.5**
Cai and Zhao (2016)	95.5	96.5	–	–	–
Zhang et al. (2016b)	95.1	97.0	–	–	–
Zhang et al. (2016b)*	95.7	**97.7**	–	–	–
Pei et al. (2014)	95.2	97.2	–	–	–
Sun et al. (2012)	95.4	97.4	–	–	–
Zhang and Clark (2007)	94.5	97.2	94.6	95.1	–
Zhang et al. (2006)	95.1	97.1	95.1	95.1	–
Sun et al. (2009)	95.2	97.3	–	94.6	–
Sun (2010)	95.2	96.9	95.2	95.6	–
Wang et al. (2014)	95.3	97.4	95.4	94.7	–
Xia et al. (2016)	–	–	–	–	95.4

Table 8: Main results on other test datasets.

ual discrete features into their word-based neural model. We achieve the best reported F-score on this dataset. To our knowledge, this is the first time a pure neural network model outperforms all existing methods on this dataset, allowing the use of external data [7]. We also evaluate our model pretrained only on punctuation and auto-segmented data, which do not include additional manual labels. The results on CTB test data show the accuracy of 95.8% and 95.7%, respectivley, which are comparable with those statistical semi-supervised methods (Sun and Xu, 2011; Wang et al., 2011). They are also among the top performance methods in Table 7. Compared with discrete semi-supervised methods (Sun and Xu, 2011; Wang et al., 2011), our semi-supervised model is free from hand-crafted features.

In addition to CTB6, which has been the most commonly adopted by recent segmentation research, we additionally evaluate our results on the SIGHAN 2005 bakeoff and Weibo datasets, to examine cross domain robustness. Different state-of-the-art methods for which results are recorded on these datasets are listed in Table 8. Most neural models reported results only on the PKU [8] and MSR datasets of the bakeoff test sets, which are in simplified Chinese. The AS and CityU corpora are in traditional Chinese, sourced from Taiwan and Hong Kong corpora, respectively. We map them into simplified Chinese before segmentation. The Weibo corpus is in a yet different genre, being social media text. Xia et al. (2016) achieved the best results on this dataset by using a statistical model with features learned using external lexicons, the CTB7 corpus and the People Daily corpus. Similar to Table 7, our method gives the best accuracies on all corpora except for MSR, where it underperforms the hybrid model of Zhang et al. (2016b) by 0.2%. To our knowledge, we are the first to report results for a neural segmentor on more than 3 datasets, with competitive results consistently. It verifies that knowledge learned from a certain set of resources can be used to enhance cross-domain robustness in training a neural segmentor for different datasets, which is of practical importance.

6 Conclusion

We investigated rich external resources for enhancing neural word segmentation, by building a globally optimised beam-search model that leverages both character and word contexts. Taking each type of external resource as an auxiliary classification task, we use neural multi-task learning to pre-train a set of shared parameters for character contexts. Results show that rich pretraining leads to 15.4% relative error reduction, and our model gives results highly competitive to the best systems on six different benchmarks.

Acknowledgments

We thank the anonymous reviewers for their insightful comments and the support of NSFC 61572245. We would like to thank Meishan Zhang for his insightful discussion and assisting coding. Yue Zhang is the corresponding author.

[7] We did not investigate the use of lexicons (Chen et al., 2015a,b) in our research, since lexicons might cover different OOV in the training and test data, and hence directly affecting the accuracies, which makes it relatively difficult to compare different methods fairly unless a single lexicon is used for all methods, as observed by Cai and Zhao (2016).

[8] We notice that both PKU dataset and our heterogenous data are based on the news of People's Daily. While the heterogenous data only collect news from Febuary 1998 to June 1998, it does not contain the sentences in the dev and test datasets of PKU.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *ACL*. Association for Computational Linguistics, pages 2442–2452. https://doi.org/10.18653/v1/P16-1231.

Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, Springer, pages 437–478.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. In *ACL*. Association for Computational Linguistics, pages 409–420. https://doi.org/10.18653/v1/P16-1039.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In *ACL*. Association for Computational Linguistics. https://doi.org/10.3115/v1/P15-1168.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*. Association for Computational Linguistics, pages 1385–1394. https://doi.org/10.18653/v1/D15-1141.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL*. Association for Computational Linguistics, page 111. http://aclweb.org/anthology/P04-1015.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*. volume 133.

Felix A Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*. IEEE, volume 3, pages 189–194.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and pos tagging: a case study. In *ACL-IJCNLP*. Association for Computational Linguistics, pages 522–530. http://aclweb.org/anthology/P09-1059.

Zhongguo Li and Maosong Sun. 2009. Punctuation as implicit annotations for chinese word segmentation. *Computational Linguistics* 35(4):505–512. http://aclweb.org/anthology/J09-4006.

Yang Liu and Yue Zhang. 2012. Unsupervised domain adaptation for joint segmentation and pos-tagging. In *COLING*. pages 745–754. http://aclweb.org/anthology/C12-2073.

Mairgup Mansur, Wenzhe Pei, and Baobao Chang. 2013. Feature-based neural language model and chinese word segmentation. In *IJCNLP*. pages 1271–1277. http://aclweb.org/anthology/I13-1181.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological analysis for unsegmented languages using recurrent neural network language model. In *EMNLP*. Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1276.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *EMNLP*. Association for Computational Linguistics, pages 277–284. http://aclweb.org/anthology/W04-3236.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *ACL*. Association for Computational Linguistics, pages 293–303. https://doi.org/10.3115/v1/P14-1028.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *COLING*. page 562. http://aclweb.org/anthology/C04-1081.

Xipeng Qiu, Peng Qian, and Zhan Shi. 2016. Overview of the nlpcc-iccpol 2016 shared task: Chinese word segmentation for micro-blog texts. In *International Conference on Computer Processing of Oriental Languages*. Springer, pages 901–906.

Valentin I Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010. Profiting from mark-up: Hyper-text annotations for guided parsing. In *ACL*. Association for Computational Linguistics, pages 1278–1287. http://aclweb.org/anthology/P10-1130.

Richard Sproat, William Gale, Chilin Shih, and Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for chinese. *Computational linguistics* 22(3):377–404. http://aclweb.org/anthology/J96-3004.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *COLING*. pages 1211–1219. http://aclweb.org/anthology/C10-2139.

Weiwei Sun and Jia Xu. 2011. Enhancing chinese word segmentation using unlabeled data. In *EMNLP*. Association for Computational Linguistics, pages 970–979. http://aclweb.org/anthology/D11-1090.

Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *ACL*. Association for Computational Linguistics, pages 253–262. http://aclweb.org/anthology/P12-1027.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. A discriminative latent variable chinese segmenter with hybrid word/character information. In *NAACL-HLT*. Association for Computational Linguistics, pages 56–64. http://aclweb.org/anthology/N09-1007.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*.

Mengqiu Wang, Rob Voigt, and Christopher D Manning. 2014. Two knives cut better than one: Chinese word segmentation with dual decomposition. In *ACL*. Association for Computational Linguistics, pages 193–198. https://doi.org/10.3115/v1/P14-2032.

Yiou Wang, Yoshimasa Tsuruoka Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *IJCNLP*. pages 309–317. http://www.aclweb.org/anthology/I11-1035.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*. Association for Computational Linguistics, pages 1296–1306. http://aclweb.org/anthology/D16-1137.

Qingrong Xia, Zhenghua Li, Jiayuan Chao, and Min Zhang. 2016. Word segmentation on micro-blog texts with external lexicon and heterogeneous data. In *International Conference on Computer Processing of Oriental Languages*. Springer.

Jingjing Xu and Xu Sun. 2016. Dependency-based gated recursive neural network for chinese word segmentation. In *ACL*. Association for Computational Linguistics, page 567. https://doi.org/10.18653/v1/P16-2092.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering* 11(02):207–238.

Nianwen Xue et al. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing* 8(1):29–48.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for chinese word segmentation. In *EMNLP*. Association for Computational Linguistics, pages 311–321. http://aclweb.org/anthology/D13-1031.

Meishan Zhang, Jie Yang, Zhiyang Teng, and Yue Zhang. 2016a. Libn3l: a lightweight package for neural nlp. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*. https://doi.org/10.1145/322234.322243.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level chinese dependency parsing. In *ACL*. Association for Computational Linguistics, pages 1326–1336. https://doi.org/10.3115/v1/P14-1125.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016b. Transition-based neural word segmentation. In *ACL*. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1040.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for chinese word segmentation. In *NAACL*. Association for Computational Linguistics, pages 193–196. http://aclweb.org/anthology/N06-2049.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *ACL*. Association for Computational Linguistics, volume 45, page 840. http://aclweb.org/anthology/P07-1106.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and pos tagging using a single perceptron. In *ACL*. Association for Computational Linguistics, pages 888–896. http://aclweb.org/anthology/P08-1101.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics* 37(1):105–151. https://doi.org/10.1162/coli_a_00037.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006. Effective tag set selection in chinese word segmentation via conditional random field modeling. In *PACLIC*. Citeseer, volume 20, pages 87–94. http://aclweb.org/anthology/Y06-1012.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*. Association for Computational Linguistics, pages 647–657. http://aclweb.org/anthology/D13-1061.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *ACL*. Association for Computational Linguistics, pages 1213–1222. https://doi.org/10.3115/v1/P15-1117.

Neural Machine Translation via Binary Code Prediction

Yusuke Oda[†] **Philip Arthur**[†] **Graham Neubig**[‡†] **Koichiro Yoshino**[†§] **Satoshi Nakamura**[†]
[†] Nara Institute of Science and Technoloty, 8916-5 Takayama-cho, Ikoma, Nara 630-0192, Japan
[‡] Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
[§] Japan Science and Technology Agency, 4-1-8 Hon-machi, Kawaguchi, Saitama 332-0012, Japan
{oda.yusuke.on9, philip.arthur.om0}@is.naist.jp, gneubig@cs.cmu.edu,
{koichiro, s-nakamura}@is.naist.jp

Abstract

In this paper, we propose a new method for calculating the output layer in neural machine translation systems. The method is based on predicting a binary code for each word and can reduce computation time/memory requirements of the output layer to be logarithmic in vocabulary size in the best case. In addition, we also introduce two advanced approaches to improve the robustness of the proposed model: using error-correcting codes and combining softmax and binary codes. Experiments on two English ↔ Japanese bidirectional translation tasks show proposed models achieve BLEU scores that approach the softmax, while reducing memory usage to the order of less than 1/10 and improving decoding speed on CPUs by x5 to x10.

1 Introduction

When handling broad or open domains, machine translation systems usually have to handle a large vocabulary as their inputs and outputs. This is particularly a problem in neural machine translation (NMT) models (Sutskever et al., 2014), such as the attention-based models (Bahdanau et al., 2014; Luong et al., 2015) shown in Figure 1. In these models, the output layer is required to generate a specific word from an internal vector, and a large vocabulary size tends to require a large amount of computation to predict each of the candidate word probabilities.

Because this is a significant problem for neural language and translation models, there are a number of methods proposed to resolve this problem, which we detail in Section 2.2. However, none of these previous methods simultaneously satisfies the following desiderata, all of which, we argue, are desirable for practical use in NMT systems:

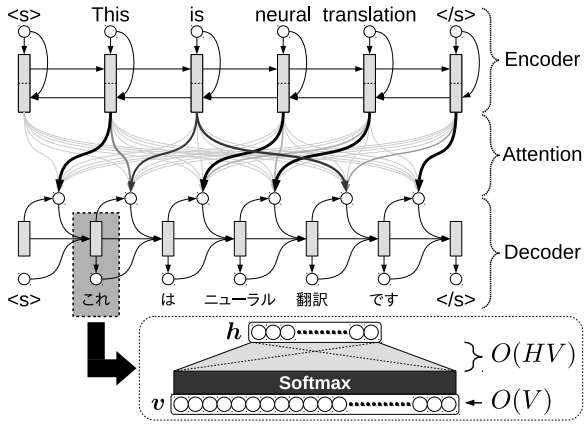


Figure 1: Encoder-decoder-attention NMT model and computation amount of the output layer.

Memory efficiency: The method should not require large memory to store the parameters and calculated vectors to maintain scalability in resource-constrained environments.

Time efficiency: The method should be able to train the parameters efficiently, and possible to perform decoding efficiently with choosing the candidate words from the full probability distribution. In particular, the method should be performed fast on general CPUs to suppress physical costs of computational resources for actual production systems.

Compatibility with parallel computation: It should be easy for the method to be mini-batched and optimized to run efficiently on GPUs, which are essential for training large NMT models.

In this paper, we propose a method that satisfies all of these conditions: requires significantly less memory, fast, and is easy to implement mini-batched on GPUs. The method works by not predicting a softmax over the entire output vocab-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 850–860
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1079

ulary, but instead by encoding each vocabulary word as a vector of binary variables, then independently predicting the bits of this binary representation. In order to represent a vocabulary size of 2^n, the binary representation need only be at least n bits long, and thus the amount of computation and size of parameters required to select an output word is only $O(\log V)$ in the size of the vocabulary V, a great reduction from the standard linear increase of $O(V)$ seen in the original softmax.

While this idea is simple and intuitive, we found that it alone was not enough to achieve competitive accuracy with real NMT models. Thus we make two improvements: First, we propose a hybrid model, where the high frequency words are predicted by a standard softmax, and low frequency words are predicted by the proposed binary codes separately. Second, we propose the use of convolutional error correcting codes with Viterbi decoding (Viterbi, 1967), which add redundancy to the binary representation, and even in the face of localized mistakes in the calculation of the representation, are able to recover the correct word.

In experiments on two translation tasks, we find that the proposed hybrid method with error correction is able to achieve results that are competitive with standard softmax-based models while reducing the output layer to a fraction of its original size.

2 Problem Description and Prior Work

2.1 Formulation and Standard Softmax

Most of current NMT models use *one-hot representations* to represent the words in the output vocabulary – each word w is represented by a unique sparse vector $\boldsymbol{e}_{\mathrm{id}(w)} \in \mathbb{R}^V$, in which only one element at the position corresponding to the word ID $\mathrm{id}(w) \in \{x \in \mathbb{N} \mid 1 \leq x \leq V\}$ is 1, while others are 0. V represents the vocabulary size of the target language. NMT models optimize network parameters by treating the one-hot representation $\boldsymbol{e}_{\mathrm{id}(w)}$ as the true probability distribution, and minimizing the cross entropy between it and the softmax probability $\boldsymbol{v}$:

$$
\begin{aligned}
L_{\mathcal{H}}(\boldsymbol{v}, \mathrm{id}(w)) &:= \mathcal{H}(\boldsymbol{e}_{\mathrm{id}(w)}, \boldsymbol{v}), && (1)\\
&= \log \operatorname{sum} \exp \boldsymbol{u} - u_{\mathrm{id}(w)}, && (2)\\
\boldsymbol{v} &:= \exp \boldsymbol{u} / \operatorname{sum} \exp \boldsymbol{u}, && (3)\\
\boldsymbol{u} &:= W_{hu}\boldsymbol{h} + \boldsymbol{\beta}_u, && (4)
\end{aligned}
$$

where $\operatorname{sum} \boldsymbol{x}$ represents the sum of all elements in $\boldsymbol{x}$, x_i represents the i-th element of $\boldsymbol{x}$, $W_{hu} \in \mathbb{R}^{V \times H}$ and $\boldsymbol{\beta}_u \in \mathbb{R}^V$ are trainable parameters and H is the total size of hidden layers directly connected to the output layer.

According to Equation (4), this model clearly requires time/space computation in proportion to $O(HV)$, and the actual load of the computation of the output layer is directly affected by the size of vocabulary V, which is typically set around tens of thousands (Sutskever et al., 2014).

2.2 Prior Work on Suppressing Complexity of NMT Models

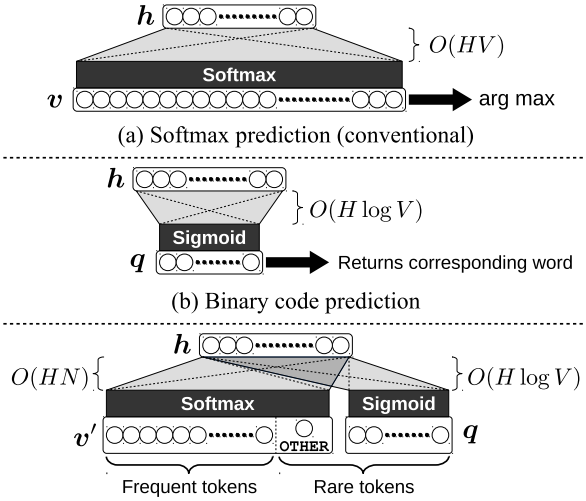
Several previous works have proposed methods to reduce computation in the output layer. The *hierarchical softmax* (Morin and Bengio, 2005) predicts each word based on binary decision and reduces computation time to $O(H \log V)$. However, this method still requires $O(HV)$ space for the parameters, and requires calculation much more complicated than the standard softmax, particularly at test time.

The *differentiated softmax* (Chen et al., 2016) divides words into clusters, and predicts words using separate part of the hidden layer for each word clusters. This method make the conversion matrix of the output layer sparser than a fully-connected softmax, and can reduce time/space computation amount by ignoring zero part of the matrix. However, this method restricts the usage of hidden layer, and the size of the matrix is still in proportion to V.

Sampling-based approximations (Mnih and Teh, 2012; Mikolov et al., 2013) to the denominator of the softmax have also been proposed to reduce calculation at training. However, these methods are basically not able to be applied at test time, still require heavy computation like the standard softmax.

Vocabulary selection approaches (Mi et al., 2016; L'Hostis et al., 2016) can also reduce the vocabulary size at testing, but these methods abandon full search over the target space and the quality of picked vocabularies directly affects the translation quality.

Other methods using characters (Ling et al., 2015) or subwords (Sennrich et al., 2016; Chitnis and DeNero, 2015) can be applied to suppress the vocabulary size, but these methods also make for longer sequences, and thus are not a direct solution to problems of computational efficiency.



(a) Softmax prediction (conventional)

(b) Binary code prediction

(c) Hybrid prediction (softmax & binary code)

Figure 2: Designs of output layers.

3 Binary Code Prediction Models

3.1 Representing Words using Bit Arrays

Figure 2(a) shows the conventional softmax prediction, and Figure 2(b) shows the binary code prediction model proposed in this study. Unlike the conventional softmax, the proposed method predicts each output word indirectly using dense bit arrays that correspond to each word. Let $\boldsymbol{b}(w) := [b_1(w), b_2(w), \cdots, b_B(w)] \in \{0,1\}^B$ be the target bit array obtained for word w, where each $b_i(w) \in \{0,1\}$ is an independent binary function given w, and B is the number of bits in whole array. For convenience, we introduce some constraints on $\boldsymbol{b}$. First, a word w is mapped to only one bit array $\boldsymbol{b}(w)$. Second, all unique words can be discriminated by $\boldsymbol{b}$, i.e., all bit arrays satisfy that:[1]

$$\mathrm{id}(w) \neq \mathrm{id}(w') \Rightarrow \boldsymbol{b}(w) \neq \boldsymbol{b}(w'). \quad (5)$$

Third, multiple bit arrays can be mapped to the same word as described in Section 3.5. By considering second constraint, we can also constrain $B \geq \lceil \log_2 V \rceil$, because $\boldsymbol{b}$ should have at least V unique representations to distinguish each word. The output layer of the network independently predicts B probability values $\boldsymbol{q} := [q_1(\boldsymbol{h}), q_2(\boldsymbol{h}), \cdots, q_B(\boldsymbol{h})] \in [0,1]^B$ using the current hidden values $\boldsymbol{h}$ by logistic regressions:

$$\boldsymbol{q}(\boldsymbol{h}) = \sigma(W_{hq}\boldsymbol{h} + \boldsymbol{\beta}_q), \quad (6)$$
$$\sigma(\boldsymbol{x}) := 1/(1+\exp(-\boldsymbol{x})), \quad (7)$$

where $W_{hq} \in \mathbb{R}^{B\times H}$ and $\boldsymbol{\beta}_q \in \mathbb{R}^B$ are trainable parameters. When we assume that each q_i is the probability that "the i-th bit becomes 1," the joint probability of generating word w can be represented as:

$$\Pr(\boldsymbol{b}(w)|\boldsymbol{q}(\boldsymbol{h})) := \prod_{i=1}^{B} \left(b_i q_i + \bar{b}_i \bar{q}_i\right), \quad (8)$$

where $\bar{x} := 1 - x$. We can easily obtain the maximum-probability bit array from $\boldsymbol{q}$ by simply assuming the i-th bit is 1 if $q_i \geq 1/2$, or 0 otherwise. However, this calculation may generate invalid bit arrays which do not correspond to actual words according to the mapping between words and bit arrays. For now, we simply assume that $w = \texttt{UNK}$ (*unknown*) when such bit arrays are obtained, and discuss alternatives later in Section 3.5.

The constraints described here are very general requirements for bit arrays, which still allows us to choose between a wide variety of mapping functions. However, designing the most appropriate mapping method for NMT models is not a trivial problem. In this study, we use a simple mapping method described in Algorithm 1, which was empirically effective in preliminary experiments.[2] Here, $\mathcal{V}$ is the set of V target words including 3 extra markers: UNK, BOS (*begin-of-sentence*), and EOS (*end-of-sentence*), and $\mathrm{rank}(w) \in \mathbb{N}_{>0}$ is the rank of the word according to their frequencies in the training corpus. Algorithm 1 is one of the minimal mapping methods (i.e., satisfying $B = \lceil \log_2 V \rceil$), and generated bit arrays have the characteristics that their *higher* bits roughly represents the frequency of corresponding words (e.g., if w is frequently appeared in the training corpus, higher bits in $\boldsymbol{b}(w)$ tend to become 0).

3.2 Loss Functions

For learning correct binary representations, we can use any loss functions that is (sub-)differentiable and satisfies a constraint that:

$$L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}) \begin{cases} = \epsilon_L, & \text{if } \boldsymbol{q} = \boldsymbol{b}, \\ \geq \epsilon_L, & \text{otherwise}, \end{cases} \quad (9)$$

[1] We designed this injective condition using the $\mathrm{id}(\cdot)$ function to ignore task-specific sensitivities between different word surfaces (e.g. cases, ligatures, etc.).

[2] Other methods examined included random codes, Huffman codes (Huffman, 1952) and Brown clustering (Brown et al., 1992) with zero-padding to adjust code lengths, and some original allocation methods based on the word2vec embeddings (Mikolov et al., 2013).

Algorithm 1 Mapping words to bit arrays.

Require: $w \in \mathcal{V}$
Ensure: $\boldsymbol{b} \in \{0,1\}^B$ = Bit array representing w

$$x := \begin{cases} 0, & \text{if } w = \texttt{UNK} \\ 1, & \text{if } w = \texttt{BOS} \\ 2, & \text{if } w = \texttt{EOS} \\ 2 + \mathrm{rank}(w), & \text{otherwise} \end{cases}$$

$b_i := \lfloor x / 2^{i-1} \rfloor \mod 2$
$\boldsymbol{b} \leftarrow [b_1, b_2, \cdots, b_B]$

where ϵ_L is the minimum value of the loss function which typically does not affect the gradient descent methods. For example, the *squared-distance*:

$$L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}) := \sum_{i=1}^{B} (q_i - b_i)^2, \tag{10}$$

or the *cross-entropy*:

$$L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}) := -\sum_{i=1}^{B} \left(b_i \log q_i + \bar{b}_i \log \bar{q}_i \right), \tag{11}$$

are candidates for the loss function. We also examined both loss functions in the preliminary experiments, and in this paper, we only used the squared-distance function (Equation (10)), because this function achieved higher translation accuracies than Equation (11).[3]

3.3 Efficiency of the Binary Code Prediction

The computational complexity for the parameters W_{hq} and $\boldsymbol{\beta}_q$ is $O(HB)$. This is equal to $O(H \log V)$ when using a minimal mapping method like that shown in Algorithm 1, and is significantly smaller than $O(HV)$ when using standard softmax prediction. For example, if we chose $V = 65536 = 2^{16}$ and use Algorithm 1's mapping method, then $B = 16$ and total amount of computation in the output layer could be suppressed to 1/4096 of its original size.

On a different note, the binary code prediction model proposed in this study shares some ideas with the hierarchical softmax (Morin and Bengio, 2005) approach. Actually, when we used a binary-tree based mapping function for $\boldsymbol{b}$, our model can be interpreted as the hierarchical softmax with two strong constraints for guaranteeing independence between all bits: all nodes in the same level of the hierarchy share their parameters, and all levels of the hierarchy are predicted independently of each other. By these constraints, all bits in $\boldsymbol{b}$ can be calculated in parallel. This is particularly important because it makes the model conducive to being calculated on parallel computation backends such as GPUs.

However, the binary code prediction model also introduces problems of robustness due to these strong constraints. As the experimental results show, the simplest prediction model which directly maps words into bit arrays seriously decreases translation quality. In Sections 3.4 and 3.5, we introduce two additional techniques to prevent reductions of translation quality and improve robustness of the binary code prediction model.

3.4 Hybrid Softmax/Binary Model

According to the Zipf's law (Zipf, 1949), the distribution of word appearances in an actual corpus is biased to a small subset of the vocabulary. As a result, the proposed model mostly learns characteristics for frequent words and cannot obtain enough opportunities to learn for rare words. To alleviate this problem, we introduce a hybrid model using both softmax prediction and binary code prediction as shown in Figure 2(c). In this model, the output layer calculates a standard softmax for the $N-1$ most frequent words and an `OTHER` marker which indicates all rare words. When the softmax layer predicts `OTHER`, then the binary code layer is used to predict the representation of rare words. In this case, the actual probability of generating a particular word can be separated into two equations according to the frequency of words:

$$\Pr(w|\boldsymbol{h}) \simeq \begin{cases} v'_{\mathrm{id}(w)}, & \text{if } \mathrm{id}(w) < N, \\ v'_N \cdot \pi(w, \boldsymbol{h}), & \text{otherwise}, \end{cases} \tag{12}$$

$$\boldsymbol{v}' := \exp \boldsymbol{u}' / \mathrm{sum} \exp \boldsymbol{u}', \tag{13}$$

$$\boldsymbol{u}' := W_{hu'} \boldsymbol{h} + \boldsymbol{\beta}_{u'}, \tag{14}$$

$$\pi(w, \boldsymbol{h}) := \Pr(\boldsymbol{b}(w) | \boldsymbol{q}(\boldsymbol{h})), \tag{15}$$

where $W_{hu'} \in \mathbb{R}^{N \times H}$ and $\boldsymbol{\beta}_{u'} \in \mathbb{R}^N$ are trainable parameters, and $\mathrm{id}(w)$ assumes that the value corresponds to the rank of frequency of each word. We also define the loss function for the hybrid

[3] In terms of learning probabilistic models, we should remind that using Eq. (10) is an approximation of Eq. (11). The output bit scores trained by Eq. (10) do not represent actual word perplexities, and this characteristics imposes some practical problems when comparing multiple hypotheses (e.g., reranking, beam search, etc.). We could ignore this problem in this paper because we only evaluated the one-best results in experiments.

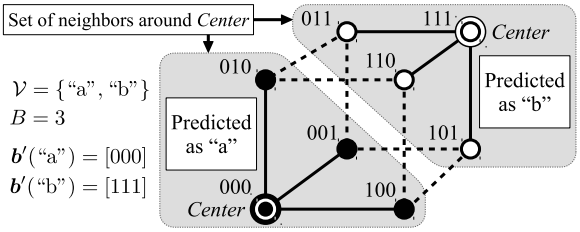


Figure 3: Example of the classification problem using redundant bit array mapping.

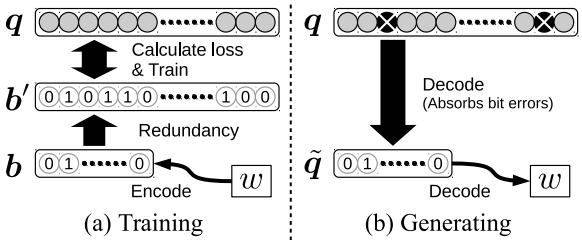


Figure 4: Training and generation processes with error-correcting code.

model using both softmax and binary code losses:

$$L := \begin{cases} l_{\mathcal{H}}(\mathrm{id}(w)), & \text{if } \mathrm{id}(w) < N, \\ l_{\mathcal{H}}(N) + l_{\mathcal{B}}, & \text{otherwise,} \end{cases} \quad (16)$$

$$l_{\mathcal{H}}(i) := \lambda_{\mathcal{H}} L_{\mathcal{H}}(\boldsymbol{v}', i), \quad (17)$$

$$l_{\mathcal{B}} := \lambda_{\mathcal{B}} L_{\mathcal{B}}(\boldsymbol{q}, \boldsymbol{b}), \quad (18)$$

where $\lambda_{\mathcal{H}}$ and $\lambda_{\mathcal{B}}$ are hyper-parameters to determine strength of both softmax/binary code losses. These also can be adjusted according to the training data, but in this study, we only used $\lambda_{\mathcal{H}} = \lambda_{\mathcal{B}} = 1$ for simplicity.

The computational complexity of the hybrid model is $O(H(N + \log V))$, which is larger than the original binary code model $O(H \log V)$. However, N can be chosen as $N \ll V$ because the softmax prediction is only required for a few frequent words. As a result, we can control the actual computation for the hybrid model to be much smaller than the standard softmax complexity $O(HV)$,

The idea of separated prediction of frequent words and rare words comes from the differentiated softmax (Chen et al., 2016) approach. However, our output layer can be configured as a fully-connected network, unlike the differentiated softmax, because the actual size of the output layer is still small after applying the hybrid model.

3.5 Applying Error-correcting Codes

The 2 methods proposed in previous sections impose constraints for all bits in $\boldsymbol{q}$, and the value of each bit must be estimated correctly for the correct word to be chosen. As a result, these models may generate incorrect words due to even a single bit error. This problem is the result of dense mapping between words and bit arrays, and can be avoided by creating *redundancy* in the bit array. Figure 3 shows a simple example of how this idea works when discriminating 2 words using 3 bits. In this case, the actual words are obtained by estimating the nearest *centroid* bit array according to the Hamming distance between each centroid and the predicted bit array. This approach can predict correct words as long as the predicted bit arrays are in the set of *neighbors* for the correct centroid (gray regions in the Figure 3), i.e., up to a 1-bit error in the predicted bits can be corrected. This ability to be robust to errors is a central idea behind *error-correcting codes* (Shannon, 1948). In general, an error-correcting code has the ability to correct up to $\lfloor (d-1)/2 \rfloor$ bit errors when all centroids differ d bits from each other (Golay, 1949). d is known as the *free distance* determined by the design of error-correcting codes. Error-correcting codes have been examined in some previous work on multi-class classification tasks, and have reported advantages from the raw classification (Dietterich and Bakiri, 1995; Klautau et al., 2003; Liu, 2006; Kouzani and Nasireding, 2009; Kouzani, 2010; Ferng and Lin, 2011, 2013). In this study, we applied an error-correcting algorithm to the bit array obtained from Algorithm 1 to improve robustness of the output layer in an NMT system. A challenge in this study is trying a large classification (#classes $>$ 10,000) with error-correction, unlike previous studies focused on solving comparatively small tasks (#classes $<$ 100). And this study also tries to solve a generation task unlike previous studies. As shown in the experiments, we found that this approach is highly effective in these tasks.

Figure 4 (a) and (b) illustrate the training and generation processes for the model with error-correcting codes. In the training, we first convert the original bit arrays $\boldsymbol{b}(w)$ to a *center* bit array $\boldsymbol{b}'$ in the space of error-correcting code: $\boldsymbol{b}'(\boldsymbol{b}) := [\boldsymbol{b}'_1(\boldsymbol{b}), \boldsymbol{b}'_2(\boldsymbol{b}), \cdots, \boldsymbol{b}'_{B'}(\boldsymbol{b})] \in \{0,1\}^{B'}$, where $B'(B) \geq B$ is the number of bits in the error-correcting code. The NMT model learns its

Algorithm 2 Encoding into a convolutional code.

Require: $\boldsymbol{b} \in \{0,1\}^B$
Ensure: $\boldsymbol{b}' \in \{0,1\}^{2(B+6)}$ = Redundant bit array

$$x[t] := \begin{cases} b_t, & \text{if } 1 \le t \le B \\ 0, & \text{otherwise} \end{cases}$$
$$y_t^1 := x[t-6\,..\,t] \cdot [1001111] \mod 2$$
$$y_t^2 := x[t-6\,..\,t] \cdot [1101101] \mod 2$$
$$\boldsymbol{b}' \leftarrow [y_1^1, y_1^2, y_2^1, y_2^2, \cdots, y_{B+6}^1, y_{B+6}^2]$$

Algorithm 3 Decoding from a convolutional code.

Require: $\boldsymbol{q} \in (0,1)^{2(B+6)}$
Ensure: $\tilde{\boldsymbol{q}} \in \{0,1\}^B$ = Restored bit array

$g(q,b) := b \log q + (1-b)\log(1-q)$
$$\phi_0[\boldsymbol{s} \mid \boldsymbol{s} \in \{0,1\}^6] \leftarrow \begin{cases} 0, & \text{if } \boldsymbol{s} = [000000] \\ -\infty, & \text{otherwise} \end{cases}$$
for $t = 1 \to B+6$ **do**
  for $\boldsymbol{s}^{\text{cur}} \in \{0,1\}^6$ **do**
    $\boldsymbol{s}^{\text{prev}}(x) := [x] \circ \boldsymbol{s}^{\text{cur}}[1\,..\,5]$
    $o_1(x) := ([x] \circ \boldsymbol{s}^{\text{cur}}) \cdot [1001111] \mod 2$
    $o_2(x) := ([x] \circ \boldsymbol{s}^{\text{cur}}) \cdot [1101101] \mod 2$
    $g'(x) := g(q_{2t-1}, o_1(x)) + g(q_{2t}, o_2(x))$
    $\phi'(x) := \phi_{t-1}[\boldsymbol{s}^{\text{prev}}(x)] + g'(x)$
    $\hat{x} \leftarrow \arg\max_{x \in \{0,1\}} \phi'(x)$
    $r_t[\boldsymbol{s}^{\text{cur}}] \leftarrow \boldsymbol{s}^{\text{prev}}(\hat{x})$
    $\phi_t[\boldsymbol{s}^{\text{cur}}] \leftarrow \phi'(\hat{x})$
  end for
end for
$\boldsymbol{s}' \leftarrow [000000]$
for $t = B \to 1$ **do**
  $\boldsymbol{s}' \leftarrow r_{t+6}[\boldsymbol{s}']$
  $\tilde{q}_t \leftarrow s'_1$
end for
$\tilde{\boldsymbol{q}} \leftarrow [\tilde{q}_1, \tilde{q}_2, \cdots, \tilde{q}_B]$

parameters based on the loss between predicted probabilities $\boldsymbol{q}$ and $\boldsymbol{b}'$. Note that typical error-correcting codes satisfy $O(B'/B) = O(1)$, and this characteristic efficiently suppresses the increase of actual computation cost in the output layer due to the application of the error-correcting code. In the generation of actual words, the decoding method of the error-correcting code converts the redundant predicted bits $\boldsymbol{q}$ into a dense representation $\tilde{\boldsymbol{q}} := [\tilde{\boldsymbol{q}}_1(\boldsymbol{q}), \tilde{\boldsymbol{q}}_2(\boldsymbol{q}), \cdots, \tilde{\boldsymbol{q}}_B(\boldsymbol{q})]$, and uses $\tilde{\boldsymbol{q}}$ as the bits to restore the word, as is done in the method described in the previous sections.

It should be noted that the method for performing error correction directly affects the quality of the whole NMT model. For example, the mapping shown in Figure 3 has only 3 bits and it is clear that these bits represent exactly the same information as each other. In this case, all bits can be estimated using exactly the same parameters, and we can not expect that we will benefit significantly from applying this redundant representation. Therefore, we need to choose an error correction method in which the characteristics of original bits should be distributed in various positions of the resulting bit arrays so that errors in bits are not highly correlated with each-other. In addition, it is desirable that the decoding method of the applied error-correcting code can directly utilize the probabilities of each bit, because $\boldsymbol{q}$ generated by the network will be a continuous probabilities between zero and one.

In this study, we applied convolutional codes (Viterbi, 1967) to convert between original and redundant bits. Convolutional codes perform a set of bit-wise convolutions between original bits and weight bits (which are hyper-parameters). They are well-suited to our setting here because they distribute the information of original bits in different places in the resulting bits, work robustly for random bit errors, and can be decoded using bit probabilities directly.

Algorithm 2 describes the particular convolutional code that we applied in this study, with two convolution weights [1001111] and [1101101] as fixed hyper-parameters.[4] Where $\boldsymbol{x}[i\,..\,j] := [x_i, \cdots, x_j]$ and $\boldsymbol{x} \cdot \boldsymbol{y} := \sum_i x_i y_i$. On the other hand, there are various algorithms to decode convolutional codes with the same format which are based on different criteria. In this study, we use the decoding method described in Algorithm 3, where $\boldsymbol{x} \circ \boldsymbol{y}$ represents the concatenation of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. This method is based on the Viterbi algorithm (Viterbi, 1967) and estimates original bits by directly using probability of redundant bits. Although Algorithm 3 looks complicated, this algorithm can be performed efficiently on CPUs at test time, and is not necessary at training time when we are simply performing calculation of Equation (6). Algorithm 2 increases the number of bits from B into $B' = 2(B+6)$, but does not restrict the actual value of B.

[4] We also examined many configurations of convolutional codes which have different robustness and computation costs, and finally chose this one.

Table 1: Details of the corpus.

Name		ASPEC	BTEC
Languages		En ↔ Ja	
#sentences	Train	2.00 M	465. k
	Dev	1,790	510
	Test	1,812	508
Vocabulary size V		65536	25000

Table 2: Evaluated methods.

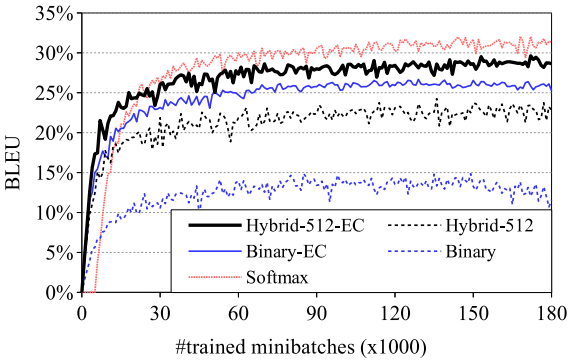
Name	Summary
Softmax	Softmax prediction (Fig. 2(a))
Binary	Fig. 2(b) w/ raw bit array
Hybrid-N	Fig. 2(c) w/ softmax size N
Binary-EC	*Binary* w/ error-correction
Hybrid-N-EC	*Hybrid-N* w/ error-correction

4 Experiments

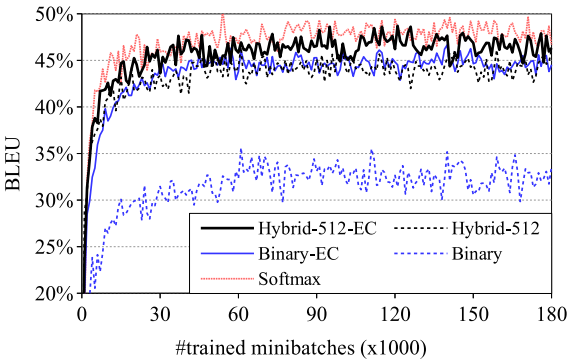
4.1 Experimental Settings

We examined the performance of the proposed methods on two English-Japanese bidirectional translation tasks which have different translation difficulties: ASPEC (Nakazawa et al., 2016) and BTEC (Takezawa, 1999). Table 1 describes details of two corpora. To prepare inputs for training, we used `tokenizer.perl` in Moses (Koehn et al., 2007) and KyTea (Neubig et al., 2011) for English/Japanese tokenizations respectively, applied `lowercase.perl` from Moses, and replaced out-of-vocabulary words such that $\mathrm{rank}(w) > V - 3$ into the `UNK` marker.

We implemented each NMT model using C++ in the DyNet framework (Neubig et al., 2017) and trained/tested on 1 GPU (GeForce GTX TITAN X). Each test is also performed on CPUs to compare its processing time. We used a bidirectional RNN-based encoder applied in Bahdanau et al. (2014), unidirectional decoder with the same style of (Luong et al., 2015), and the *concat* global attention model also proposed in Luong et al. (2015). Each recurrent unit is constructed using a 1-layer LSTM (input/forget/output gates and non-peepholes) (Gers et al., 2000) with 30% dropout (Srivastava et al., 2014) for the input/output vectors of the LSTMs. All word embeddings, recurrent states and model-specific hidden states are designed with 512-dimentional vectors. Only output layers and loss functions are replaced, and other network architectures are identical for the conventional/proposed models. We used the Adam optimizer (Kingma and Ba, 2014) with fixed hyper-parameters $\alpha = 0.001, \beta_1 = 0.9\, \beta_2 = 0.999, \varepsilon = 10^{-8}$, and mini-batches with 64 sentences sorted according to their sequence lengths. For evaluating the quality of each model, we calculated case-insensitive BLEU (Papineni et al., 2002) every 1000 mini-batches. Table 2 lists summaries of all methods we examined in experiments.



(a) ASPEC (En → Ja)



(b) BTEC (En → Ja)

Figure 5: Training curves over 180,000 epochs.

4.2 Results and Discussion

Table 3 shows the BLEU on the test set (**bold** and *italic* faces indicate the best and second places in each task), number of bits B (or B') for the binary code, actual size of the output layer $\#_{\mathrm{out}}$, number of parameters in the output layer $\#_{W,\boldsymbol{\beta}}$, as well as the ratio of $\#_{W,\boldsymbol{\beta}}$ or amount of whole parameters compared with *Softmax*, and averaged processing time at training (per mini-batch on GPUs) and test (per sentence on GPUs/CPUs), respectively. Figure 5(a) and 5(b) shows training curves up to 180,000 epochs about some English→Japanese settings. To relax instabilities of translation qualities while training (as shown in Figure 5(a) and 5(b)), each BLEU in Table 3 is calculated by averaging actual test BLEU of 5 consecutive results

Table 3: Comparison of BLEU, size of output layers, number of parameters and processing time.

Corpus	Method	BLEU % EnJa	BLEU % JaEn	B	$\#_{\text{out}}$	$\#_{W,\beta}$	Ratio of #params $\#_{W,\beta}$	Ratio of #params All	Time (En→Ja) [ms] Train	Time (En→Ja) [ms] Test: GPU / CPU
ASPEC	*Softmax*	**31.13**	**21.14**	—	65536	33.6 M	1/1	1	1026.	121.6 / 2539.
	Binary	13.78	6.953	16	16	8.21 k	1/4.10 k	0.698	711.2	73.08 / 122.3
	Hybrid-512	22.81	13.95	16	528	271. k	1/124.	0.700	843.6	81.28 / 127.5
	Hybrid-2048	27.73	16.92	16	2064	1.06 M	1/31.8	0.707	837.1	82.28 / 159.3
	Binary-EC	25.95	18.02	44	44	22.6 k	1/1.49 k	0.698	712.0	78.75 / 164.0
	Hybrid-512-EC	29.07	18.66	44	556	285. k	1/118.	0.700	850.3	80.30 / 180.2
	Hybrid-2048-EC	*30.05*	*19.66*	44	2092	1.07 M	1/31.4	0.707	851.6	77.83 / 201.3
BTEC	*Softmax*	*47.72*	45.22	—	25000	12.8 M	1/1	1	325.0	34.35 / 323.3
	Binary	31.83	31.90	15	15	7.70 k	1/1.67 k	0.738	250.7	27.98 / 54.62
	Hybrid-512	44.23	43.50	15	527	270. k	1/47.4	0.743	300.7	28.83 / 66.13
	Hybrid-2048	46.13	45.76	15	2063	1.06 M	1/12.1	0.759	307.7	28.25 / 67.40
	Binary-EC	44.48	41.21	42	42	21.5 k	1/595.	0.738	255.6	28.02 / 69.76
	Hybrid-512-EC	47.20	*46.52*	42	554	284. k	1/45.1	0.744	307.8	28.44 / 56.98
	Hybrid-2048-EC	**48.17**	**46.58**	42	2090	1.07 M	1/12.0	0.760	311.0	28.47 / 69.44

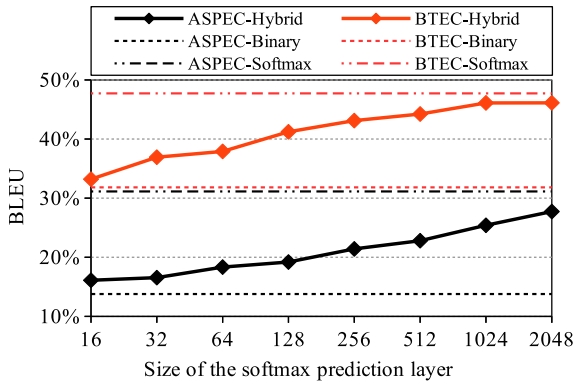


Figure 6: BLEU changes in the *Hybrid-N* methods according to the softmax size (En→Ja).

around the epoch that has the highest dev BLEU.

First, we can see that each proposed method largely suppresses the actual size of the output layer from ten to one thousand times compared with the standard softmax. By looking at the total number of parameters, we can see that the proposed models require only 70% of the actual memory, and the proposed model reduces the total number of parameters for the output layers to a practically negligible level. Note that most of remaining parameters are used for the embedding lookup at the input layer in both encoder/decoder. These still occupy $O(EV)$ memory, where E represents the size of each embedding layer and usually $O(E/H) = O(1)$. These are not targets to be reduced in this study because these values rarely are accessed at test time because we only need to access them for input words, and do not need them to always be in the physical memory. It might be possible to apply a similar binary representation as that of output layers to the input layers as well, then express the word embedding by multiplying this binary vector by a word embedding matrix. This is one potential avenue of future work.

Taking a look at the BLEU for the simple *Binary* method, we can see that it is far lower than other models for all tasks. This is expected, as described in Section 3, because using raw bit arrays causes many one-off estimation errors at the output layer due to the lack of robustness of the output representation. In contrast, *Hybrid-N* and *Binary-EC* models clearly improve BLEU from *Binary*, and they approach that of *Softmax*. This demonstrates that these two methods effectively improve the robustness of binary code prediction models. Especially, *Binary-EC* generally achieves higher quality than *Hybrid-512* despite the fact that it suppress the number of parameters by about 1/10. These results show that introducing redundancy to target bit arrays is more effective than incremental prediction. In addition, the *Hybrid-N-EC* model achieves the highest BLEU in all proposed methods, and in particular, comparative or higher BLEU than *Softmax* in BTEC. This behavior clearly demonstrates that these two methods are orthogonal, and combining them together can be effective. We hypothesize that the lower quality of *Softmax* in BTEC is caused by an over-fitting due to the large number of parameters required in the softmax prediction.

The proposed methods also improve actual computation time in both training and test. In particular on CPU, where the computation speed is directly affected by the size of the output layer, the proposed methods translate significantly faster

than *Softmax* by x5 to x20. In addition, we can also see that applying error-correcting code is also effictive with respect to the decoding speed.

Figure 6 shows the trade-off between the translation quality and the size of softmax layers in the hybrid prediction model (Figure 2(c)) without error-correction. According to the model definition in Section 3.4, the softmax prediction and raw binary code prediction can be assumed to be the upper/lower-bound of the hybrid prediction model. The curves in Figure 6 move between *Softmax* and *Binary* models, and this behavior intuitively explains the characteristics of the hybrid prediction. In addition, we can see that the BLEU score in BTEC quickly improves, and saturates at $N = 1024$ in contrast to the ASPEC model, which is still improving at $N = 2048$. We presume that the shape of curves in Figure 6 is also affected by the difficulty of the corpus, i.e., when we train the hybrid model for easy datasets (e.g., BTEC is easier than ASPEC), it is enough to use a small softmax layer (e.g. $N \leq 1024$).

5 Conclusion

In this study, we proposed neural machine translation models which indirectly predict output words via binary codes, and two model improvements: a hybrid prediction model using both softmax and binary codes, and introducing error-correcting codes to introduce robustness of binary code prediction. Experiments show that the proposed model can achieve comparative translation qualities to standard softmax prediction, while significantly suppressing the amount of parameters in the output layer, and improving calculation speeds while training and especially testing.

One interesting avenue of future work is to automatically learn encodings and error correcting codes that are well-suited for the type of binary code prediction we are performing here. In Algorithms 2 and 3 we use convolutions that were determined heuristically, and it is likely that learning these along with the model could result in improved accuracy or better compression capability.

Acknowledgments

Part of this work was supported by JSPS KAKENHI Grant Numbers JP16H05873 and JP17H00747, and Grant-in-Aid for JSPS Fellows Grant Number 15J10649.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18(4):467–479.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1975–1985. http://www.aclweb.org/anthology/P16-1186.

Rohan Chitnis and John DeNero. 2015. Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2088–2093. http://aclweb.org/anthology/D15-1249.

Thomas G. Dietterich and Ghulum Bakiri. 1995. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2:263–286.

Chun-Sung Ferng and Hsuan-Tien Lin. 2011. Multi-label classification with error-correcting codes. *Journal of Machine Learning Research* 20:281–295.

Chun-Sung Ferng and Hsuan-Tien Lin. 2013. Multi-label classification using error-correcting codes of hard or soft bits. *IEEE transactions on neural networks and learning systems* 24(11):1888–1900.

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

Marcel J. E. Golay. 1949. Notes on digital coding. *Proceedings of the Institute of Radio Engineers* 37:657.

David A. Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers* 40(9):1098–1101.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Aldebaro Klautau, Nikola Jevtić, and Alon Orlitsky. 2003. On nearest-neighbor error-correcting output codes with application to all-pairs multiclass support vector machines. *Journal of Machine Learning Research* 4(April):1–15.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Association for Computational Linguistics, Prague, Czech Republic, pages 177–180. http://www.aclweb.org/anthology/P07-2045.

Abbas Z Kouzani. 2010. Multilabel classification using error correction codes. In *International Symposium on Intelligence Computation and Applications*. Springer, pages 444–454.

Abbas Z Kouzani and Gulisong Nasireding. 2009. Multilabel classification by bch code and random forests. *International journal of recent trends in engineering* 2(1):113–116.

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary selection strategies for neural machine translation. *arXiv preprint arXiv:1610.00072* .

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586* .

Yang Liu. 2006. Using svm and error-correcting codes for multiclass dialog act classification in meeting corpus. In *INTERSPEECH*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Vocabulary manipulation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 124–129. http://anthology.aclweb.org/P16-2021.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of Tenth International Workshop on Artificial Intelligence and Statistics*. volume 5, pages 246–252.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Portoro, Slovenia, pages 2204–2208.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 529–533. http://www.aclweb.org/anthology/P11-2093.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725. http://www.aclweb.org/anthology/P16-1162.

Claude E. Shannon. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27(3):379–423.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Toshiyuki Takezawa. 1999. Building a bilingual travel conversation database for speech translation research. In *Proc. of the 2nd international workshop on East-Asian resources and evaluation conference on language resources and evaluation*. pages 17–20.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory* 13(2):260–269.

George. K. Zipf. 1949. *Human behavior and the principle of least effort.*. Addison-Wesley Press.

What do Neural Machine Translation Models Learn about Morphology?

Yonatan Belinkov[1] **Nadir Durrani**[2] **Fahim Dalvi**[2] **Hassan Sajjad**[2] **James Glass**[1]

[1]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA
{belinkov, glass}@mit.edu
[2]Qatar Computing Research Institute, HBKU, Doha, Qatar
{ndurrani, faimaduddin, hsajjad}@qf.org.qa

Abstract

Neural machine translation (MT) models obtain state-of-the-art performance while maintaining a simple, end-to-end architecture. However, little is known about what these models learn about source and target languages during the training process. In this work, we analyze the representations learned by neural MT models at various levels of granularity and empirically evaluate the quality of the representations for learning morphology through extrinsic part-of-speech and morphological tagging tasks. We conduct a thorough investigation along several parameters: word-based vs. character-based representations, depth of the encoding layer, the identity of the target language, and encoder vs. decoder representations. Our data-driven, quantitative evaluation sheds light on important aspects in the neural MT system and its ability to capture word structure.[1]

1 Introduction

Neural network models are quickly becoming the predominant approach to machine translation (MT). Training neural MT (NMT) models can be done in an end-to-end fashion, which is simpler and more elegant than traditional MT systems. Moreover, NMT systems have become competitive with, or better than, the previous state-of-the-art, especially since the introduction of sequence-to-sequence models and the attention mechanism (Bahdanau et al., 2014; Sutskever et al., 2014). The improved translation quality is often attributed to better handling of non-local dependencies and morphology generation (Luong and Manning, 2015; Bentivogli et al., 2016; Toral and Sánchez-Cartagena, 2017).

However, little is known about what and how much these models learn about each language and its features. Recent work has started exploring the role of the NMT encoder in learning source syntax (Shi et al., 2016), but research studies are yet to answer important questions such as: *(i)* what do NMT models learn about word morphology? *(ii)* what is the effect on learning when translating into/from morphologically-rich languages? *(iii)* what impact do different representations (character vs. word) have on learning? and *(iv)* what do different modules learn about the syntactic and semantic structure of a language? Answering such questions is imperative for fully understanding the NMT architecture. In this paper, we strive towards exploring *(i)*, *(ii)*, and *(iii)* by providing quantitative, data-driven answers to the following specific questions:

- Which parts of the NMT architecture capture word structure?
- What is the division of labor between different components (e.g. different layers or encoder vs. decoder)?
- How do different word representations help learn better morphology and modeling of infrequent words?
- How does the target language affect the learning of word structure?

To achieve this, we follow a simple but effective procedure with three steps: *(i)* train a neural MT system on a parallel corpus; *(ii)* use the trained model to extract feature representations for words in a language of interest; and *(iii)* train a classifier using extracted features to make predictions

[1]Our code is available at https://github.com/boknilev/nmt-repr-analysis.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 861–872
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1080

for another task. We then evaluate the quality of the trained classifier on the given task as a proxy to the quality of the extracted representations. In this way, we obtain a quantitative measure of how well the original MT system learns features that are relevant to the given task.

We focus on the tasks of part-of-speech (POS) and full morphological tagging. We investigate how different neural MT systems capture POS and morphology through a series of experiments along several parameters. For instance, we contrast word-based and character-based representations, use different encoding layers, vary source and target languages, and compare extracting features from the encoder vs. the decoder.

We experiment with several languages with varying degrees of morphological richness: French, German, Czech, Arabic, and Hebrew. Our analysis reveals interesting insights such as:

- Character-based representations are much better for learning morphology, especially for low-frequency words. This improvement is correlated with better BLEU scores. On the other hand, word-based models are sufficient for learning the structure of common words.

- Lower layers of the encoder are better at capturing word structure, while deeper networks improve translation quality, suggesting that higher layers focus more on word meaning.

- The target language impacts the kind of information learned by the MT system. Translating into morphologically-poorer languages leads to better source-side word representations. This is partly, but not completely, correlated with BLEU scores.

- The neural decoder learns very little about word structure. The attention mechanism removes much of the burden of learning word representations from the decoder.

2 Methodology

Given a source sentence $s = \{w_1, w_2, ..., w_N\}$ and a target sentence $t = \{u_1, u_2, ..., u_M\}$, we first generate a vector representation for the source sentence using an encoder (Eqn. 1) and then map this vector to the target sentence using a decoder (Eqn. 2) (Sutskever et al., 2014):

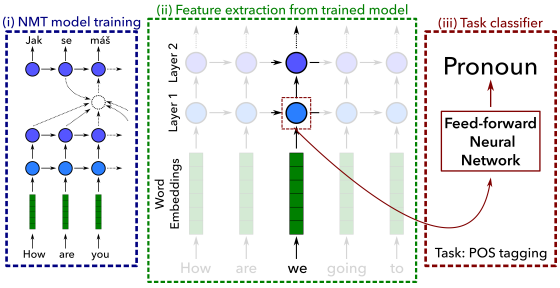


Figure 1: Illustration of our approach: (i) NMT system trained on parallel data; (ii) features extracted from pre-trained model; (iii) classifier trained using the extracted features. Here a POS tagging classifier is trained on features from the first hidden layer.

$$\texttt{ENC} : s = \{w_1, w_2, ..., w_N\} \mapsto \mathbf{s} \in \mathbb{R}^k \quad (1)$$

$$\texttt{DEC} : \mathbf{s} \in \mathbb{R}^k \mapsto t = \{u_1, u_2, ..., u_M\} \quad (2)$$

In this work, we use long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) encoder-decoders with attention (Bahdanau et al., 2014), which we train on parallel data.

After training the NMT system, we freeze the parameters of the encoder and use ENC as a feature extractor to generate vectors representing words in the sentence. Let $\texttt{ENC}_i(s)$ denote the encoded representation of word w_i. For example, this may be the output of the LSTM after word w_i. We feed $\texttt{ENC}_i(s)$ to a neural classifier that is trained to predict POS or morphological tags and evaluate the quality of the representation based on our ability to train a good classifier. By comparing the performance of classifiers trained with features from different instantiations of ENC, we can evaluate what MT encoders learn about word structure. Figure 1 illustrates this process. We follow a similar procedure for analyzing representation learning in DEC.

The classifier itself can be modeled in different ways. For example, it may be an LSTM over outputs of the encoder. However, as we are interested in assessing the quality of the representations learned by the MT system, we choose to model the classifier as a simple feed-forward neural network with one hidden layer and a ReLU non-linearity. Arguably, if the learned representations are good, then a non-linear classifier should be able to extract useful information from them.[2] We empha-

[2]We also experimented with a linear classifier and observed similar trends to the non-linear case, but overall lower results; Qian et al. (2016b) reported similar findings.

	Ar	De	Fr	Cz
	Gold/Pred	Gold/Pred	Pred	Pred
Train Tokens	0.5M/2.7M	0.9M/4M	5.2M	2M
Dev Tokens	63K/114K	45K/50K	55K	35K
Test Tokens	62K/16K	44K/25K	23K	20K
POS Tags	42	54	33	368
Morph Tags	1969	214	–	–

Table 1: Statistics for annotated corpora in Arabic (Ar), German (De), French (Fr), and Czech (Cz).

size that our goal is not to beat the state-of-the-art on a given task, but rather to analyze what NMT models learn about morphology. The classifier is trained with a cross-entropy loss; more details about its architecture are given in the supplementary material (appendix A.1).

3 Data

Language pairs We experiment with several language pairs, including morphologically-rich languages, that have received relatively significant attention in the MT community. These include Arabic-, German-, French-, and Czech-English pairs. To broaden our analysis and study the effect of having morphologically-rich languages on both source and target sides, we also include Arabic-Hebrew, two languages with rich and similar morphological systems, and Arabic-German, two languages with rich but different morphologies.

MT data Our translation models are trained on the WIT[3] corpus of TED talks (Cettolo et al., 2012; Cettolo, 2016) made available for IWSLT 2016. This allows for comparable and cross-linguistic analysis. Statistics about each language pair are given in Table 1 (under Pred). We use official dev and test sets for tuning and testing. Reported figures are the averages over test sets.

Annotated data We use two kinds of datasets to train POS and morphological classifiers: gold-standard and predicted tags. For predicted tags, we simply used freely available taggers to annotate the MT data. For gold tags, we use gold-annotated datasets. Table 1 provides statistics for datasets with gold and predicted tags; see the supplementary material (appendix A.2) for more details about taggers and gold data. We train and test our classifiers on predicted annotations, and similarly on gold annotations, when we have them. We report both results wherever available.

	Gold	Pred	BLEU
	Word/Char	Word/Char	Word/Char
Ar-En	80.31/93.66	89.62/95.35	24.7/28.4
Ar-He	78.20/92.48	88.33/94.66	9.9/10.7
De-En	87.68/94.57	93.54/94.63	29.6/30.4
Fr-En	–	94.61/95.55	37.8/38.8
Cz-En	–	75.71/79.10	23.2/25.4

Table 2: POS accuracy on gold and predicted tags using word-based and character-based representations, as well as corresponding BLEU scores.

4 Encoder Analysis

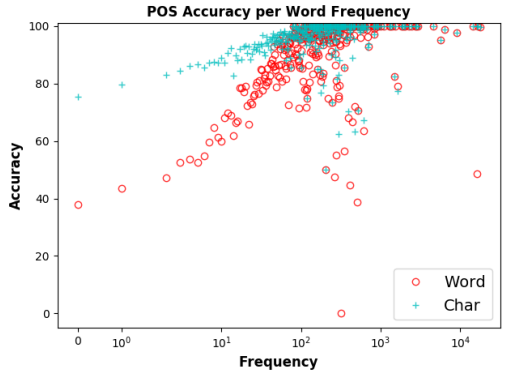
Recall that after training the NMT system we freeze its parameters and use it only to generate features for the POS/morphology classifier. Given a trained encoder $\texttt{ENC}$ and a sentence s with POS/morphology annotation, we generate word features $\texttt{ENC}_i(s)$ for every word in the sentence. We then train a classifier that uses the features $\texttt{ENC}_i(s)$ to predict POS or morphological tags.

4.1 Effect of word representation

In this section, we compare different word representations extracted with different encoders. Our word-based model uses a word embedding matrix which is initialized randomly and learned with other NMT parameters. For a character-based model we adopt a convolutional neural network (CNN) over character embeddings that is also learned during training (Kim et al., 2015); see appendix A.1 for specific settings. In both cases we run the encoder over these representations and use its output $\texttt{ENC}_i(s)$ as features for the classifier.

Table 2 shows POS tagging accuracy using features from different NMT encoders. Char-based models always generate better representations for POS tagging, especially in the case of morphologically-richer languages like Arabic and Czech. We observed a similar pattern in the full morphological tagging task. For example, we obtain morphological tagging accuracy of 65.2/79.66 and 67.66/81.66 using word/char-based representations from the Arabic-Hebrew and Arabic-English encoders, respectively.[3] The superior morphological power of the char-based model also manifests in better translation quality (measured by BLEU), as shown in Table 2.

[3]The results are not far below dedicated taggers (e.g. 95.1/84.1 on Arabic POS/morphology (Pasha et al., 2014)), indicating that NMT models learn quite good representations.



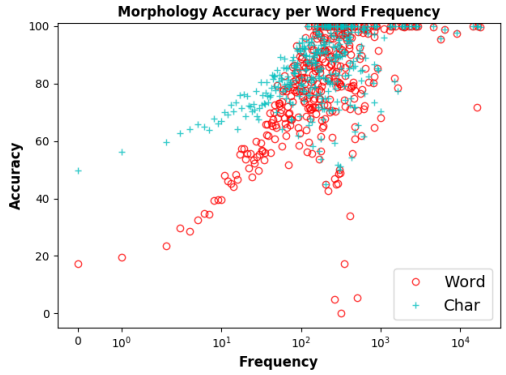


Figure 2: POS and morphological tagging accuracy of word-based and character-based models per word frequency in the training data. Best viewed in color.

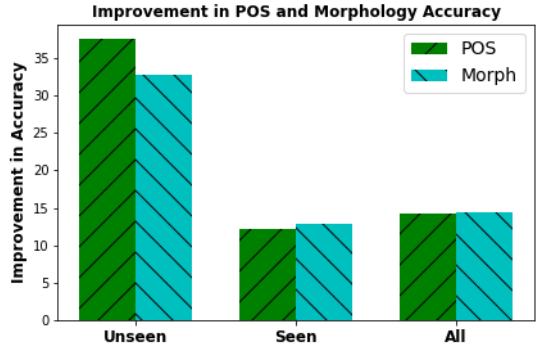


Figure 3: Improvement in POS/morphology accuracy of character-based vs. word-based models for words unseen/seen in training, and for all words.

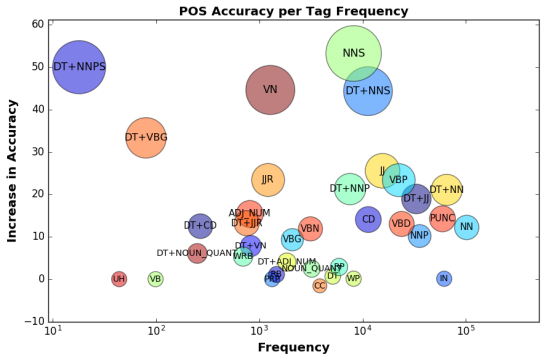
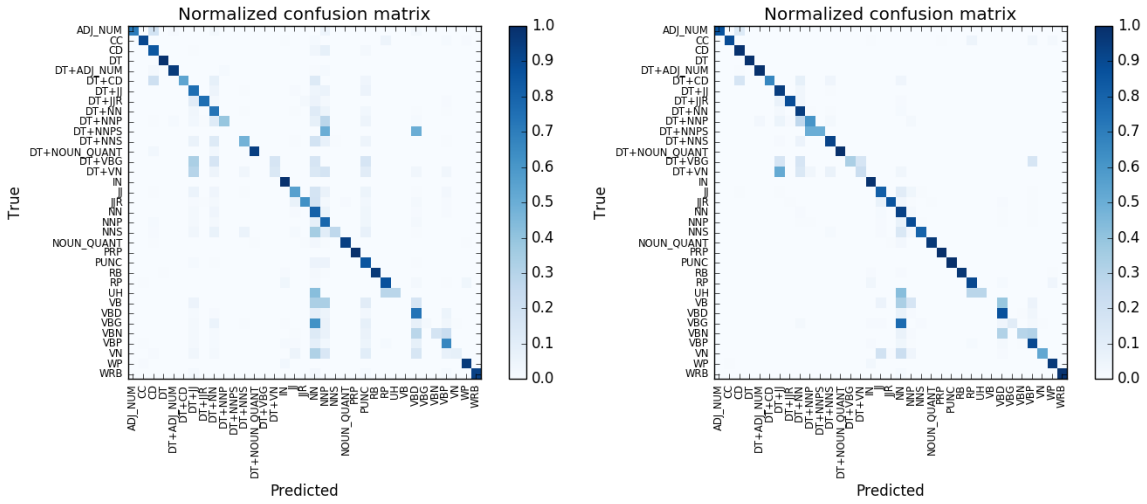


Figure 4: Increase in POS accuracy with char- vs. word-based representations per tag frequency in the training set; larger bubbles reflect greater gaps.

Impact of word frequency Let us look more closely at an example case: Arabic POS and morphological tagging. Figure 3 shows the effect of using word-based vs. char-based feature representations, obtained from the encoder of the Arabic-Hebrew system (other language pairs exhibit similar trends). Clearly, the char-based model is superior to the word-based one. This is true for the overall accuracy (+14.3% in POS, +14.5% in morphology), but more so in OOV words (+37.6% in POS, +32.7% in morphology). Figure 2 shows that the gap between word-based and char-based representations increases as the frequency of the word in the training data decreases. In other words, the more frequent the word, the less need there is for character information. These findings make intuitive sense: the char-based model is able to learn character n-gram patterns that are important for identifying word structure, but as the word becomes more frequent the word-based model has seen enough examples to make a decision.

Analyzing specific tags In Figure 5 we plot confusion matrices for POS tagging using word-based and char-based representations (from Arabic encoders). While the char-based representations are overall better, the two models still share similar misclassified tags. Much of the confusion comes from wrongly predicting nouns (NN, NNP). In the word-based case, relatively many tags with determiner (DT+NNP, DT+NNPS, DT+NNS, DT+VBG) are wrongly predicted as non-determined nouns (NN, NNP). In the char-based case, this hardly happens. This suggests that the char-based representations are predictive of the presence of a determiner, which in Arabic is expressed as the prefix "Al-" (the definite article), a pattern easily captured by a char-based model.

In Figure 4 we plot the difference in POS accuracy when moving from word-based to char-based representations, per POS tag frequency in the training data. Tags closer to the upper-right corner occur more frequently in the training set and are



(a) Word-based representations.

(b) Character-based representations.

Figure 5: Confusion matrices for POS tagging using word-based and character-based representations.

better predicted by char-based compared to word-based representations. There are a few fairly frequent tags (in the middle-bottom part of the figure) whose accuracy does not improve much when moving from word- to char-based representations: mostly conjunctions, determiners, and certain particles (CC, DT, WP). But there are several very frequent tags (NN, DT+NN, DT+JJ, VBP, and even PUNC) whose accuracy improves quite a lot. Then there are plural nouns (NNS, DT+NNS) where the char-based model really shines, which makes sense linguistically as plurality in Arabic is usually expressed by certain suffixes ("-wn/yn" for masc. plural, "-At" for fem. plural). The char-based model is thus especially good with frequent tags and infrequent words, which is understandable given that infrequent words typically belong to frequent open categories like nouns and verbs.

4.2 Effect of encoder depth

Modern NMT systems use very deep architectures with up to 8 or 16 layers (Wu et al., 2016; Zhou et al., 2016). We would like to understand what kind of information different layers capture. Given a trained NMT model with multiple layers, we extract feature representations from the different layers in the encoder. Let $\texttt{ENC}_i^l(s)$ denote the encoded representation of word w_i after the l-th layer. We can vary l and train different classifiers to predict POS or morphological tags. Here we focus on the case of a 2-layer encoder-decoder model for simplicity ($l \in \{1, 2\}$).

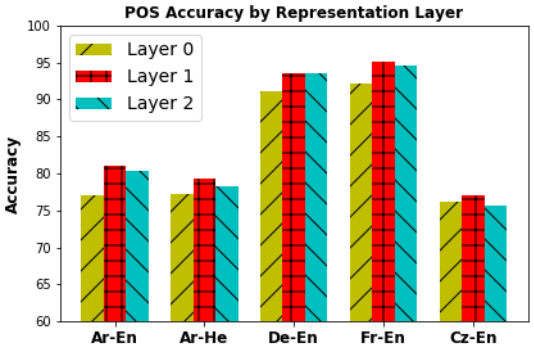


Figure 6: POS tagging accuracy using representations from layers 0 (word vectors), 1, and 2, taken from encoders of different language pairs.

Figure 6 shows POS tagging results using representations from different encoding layers across five language pairs. The general trend is that passing word vectors through the NMT encoder improves POS tagging, which can be explained by the contextual information contained in the representations after one layer. However, it turns out that representations from the 1st layer are better than those from the 2nd layer, at least for the purpose of capturing word structure. Figure 7 demonstrates that the same pattern holds for both word-based and char-based representations, on Arabic POS and morphological tagging. In all cases, layer 1 representations are better than layer 2 representations.[4] In contrast, BLEU scores ac-

[4] We found this result to be also true in French, German, and Czech experiments; see appendix A.3.

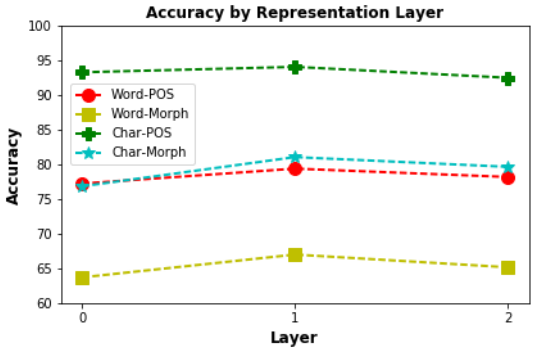


Figure 7: POS and morphological tagging accuracy across layers. Layer 0: word vectors or char-based representations before the encoder; layers 1 and 2: representations after the 1st and 2nd layers.

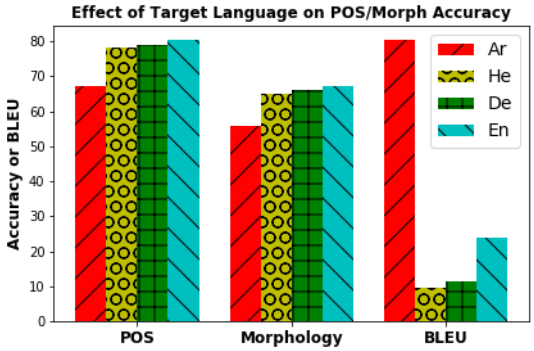


Figure 8: Effect of target language on representation quality of the Arabic source.

tually increase when training 2-layer vs. 1-layer models (+1.11/+0.56 BLEU for Arabic-Hebrew word/char-based models). Thus translation quality improves when adding layers but morphology quality degrades. Intuitively, it seems that lower layers of the network learn to represent word structure while higher layers are more focused on word meaning. A similar pattern was recently observed in a joint language-vision deep recurrent network (Gelderloos and Chrupała, 2016).

4.3 Effect of target language

While translating from morphologically-rich languages is challenging, translating into such languages is even harder. For instance, our basic system obtains BLEU scores of 24.69/23.2 on Arabic/Czech to English, but only 13.37/13.9 on English to Arabic/Czech. How does the target language affect the learned source language representations? Does translating into a morphologically-rich language require more knowledge about source language morphology? In order to investigate these questions, we fix the source language and train NMT models using different target languages. For example, given an Arabic source side, we train Arabic-to-English/Hebrew/German systems. These target languages represent a morphologically-poor language (English), a morphologically-rich language with similar morphology to the source language (Hebrew), and a morphologically-rich language with different morphology (German). To make a fair comparison, we train the models on the intersection of the training data based on the source language. In this way the experimental setup is completely identical: the models are trained on the same Arabic sentences with different translations.

Figure 8 shows POS and morphological tagging accuracy of word-based representations from the NMT encoders, as well as corresponding BLEU scores. As expected, translating into English is easier than translating into the morphologically-richer Hebrew and German, resulting in higher BLEU scores. Despite their similar morphological systems, translating Arabic to Hebrew is worse than Arabic to German, which can be attributed to the richer Hebrew morphology compared to German. POS and morphology accuracies share an intriguing pattern: the representations that are learned when translating into English are better for predicting POS or morphology than those learned when translating into German, which are in turn better than those learned when translating into Hebrew. This is remarkable given that English is a morphologically-poor language that does not display many of the morphological properties that are found in the Arabic source. In contrast, German and Hebrew have richer morphologies, so one could expect that translating into them would make the model learn more about morphology.

A possible explanation for this phenomenon is that the Arabic-English model is simply better than the Arabic-Hebrew and Arabic-German models, as hinted by the BLEU scores in Table 2. The inherent difficulty in translating Arabic to Hebrew/German may affect the ability to learn good representations of word structure. To probe this more, we trained an Arabic-Arabic autoencoder on the same training data. We found that it learns to recreate the test sentences extremely well, with very high BLEU scores (Figure 8). However, its

word representations are actually inferior for the purpose of POS/morphological tagging. This implies that higher BLEU does not necessarily entail better morphological representations. In other words, a better translation model learns more informative representations, but only when it is actually learning to translate rather than merely memorizing the data as in the autoencoder case. We found this to be consistently true also for char-based experiments, and in other language pairs.

5 Decoder Analysis

So far we only looked at the encoder. However, the decoder DEC is a crucial part in an MT system with access to both source and target sentences. In order to examine what the decoder learns about morphology, we first train an NMT system on the parallel corpus. Then, we use the trained model to encode a source sentence and extract features for words in the target sentence. These features are used to train a classifier on POS or morphological tagging on the target side.[5] Note that in this case the decoder is given the correct target words one-by-one, similar to the usual NMT training regime.

Table 3 (1st row) shows the results of using representations extracted with ENC and DEC from the Arabic-English and English-Arabic models, respectively. There is clearly a huge drop in representation quality with the decoder.[6] At first, this drop seems correlated with lower BLEU scores in English to Arabic vs. Arabic to English. However, we observed similar low POS tagging accuracy using decoder representations from high-quality NMT models. For instance, the French-to-English system obtains 37.8 BLEU, but its decoder representations give a mere 54.26% accuracy on English POS tagging.

As an alternative explanation for the poor quality of the decoder representations, consider the fundamental tasks of the two NMT modules: encoder and decoder. The encoder's task is to create a generic, close to language-independent representation of the source sentence, as shown by recent evidence from multilingual NMT (Johnson et al., 2016). The decoder's task is to use this representation to generate the target sentence in a specific language. Presumably, it is sufficient for the decoder to learn a strong language model in order to produce morphologically-correct output, without learning much about morphology, while the encoder needs to learn quite a lot about source language morphology in order to create a good generic representation. In the following section we show that the attention mechanism also plays an important role in the division of labor between encoder and decoder.

[5] In this section we only experiment with predicted tags as there are no parallel data with gold POS/morphological tags that we are aware of.

[6] Note that the decoder results are above a majority baseline of 20%, so the decoder is still learning something about the target language.

Attn	POS Accuracy ENC	POS Accuracy DEC	BLEU Ar-En	BLEU En-Ar
✓	89.62	43.93	24.69	13.37
✗	74.10	50.38	11.88	5.04

Table 3: POS tagging accuracy using encoder and decoder representations with/without attention.

5.1 Effect of attention

Consider the role of the attention mechanism in learning useful representations: during decoding, the attention weights are combined with the decoder's hidden states to generate the current translation. These two sources of information need to jointly point to the most relevant source word(s) and predict the next most likely word. Thus, the decoder puts significant emphasis on mapping back to the source sentence, which may come at the expense of obtaining a meaningful representation of the current word. We hypothesize that the attention mechanism hurts the quality of the target word representations learned by the decoder.

To test this hypothesis, we train NMT models with and without attention and compare the quality of their learned representations. As Table 3 shows (compare 1st and 2nd rows), removing the attention mechanism decreases the quality of the encoder representations, but improves the quality of the decoder representations. Without the attention mechanism, the decoder is forced to learn more informative representations of the target language.

5.2 Effect of word representation

We also conducted experiments to verify our findings regarding word-based versus character-based representations on the decoder side. By character representation we mean a character CNN on the input words. The decoder predictions are still done at the word-level, which enables us to use its hidden states as word representations.

Table 4 shows POS accuracy of word-based vs. char-based representations in the encoder and decoder. While char-based representations improve the encoder, they do not help the decoder. BLEU scores behave similarly: the char-based model leads to better translations in Arabic-to-English, but not in English-to-Arabic. A possible explanation for this phenomenon is that the decoder's predictions are still done at word level even with the char-based model (which encodes the target input but not the output). In practice, this can lead to generating unknown words. Indeed, in Arabic-to-English the char-based model reduces the number of generated unknown words in the MT test set by 25%, while in English-to-Arabic the number of unknown words remains roughly the same between word-based and char-based models.

	POS Accuracy		BLEU	
	ENC	DEC	Ar-En	En-Ar
Word	89.62	43.93	24.69	13.37
Char	95.35	44.54	28.42	13.00

Table 4: POS tagging accuracy using word-based and char-based encoder/decoder representations.

6 Related Work

Analysis of neural models The opacity of neural networks has motivated researchers to analyze such models in different ways. One line of work visualizes hidden unit activations in recurrent neural networks that are trained for a given task (Elman, 1991; Karpathy et al., 2015; Kádár et al., 2016; Qian et al., 2016a). While such visualizations illuminate the inner workings of the network, they are often qualitative in nature and somewhat anecdotal. A different approach tries to provide a quantitative analysis by correlating parts of the neural network with linguistic properties, for example by training a classifier to predict features of interest. Different units have been used, from word embeddings (Köhn, 2015; Qian et al., 2016b), through LSTM gates or states (Qian et al., 2016a), to sentence embeddings (Adi et al., 2016). Our work is most similar to Shi et al. (2016), who use hidden vectors from a neural MT encoder to predict syntactic properties on the English source side. In contrast, we focus on representations in morphologically-rich languages and evaluate both source and target sides across several criteria. Vylomova et al. (2016) also analyze different representations for morphologically-rich languages in MT, but do not directly measure the quality of the learned representations.

Word representations in MT Machine translation systems that deal with morphologically-rich languages resort to various techniques for representing morphological knowledge, such as word segmentation (Nieflen and Ney, 2000; Koehn and Knight, 2003; Badr et al., 2008) and factored translation and reordering models (Koehn and Hoang, 2007; Durrani et al., 2014). Characters and other sub-word units have become increasingly popular in neural MT, although they had also been used in phrase-based MT for handling morphologically-rich (Luong et al., 2010) or closely related language pairs (Durrani et al., 2010; Nakov and Tiedemann, 2012). In neural MT, such units are obtained in a pre-processing step—e.g. by byte-pair encoding (Sennrich et al., 2016) or the word-piece model (Wu et al., 2016)—or learned during training using a character-based convolutional/recurrent sub-network (Costa-jussà and Fonollosa, 2016; Luong and Manning, 2016; Vylomova et al., 2016). The latter approach has the advantage of keeping the original word boundaries without requiring pre- and post-processing. Here we focus on a character CNN which has been used in language modeling and machine translation (Kim et al., 2015; Belinkov and Glass, 2016; Costa-jussà and Fonollosa, 2016; Jozefowicz et al., 2016; Sajjad et al., 2017). We evaluate the quality of different representations learned by an MT system augmented with a character CNN in terms of POS and morphological tagging, and contrast them with a purely word-based system.

7 Conclusion

Neural networks have become ubiquitous in machine translation due to their elegant architecture and good performance. The representations they use for linguistic units are crucial for obtaining high-quality translation. In this work, we investigated how neural MT models learn word structure. We evaluated their representation quality on POS and morphological tagging in a number of languages. Our results lead to the following conclusions:

- Character-based representations are better than word-based ones for learning morphology, especially in rare and unseen words.

- Lower layers of the neural network are better at capturing morphology, while deeper networks improve translation performance. We hypothesize that lower layers are more focused on word structure, while higher ones are focused on word meaning.

- Translating into morphologically-poorer languages leads to better source-side representations. This is partly, but not completely, correlated with BLEU scores.

- The attentional decoder learns impoverished representations that do not carry much information about morphology.

These insights can guide further development of neural MT systems. For instance, jointly learning translation and morphology can possibly lead to better representations and improved translation. Our analysis indicates that this kind of approach should take into account factors such as the encoding layer and the type of word representation.

Another area for future work is to extend the analysis to other word representations (e.g. byte-pair encoding), deeper networks, and more semantically-oriented tasks such as semantic role-labeling or semantic parsing.

Acknowledgments

We would like to thank Helmut Schmid for providing the Tiger corpus, members of the MIT Spoken Language Systems group for helpful comments, and the three anonymous reviewers for their useful suggestions. This research was carried out in collaboration between the HBKU Qatar Computing Research Institute (QCRI) and the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL).

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks. *arXiv preprint arXiv:1608.04207* .

Ibrahim Badr, Rabih Zbib, and James Glass. 2008. Segmentation for English-to-Arabic Statistical Machine Translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*. Columbus, Ohio, HLT-Short '08, pages 153–156. http://dl.acm.org/citation.cfm?id=1557690.1557732.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473* .

Yonatan Belinkov and James Glass. 2016. Large-Scale Machine Translation between Arabic and Hebrew: Available Corpora and Initial Results. In *Proceedings of the Workshop on Semitic Machine Translation*. Association for Computational Linguistics, Austin, Texas, pages 7–12.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus Phrase-Based Machine Translation Quality: a Case Study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 257–267. https://aclweb.org/anthology/D16-1025.

Mauro Cettolo. 2016. An Arabic-Hebrew parallel corpus of TED talks. In *Proceedings of the Workshop on Semitic Machine Translation*. Association for Computational Linguistics, Austin, Texas, pages 1–6.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT^3: Web Inventory of Transcribed and Translated Talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*. Trento, Italy, pages 261–268.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 357–361. http://anthology.aclweb.org/P16-2058.

Nadir Durrani, Philipp Koehn, Helmut Schmid, and Alexander Fraser. 2014. Investigating the Usefulness of Generalized Word Representations in SMT. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 421–432. http://www.aclweb.org/anthology/C14-1041.

Nadir Durrani, Hassan Sajjad, Alexander Fraser, and Helmut Schmid. 2010. Hindi-to-Urdu Machine Translation through Transliteration. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 465–474. http://www.aclweb.org/anthology/P10-1048.

Jeffrey L Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning* 7(2-3):195–225.

Lieke Gelderloos and Grzegorz Chrupała. 2016. From phonemes to images: levels of representation in a recurrent neural model of visually-grounded language learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1309–1319. http://aclweb.org/anthology/C16-1124.

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

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *arXiv preprint arXiv:1611.04558* .

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the Limits of Language Modeling. *arXiv preprint arXiv:1602.02410* .

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2016. Representation of linguistic form and function in recurrent neural networks. *arXiv preprint arXiv:1602.08952* .

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and Understanding Recurrent Networks. *arXiv preprint arXiv:1506.02078* .

Yoon Kim. 2016. Seq2seq-attn. `https://github.com/harvardnlp/seq2seq-attn`.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware Neural Language Models. *arXiv preprint arXiv:1508.06615* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* .

Philipp Koehn and Hieu Hoang. 2007. Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Association for Computational Linguistics, Prague, Czech Republic, pages 868–876. http://www.aclweb.org/anthology/D07-1091.

Philipp Koehn and Kevin Knight. 2003. Empirical Methods for Compound Splitting. In *10th Conference of the European Chapter of the Association for Computational Linguistics*. pages 187–194. http://www.aclweb.org/anthology/E03-1076.

Arne Köhn. 2015. What's in an Embedding? Analyzing Word Embeddings through Multilingual Evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2067–2073. http://aclweb.org/anthology/D15-1246.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domains. In *Proceedings of the International Workshop on Spoken Language Translation*. Da Nang, Vietnam.

Minh-Thang Luong and D. Christopher Manning. 2016. Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1054–1063. https://doi.org/10.18653/v1/P16-1100.

Minh-Thang Luong, Preslav Nakov, and Min-Yen Kan. 2010. A Hybrid Morpheme-Word Representation for Machine Translation of Morphologically Rich Languages. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 148–157. http://aclweb.org/anthology/D10-1015.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient Higher-Order CRFs for Morphological Tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 322–332. http://www.aclweb.org/anthology/D13-1032.

Preslav Nakov and Jörg Tiedemann. 2012. Combining Word-Level and Character-Level Models for Machine Translation Between Closely-Related Languages. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Jeju, Korea, ACL '12, pages 301–305. http://aclweb.org/anthology/P12-2059.

Sonja Nieflen and Hermann Ney. 2000. Improving SMT quality with morpho-syntactic analysis. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*. http://www.aclweb.org/anthology/C00-2162.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland, pages 1094–1101.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016a. Analyzing Linguistic Knowledge in Sequential Model of Sentence. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 826–835. https://aclweb.org/anthology/D16-1079.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016b. Investigating Language Universal and Specific Properties in Word Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1478–1488. http://www.aclweb.org/anthology/P16-1140.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, Ahmed Abdelali, Yonatan Belinkov, and Stephan Vogel. 2017. Challenging Language-Dependent Segmentation for Arabic: An Application to Machine Translation and Part-of-Speech Tagging. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada.

Helmut Schmid. 1994. Part-of-Speech Tagging with Neural Networks. In *Proceedings of the 15th International Conference on Computational Linguistics (Coling 1994)*. Coling 1994 Organizing Committee, Kyoto, Japan, pages 172–176.

Helmut Schmid. 2000. LoPar: Design and Implementation. Bericht des Sonderforschungsbereiches "Sprachtheoretische Grundlagen fr die Computerlinguistik" 149, Institute for Computational Linguistics, University of Stuttgart.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725. http://www.aclweb.org/anthology/P16-1162.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does String-Based Neural MT Learn Source Syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1526–1534. https://aclweb.org/anthology/D16-1159.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in neural information processing systems*. pages 3104–3112.

Antonio Toral and Víctor M. Sánchez-Cartagena. 2017. A Multifaceted Evaluation of Neural versus Phrase-Based Machine Translation for 9 Language Directions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 1063–1073. http://aclweb.org/anthology/E17-1100.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2016. Word Representation Models for Morphologically Rich Languages in Neural Machine Translation. *arXiv preprint arXiv:1606.04217* .

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* .

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation. *Transactions of the Association for Computational Linguistics* 4:371–383. https://transacl.org/ojs/index.php/tacl/article/view/863.

A Supplementary Material

A.1 Training Details

POS/Morphological classifier The classifier used for all prediction tasks is a feed-forward network with one hidden layer, dropout ($\rho = 0.5$), a ReLU non-linearity, and an output layer mapping to the tag set (followed by a Softmax). The size of the hidden layer is set to be identical to the size of the encoder's hidden state (typically 500 dimensions). We use Adam (Kingma and Ba, 2014) with default parameters to minimize the cross-entropy objective. Training is run with mini-batches of size 16 and stopped once the loss on the dev set stops improving; we allow a patience of 5 epochs.

Neural MT system We train a 2-layer LSTM encoder-decoder with attention. We use the `seq2seq-attn` implementation (Kim, 2016) with the following default settings: word vectors and LSTM states have 500 dimensions, SGD with initial learning rate of 1.0 and rate decay of 0.5, and dropout rate of 0.3. The character-based model is a CNN with a highway network over characters (Kim et al., 2015) with 1000 feature maps and a kernel width of 6 characters. This model was found to be useful for translating morphologically-rich languages (Costa-jussà and Fonollosa, 2016). The MT system is trained for 20 epochs, and the model with the best dev loss is used for extracting features for the classifier.

A.2 Data and Taggers

Datasets All of the translation models are trained on the Ted talks corpus included in WIT^3 (Cettolo et al., 2012; Cettolo, 2016). Statistics about each language pair are available on the WIT^3 website: `https://wit3.fbk.eu`. For experiments using gold tags, we used the Arabic Treebank for Arabic (with the versions and splits described in the MADAMIRA manual (Pasha et al., 2014)) and the Tiger corpus for German.[7]

POS and morphological taggers We used the following tools to annotate the MT corpora: MADAMIRA (Pasha et al., 2014) for Arabic POS and morphological tags, Tree-Tagger (Schmid, 1994) for Czech and French POS tags, LoPar (Schmid, 2000) for German POS and morphological tags, and MXPOST (Ratnaparkhi, 1998) for English POS tags. These tools are recommended on the Moses website.[8] As mentioned before, our goal is not to achieve state-of-the-art results, but rather to study what different components of the NMT architecture learn about word morphology. Please refer to Mueller et al. (2013) for representative POS and morphological tagging accuracies.

A.3 Supplementary Results

We report here results that were omitted from the paper due to the space limit. Table 5 shows encoder results using different layers, languages, and representations (word/char-based). As noted in the paper, all the results consistently show that i) layer 1 performs better than layers 0 and 2; and ii) char-based representations are better than word-based for learning morphology. Table 6 shows that translating into a morphologically-poor language (English) leads to better source representations, and Table 7 provides additional decoder results.

	Layer 0	Layer 1	Layer 2
	Word/Char (POS)		
De	91.1/92.0	93.6/95.2	93.5/94.6
Fr	92.1/92.9	95.1/95.9	94.6/95.6
Cz	76.3/78.3	77.0/79.1	75.7/80.6
	Word/Char (Morphology)		
De	87.6/88.8	89.5/91.2	88.7/90.5

Table 5: POS and morphology accuracy on predicted tags using word- and char-based representations from different layers of *-to-En systems.

Source \ Target	English	Arabic	Self
German	93.5	92.7	89.3
Czech	75.7	75.2	71.8

Table 6: Impact of changing the target language on POS tagging accuracy. Self = German/Czech in rows 1/2 respectively.

	En-De	En-Cz	De-En	Fr-En
POS	53.6	36.3	53.3	54.1
BLEU	23.4	13.9	29.6	37.8

Table 7: POS accuracy and BLEU using decoder representations from different language pairs.

[7] `http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.html`

[8] `http://www.statmt.org/moses/?n=Moses.ExternalTools`

Context-Dependent Sentiment Analysis in User-Generated Videos

Soujanya Poria
Temasek Laboratories
NTU, Singapore
sporia@ntu.edu.sg

Erik Cambria
School of Computer Science and Engineering, NTU, Singapore
cambria@ntu.edu.sg

Devamanyu Hazarika
Computer Science and Engineering, NITW, India
devamanyu@sentic.net

Navonil Mazumder
Centro de Investigacin en Computacin, IPN, Mexico
navonil@sentic.net

Amir Zadeh
Language Technologies Institute, CMU, USA
abagherz@cs.cmu.edu

Louis-Philippe Morency
Language Technologies Institute, CMU, USA
morency@cs.cmu.edu

Abstract

Multimodal sentiment analysis is a developing area of research, which involves the identification of sentiments in videos. Current research considers utterances as independent entities, i.e., ignores the interdependencies and relations among the utterances of a video. In this paper, we propose a LSTM-based model that enables utterances to capture contextual information from their surroundings in the same video, thus aiding the classification process. Our method shows 5-10% performance improvement over the state of the art and high robustness to generalizability.

1 Introduction

Sentiment analysis is a 'suitcase' research problem that requires tackling many NLP sub-tasks, e.g., aspect extraction (Poria et al., 2016a), named entity recognition (Ma et al., 2016), concept extraction (Rajagopal et al., 2013), sarcasm detection (Poria et al., 2016b), personality recognition (Majumder et al., 2017), and more.

Sentiment analysis can be performed at different granularity levels, e.g., subjectivity detection simply classifies data as either subjective (opinionated) or objective (neutral), while polarity detection focuses on determining whether subjective data indicate positive or negative sentiment. Emotion recognition further breaks down the inferred polarity into a set of emotions conveyed by the subjective data, e.g., positive sentiment can be caused by joy or anticipation, while negative sentiment can be caused by fear or disgust.

Even though the primary focus of this paper is to classify sentiment in videos, we also show the performance of the proposed method for the finer-grained task of emotion recognition.

Emotion recognition and sentiment analysis have become a new trend in social media, helping users and companies to automatically extract the opinions expressed in user-generated content, especially videos. Thanks to the high availability of computers and smartphones, and the rapid rise of social media, consumers tend to record their reviews and opinions about products or films and upload them on social media platforms, such as YouTube and Facebook. Such videos often contain comparisons, which can aid prospective buyers make an informed decision.

The primary advantage of analyzing videos over text is the surplus of behavioral cues present in vocal and visual modalities. The vocal modulations and facial expressions in the visual data, along with textual data, provide important cues to better identify affective states of the opinion holder. Thus, a combination of text and video data helps to create a more robust emotion and sentiment analysis model (Poria et al., 2017a).

An utterance (Olson, 1977) is a unit of speech bound by breathes or pauses. Utterance-level sentiment analysis focuses on tagging every utterance of a video with a sentiment label (instead of assigning a unique label to the whole video). In particular, utterance-level sentiment analysis is useful to understand the sentiment dynamics of different aspects of the topics covered by the speaker throughout his/her speech.

Recently, a number of approaches to multimodal sentiment analysis, producing interesting results, have been proposed (Pérez-Rosas et al., 2013; Wollmer et al., 2013; Poria et al., 2015). However, there are major issues that remain unaddressed. Not considering the relation and dependencies among the utterances is one of such issues. State-of-the-art approaches in this area treat utterances independently and ignore the order of utterances in a video (Cambria et al., 2017b).

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 873–883
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1081

Every utterance in a video is spoken at a distinct time and in a particular order. Thus, a video can be treated as a sequence of utterances. Like any other sequence classification problem (Collobert et al., 2011), sequential utterances of a video may largely be contextually correlated and, hence, influence each other's sentiment distribution. In our paper, we give importance to the order in which utterances appear in a video.

We treat surrounding utterances as the context of the utterance that is aimed to be classified. For example, the MOSI dataset (Zadeh et al., 2016) contains a video, in which a girl reviews the movie 'Green Hornet'. At one point, she says "The Green Hornet did something similar". Normally, doing something similar, i.e., monotonous or repetitive might be perceived as negative. However, the nearby utterances "It engages the audience more", "they took a new spin on it", "and I just loved it" indicate a positive context.

The hypothesis of the independence of tokens is quite popular in information retrieval and data mining, e.g., bag-of-words model, but it has a lot limitations (Cambria and White, 2014). In this paper, we discard such an oversimplifying hypothesis and develop a framework based on long short-term memory (LSTM) that takes a sequence of utterances as input and extracts contextual utterance-level features.

The other uncovered major issues in the literature are the role of speaker-dependent versus speaker-independent models, the impact of each modality across the dataset, and generalization ability of a multimodal sentiment classifier. Leaving these issues unaddressed has presented difficulties in effective comparison of different multimodal sentiment analysis methods. In this work, we address all of these issues.

Our model preserves the sequential order of utterances and enables consecutive utterances to share information, thus providing contextual information to the utterance-level sentiment classification process. Experimental results show that the proposed framework has outperformed the state of the art on three benchmark datasets by 5-10%.

The paper is organized as follows: Section 2 provides a brief literature review on multimodal sentiment analysis; Section 3 describes the proposed method in detail; experimental results and discussion are shown in Section 4; finally, Section 5 concludes the paper.

2 Related Work

The opportunity to capture people's opinions has raised growing interest both within the scientific community, for the new research challenges, and in the business world, due to the remarkable benefits to be had from financial market prediction.

Text-based sentiment analysis systems can be broadly categorized into knowledge-based and statistics-based approaches (Cambria et al., 2017a). While the use of knowledge bases was initially more popular for the identification of polarity in text (Cambria et al., 2016; Poria et al., 2016c), sentiment analysis researchers have recently been using statistics-based approaches, with a special focus on supervised statistical methods (Socher et al., 2013; Oneto et al., 2016).

In 1974, Ekman (Ekman, 1974) carried out extensive studies on facial expressions which showed that universal facial expressions are able to provide sufficient clues to detect emotions. Recent studies on speech-based emotion analysis (Datcu and Rothkrantz, 2008) have focused on identifying relevant acoustic features, such as fundamental frequency (pitch), intensity of utterance, bandwidth, and duration.

As for fusing audio and visual modalities for emotion recognition, two of the early works were (De Silva et al., 1997) and (Chen et al., 1998). Both works showed that a bimodal system yielded a higher accuracy than any unimodal system. More recent research on audio-visual fusion for emotion recognition has been conducted at either feature level (Kessous et al., 2010) or decision level (Schuller, 2011). While there are many research papers on audio-visual fusion for emotion recognition, only a few have been devoted to multimodal emotion or sentiment analysis using textual clues along with visual and audio modalities. (Wollmer et al., 2013) and (Rozgic et al., 2012) fused information from audio, visual, and textual modalities to extract emotion and sentiment.

Poria et al. (Poria et al., 2015, 2016d, 2017b) extracted audio, visual and textual features using convolutional neural network (CNN); concatenated those features and employed multiple kernel learning (MKL) for final sentiment classification. (Metallinou et al., 2008) and (Eyben et al., 2010a) fused audio and textual modalities for emotion recognition. Both approaches relied on a feature-level fusion. (Wu and Liang, 2011) fused audio and textual clues at decision level.

3 Method

In this work, we propose a LSTM network that takes as input the sequence of utterances in a video and extracts contextual unimodal and multimodal features by modeling the dependencies among the input utterances. M number of videos, comprising of its constituent utterances, serve as the input. We represent the dataset as $U = u_1, u_2, u_3..., u_M$ and each $u_i = u_{i,1}, u_{i,2}, ..., u_i, L_i$ where L_i is the number of utterances in video u_i. Below, we present an overview of the proposed method in two major steps.

A. Context-Independent Unimodal Utterance-Level Feature Extraction

Firstly, the unimodal features are extracted without considering the contextual information of the utterances (Section 3.1).

B. Contextual Unimodal and Multimodal Classification

Secondly, the context-independent unimodal features (from Step A) are fed into a LSTM network (termed contextual LSTM) that allows consecutive utterances in a video to share information in the feature extraction process (Section 3.2).

We experimentally show that this proposed framework improves the performance of utterance-level sentiment classification over traditional frameworks.

3.1 Extracting Context-Independent Unimodal Features

Initially, the unimodal features are extracted from each utterance separately, i.e., we do not consider the contextual relation and dependency among the utterances. Below, we explain the textual, audio, and visual feature extraction methods.

3.1.1 text-CNN: Textual Features Extraction

The source of textual modality is the transcription of the spoken words. For extracting features from the textual modality, we use a CNN (Karpathy et al., 2014). In particular, we first represent each utterance as the concatenation of vectors of the constituent words. These vectors are the publicly available 300-dimensional word2vec vectors trained on 100 billion words from Google News (Mikolov et al., 2013).

The convolution kernels are thus applied to these concatenated word vectors instead of individual words. Each utterance is wrapped to a window of 50 words which serves as the input to the CNN. The CNN has two convolutional layers; the first layer has two kernels of size 3 and 4, with 50 feature maps each and the second layer has a kernel of size 2 with 100 feature maps.

The convolution layers are interleaved with max-pooling layers of window 2×2. This is followed by a fully connected layer of size 500 and softmax output. We use a rectified linear unit (ReLU) (Teh and Hinton, 2001) as the activation function. The activation values of the fully-connected layer are taken as the features of utterances for text modality. The convolution of the CNN over the utterance learns abstract representations of the phrases equipped with implicit semantic information, which with each successive layer spans over increasing number of words and ultimately the entire utterance.

3.1.2 openSMILE: Audio Feature Extraction

Audio features are extracted at 30 Hz frame-rate and a sliding window of 100 ms. To compute the features, we use openSMILE (Eyben et al., 2010b), an open-source software that automatically extracts audio features such as pitch and voice intensity. Voice normalization is performed and voice intensity is thresholded to identify samples with and without voice. Z-standardization is used to perform voice normalization.

The features extracted by openSMILE consist of several low-level descriptors (LLD), e.g., MFCC, voice intensity, pitch, and their statistics, e.g., mean, root quadratic mean, etc. Specifically, we use IS13-ComParE configuration file in openSMILE. Taking into account all functionals of each LLD, we obtained 6373 features.

3.1.3 3D-CNN: Visual Feature Extraction

We use 3D-CNN (Ji et al., 2013) to obtain visual features from the video. We hypothesize that 3D-CNN will not only be able to learn relevant features from each frame, but will also learn the changes among given number of consecutive frames.

In the past, 3D-CNN has been successfully applied to object classification on tridimensional data (Ji et al., 2013). Its ability to achieve state-of-the-art results motivated us to adopt it in our framework.

Let $vid \in \mathbb{R}^{c \times f \times h \times w}$ be a video, where c = number of channels in an image (in our case $c = 3$, since we consider only RGB images), f = number of frames, h = height of the frames, and w = width of the frames. Again, we consider the 3D convolutional filter $filt \in \mathbb{R}^{f_m \times c \times f_d \times f_h \times f_w}$, where f_m = number of feature maps, c = number of channels, f_d = number of frames (in other words depth of the filter), f_h = height of the filter, and f_w = width of the filter. Similar to 2D-CNN, $filt$ slides across video vid and generates output $convout \in \mathbb{R}^{f_m \times c \times (f-f_d+1) \times (h-f_h+1) \times (w-f_w+1)}$. Next, we apply max pooling to $convout$ to select only relevant features. The pooling will be applied only to the last three dimensions of the array $convout$.

In our experiments, we obtained best results with 32 feature maps (f_m) with the filter-size of $5 \times 5 \times 5$ (or $f_d \times f_h \times f_w$). In other words, the dimension of the filter is $32 \times 3 \times 5 \times 5 \times 5$ (or $f_m \times c \times f_d \times f_h \times f_w$). Subsequently, we apply max pooling on the output of convolution operation, with window-size being $3 \times 3 \times 3$. This is followed by a dense layer of size 300 and softmax. The activation values of this dense layer are finally used as the video features for each utterance.

3.2 Context-Dependent Feature Extraction

In sequence classification, the classification of each member is dependent on the other members. Utterances in a video maintain a sequence. We hypothesize that, within a video, there is a high probability of inter-utterance dependency with respect to their sentimental clues.

In particular, we claim that, when classifying one utterance, other utterances can provide important contextual information. This calls for a model which takes into account such inter-dependencies and the effect these might have on the target utterance. To capture this flow of informational triggers across utterances, we use a LSTM-based recurrent neural network (RNN) scheme (Gers, 2001).

3.2.1 Long Short-Term Memory

LSTM (Hochreiter and Schmidhuber, 1997) is a kind of RNN, an extension of conventional feed-forward neural network. Specifically, LSTM cells are capable of modeling long-range dependencies, which other traditional RNNs fail to do given the vanishing gradient issue. Each LSTM cell consists of an input gate i, an output gate o, and a forget gate f, to control the flow of information.

Current research (Zhou et al., 2016) indicates the benefit of using such networks to incorporate contextual information in the classification process. In our case, the LSTM network serves the purpose of context-dependent feature extraction by modeling relations among utterances. We term our architecture 'contextual LSTM'. We propose several architectural variants of it later in the paper.

3.2.2 Contextual LSTM Architecture

Let unimodal features have dimension k, each utterance is thus represented by a feature vector $\boldsymbol{x}_{i,t} \in \mathbb{R}^k$, where t represents the t^{th} utterance of the video i. For a video, we collect the vectors for all the utterances in it, to get $\boldsymbol{X_i} = [\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, ..., \boldsymbol{x}_{i,L_i}] \in \mathbb{R}^{L_i \times k}$, where L_i represents the number of utterances in the video. This matrix X_i serves as the input to the LSTM. Figure 1 demonstrates the functioning of this LSTM module.

In the procedure, *getLstmFeatures(X_i)* of Algorithm 1, each of these utterance $x_{i,t}$ is passed through a LSTM cell using the equations mentioned in line 32 to 37. The output of the LSTM cell $h_{i,t}$ is then fed into a dense layer and finally into a softmax layer (line 38 to 39). The activations of the dense layer $z_{i,t}$ are used as the context-dependent features of contextual LSTM.

3.2.3 Training

The training of the LSTM network is performed using categorical cross-entropy on each utterance's softmax output per video, i.e.,

$$loss = -\frac{1}{(\sum_{i=1}^{M} L_i)} \sum_{i=1}^{M} \sum_{j=1}^{L_i} \sum_{c=1}^{C} y_{i,c}^{j} \log_2(\hat{y}_{i,c}^{j}),$$

where M = total number of videos, L_i = number of utterances for i^{th} video, $y_{i,c}^{j}$ = original output of class c, and $\hat{y}_{i,c}^{j}$ = predicted output for j^{th} utterance of i^{th} video.

As a regularization method, dropout between the LSTM cell and dense layer is introduced to avoid overfitting. As the videos do not have the same number of utterances, padding is introduced to serve as neutral utterances. To avoid the proliferation of noise within the network, bit masking is done on these padded utterances to eliminate their effect in the network. Hyper-parameters tuning is done on the training set by splitting it into train and validation components with $80/20\%$ split.

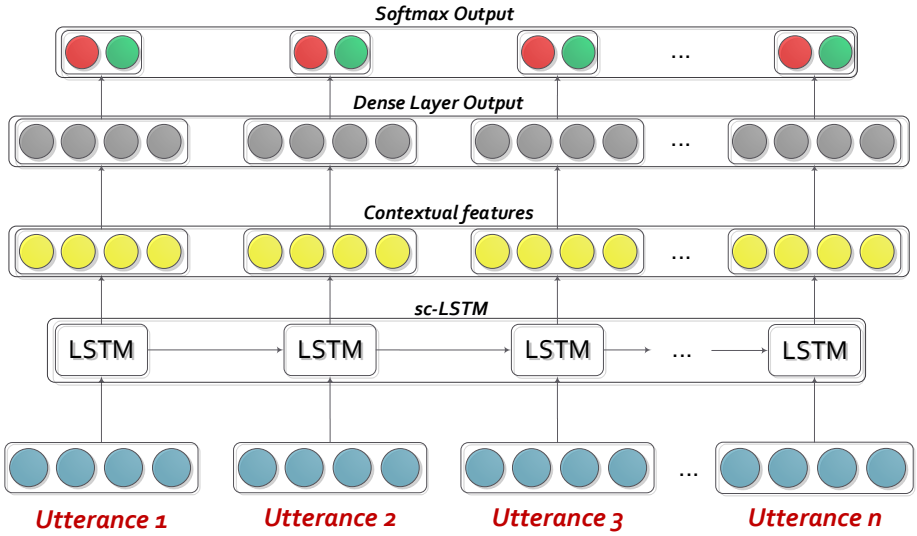


Figure 1: Contextual LSTM network: input features are passed through an unidirectional LSTM layer, followed by a dense and then a softmax layer. The dense layer activations serve as the output features.

RMSprop has been used as the optimizer which is known to resolve Adagrad's radically diminishing learning rates (Duchi et al., 2011). After feeding the training set to the network, the test set is passed through it to generate their context-dependent features. These features are finally passed through an SVM for the final classification.

Different Network Architectures We consider the following variants of the contextual LSTM architecture in our experiments.

sc-LSTM This variant of the contextual LSTM architecture consists of unidirectional LSTM cells. As this is the simple variant of the contextual LSTM, we termed it as simple contextual LSTM (sc-LSTM[1]).

h-LSTM We also investigate an architecture where the dense layer after the LSTM cell is omitted. Thus, the output of the LSTM cell $h_{i,t}$ provides our context-dependent features and the softmax layer provides the classification. We call this architecture hidden-LSTM (h-LSTM).

bc-LSTM Bi-directional LSTMs are two unidirectional LSTMs stacked together having opposite directions. Thus, an utterance can get information from utterances occurring before and after itself in the video. We replaced the regular LSTM with a bi-directional LSTM and named the resulting architecture as bi-directional contextual LSTM (bc-LSTM). The training process of this architecture is similar to sc-LSTM.

[1] http://github.com/senticnet/sc-lstm

uni-SVM In this setting, we first obtain the unimodal features as explained in Section 3.1, concatenate them and then send to an SVM for the final classification. It should be noted that using a gated recurrent unit (GRU) instead of LSTM did not improve the performance.

3.3 Fusion of Modalities

We accomplish multimodal fusion through two different frameworks, described below.

3.3.1 Non-hierarchical Framework

In this framework, we concatenate context-independent unimodal features (from Section 3.1) and feed that into the contextual LSTM networks, i.e., sc-LSTM, bc-LSTM, and h-LSTM.

3.3.2 Hierarchical Framework

Contextual unimodal features can further improve performance of the multimodal fusion framework explained in Section 3.3.1. To accomplish this, we propose a hierarchical deep network which consists of two levels.

Level-1 Context-independent unimodal features (from Section 3.1) are fed to the proposed LSTM network to get *context-sensitive* unimodal feature representations for each utterance. Individual LSTM networks are used for each modality.

Level-2 This level consists of a contextual LSTM network similar to Level-1 but independent in training and computation. Output from each LSTM network in Level-1 are concatenated and fed into this LSTM network, thus providing an inherent fusion scheme (see Figure 2).

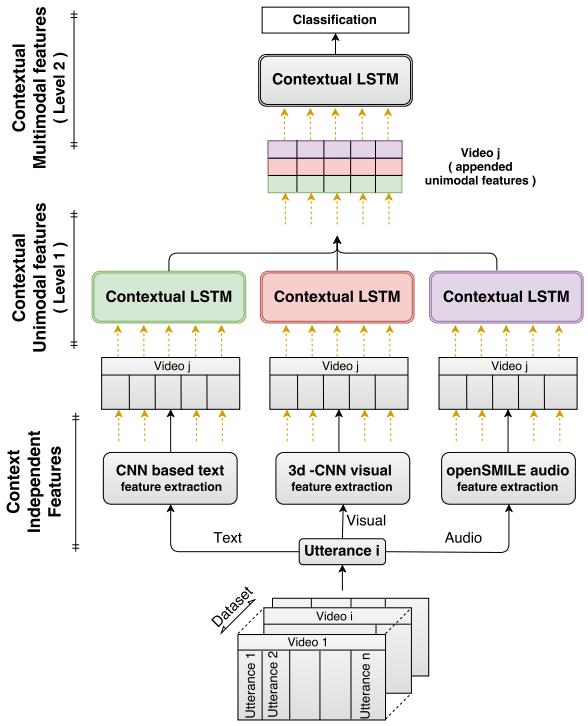


Figure 2: Hierarchical architecture for extracting context-dependent multimodal utterance features (see Figure 1 for the LSTM module).

The performance of the second level banks on the quality of the features from the previous level, with better features aiding the fusion process. Algorithm 1 describes the overall computation for utterance classification. For the hierarchical framework, we train Level-1 and Level-2 successively but separately, i.e., the training is not performed "end-to-end".

Weight		Bias	
W_i, W_f, W_c, W_o	$\in \mathbb{R}^{d \times k}$	b_i, b_f, b_c, b_o	$\in \mathbb{R}^{d}$
$P_i, P_f, P_c, P_o V_o$	$\in \mathbb{R}^{d \times d}$	b_z	$\in \mathbb{R}^{m}$
W_z	$\in \mathbb{R}^{m \times d}$	b_{sft}	$\in \mathbb{R}^{c}$
W_{sft}	$\in \mathbb{R}^{c \times m}$		

Table 1: Summary of notations used in Algorithm 1. Legenda: d = dimension of hidden unit; k = dimension of input vectors to LSTM layer; c = number of classes.

4 Experiments

4.1 Dataset details

Most of the research in multimodal sentiment analysis is performed on datasets with speaker overlap in train and test splits. Because each individual has a unique way of expressing emotions and sentiments, however, finding generic, person-independent features for sentiment analysis is very important.

Algorithm 1 Proposed Architecture

procedure TRAINARCHITECTURE(U, V)
  Train context-independent models with U
  for `i:[1,M]` **do** ▷ extract baseline features
    for `j:[1,`L_i`]` **do**
      $x_{i,j} \leftarrow TextFeatures(u_{i,j})$
      $x^{'}_{i,j} \leftarrow VideoFeatures(u_{i,j})$
      $x^{"}_{i,j} \leftarrow AudioFeatures(u_{i,j})$

  Unimodal:
  Train LSTM at Level-1 with $X, X^{'} and X^{"}$.
  for `i:[1,`M`]` **do** ▷ unimodal features
    $Z_i \leftarrow getLSTMFeatures(X_i)$
    $Z^{'}_i \leftarrow getLSTMFeatures(X^{'}_i)$
    $Z^{"}_i \leftarrow getLSTMFeatures(X^{"}_i)$

  Multimodal:
  for `i:[1,M]` **do**
    for `j:[1,`L_i`]` **do**
      if `Non-hierarchical fusion` **then**
        $x^{*}_{i,j} \leftarrow (x_{i,j} || x^{'}_{i,j} || x^{"}_{i,j})$ ▷ concatenation
      else
        if `Hierarchical fusion` **then**
          $x^{*}_{i,j} \leftarrow (z_{i,j} || z^{'}_{i,j} || z^{"}_{i,j})$ ▷ concatenation
  Train LSTM at Level-2 with X^*.
  for `i:[1,`M`]` **do** ▷ multimodal features
    $Z^{*}_i \leftarrow getLSTMFeatures(X^{*}_i)$
  testArchitecture(V)
  return Z^*
procedure TESTARCHITECTURE(V)
  Similar to training phase. V is passed through the learnt models to get the features and classification outputs. Table 1 shows the trainable parameters.
procedure GETLSTMFEATURES(X_i) ▷ for i^{th} video
  $Z_i \leftarrow \phi$
  for `t:[1,`L_i`]` **do** ▷ Table 1 provides notation
    $i_t \leftarrow \sigma(W_i x_{i,t} + P_i . h_{t-1} + b_i)$
    $\widetilde{C_t} \leftarrow tanh(W_c x_{i,t} + P_c h_{t-1} + b_c)$
    $f_t \leftarrow \sigma(W_f x_t + P_f h_{t-1} + b_f)$
    $C_t \leftarrow i_t * \widetilde{C_t} + f_t * C_{t-1}$
    $o_t \leftarrow \sigma(W_o x_t + P_o h_{t-1} + V_o C_t + b_o)$
    $h_t \leftarrow o_t * tanh(C_t)$ ▷ output of lstm cell
    $z_t \leftarrow ReLU(W_z h_t + b_z)$ ▷ dense layer
    $prediction \leftarrow softmax(W_{sft} z_t + b_{sft})$
    $Z_i \leftarrow Z_i \cup z_t$
  return Z_i

In real-world applications, the model should be robust to person idiosyncrasy but it is very difficult to come up with a generalized model from the behavior of a limited number of individuals. To this end, we perform person-independent experiments to study generalization of our model, i.e., our train/test splits of the datasets are completely disjoint with respect to speakers.

Multimodal Sentiment Analysis Datasets

MOSI The MOSI dataset (Zadeh et al., 2016) is a dataset rich in sentimental expressions where 93 people review topics in English. The videos

are segmented with each segments sentiment label scored between +3 (strong positive) to -3 (strong negative) by 5 annotators. We took the average of these five annotations as the sentiment polarity and, hence, considered only two classes (positive and negative). The train/validation set consists of the first 62 individuals in the dataset. The test set contains opinionated videos by rest 31 speakers. In particular, 1447 and 752 utterances are used in training and test, respectively.

MOUD This dataset (Pérez-Rosas et al., 2013) contains product review videos provided by 55 persons. The reviews are in Spanish (we used Google Translate API[2] to get the English transcripts). The utterances are labeled to be either *positive, negative or neutral*. However, we drop the *neutral* label to maintain consistency with previous work. Out of 79 videos in the dataset, 59 videos are considered in the train/val set.

Multimodal Emotion Recognition Datasets

IEMOCAP The IEMOCAP (Busso et al., 2008) contains the acts of 10 speakers in a two-way conversation segmented into utterances. The medium of the conversations in all the videos is English. The database contains the following categorical labels: anger, happiness, sadness, neutral, excitement, frustration, fear, surprise, and other, but we take only the first four so as to compare with the state of the art (Rozgic et al., 2012). Videos by the first 8 speakers are considered in the training set. The train/test split details are provided in Table 2, which provides information regarding train/test split of all the datasets. Table 2 also provides cross-dataset split details where the datasets MOSI and MOUD are used for training and testing, respectively. The proposed model being used on reviews from different languages allows us to analyze its robustness and generalizability.

4.1.1 Characteristic of the Datasets

In order to evaluate the robustness of our proposed method, we employ it on multiple datasets of different kinds. Both MOSI and MOUD are used for the sentiment classification task but they consist of review videos spoken in different languages, i.e., English and Spanish, respectively.

[2] http://translate.google.com

IEMOCAP dataset is different from MOSI and MOUD since it is annotated with emotion labels. Apart from this, IEMOCAP dataset was created using a different method than MOSI and MOUD. These two datasets were developed by crawling consumers' spontaneous online product review videos from popular social websites and later labeled with sentiment labels. To curate the IEMOCAP dataset, instead, subjects were provided affect-related scripts and asked to act.

As pointed out by Poria et al. (Poria et al., 2017a), acted dataset like IEMOCAP can suffer from biased labeling and incorrect acting which can further cause the poor generalizability of the models trained on the acted datasets.

Dataset	Train		Test	
	uttrnce	*video*	*uttrnce*	*video*
IEMOCAP	4290	120	1208	31
MOSI	1447	62	752	31
MOUD	322	59	115	20
MOSI → MOUD	2199	93	437	79

Table 2: uttrnce: Utterance; Person-Independent Train/Test split details of each dataset (≈ 70/30 % split). Legenda: X→Y represents train: X and test: Y; Validation sets are extracted from the shuffled training sets using 80/20 % train/val ratio.

It should be noted that the datasets' individual configuration and splits are same throughout all the experiments (i.e., context-independent unimodal feature extraction, LSTM-based context-dependent unimodal and multimodal feature extraction and classification).

4.2 Performance of Different Models

In this section, we present unimodal and multimodal sentiment analysis performance of different LSTM network variants as explained in Section 3.2.3 and comparison with the state of the art.

Hierarchical vs Non-hierarchical Fusion Framework As expected, trained contextual unimodal features help the hierarchical fusion framework to outperform the non-hierarchical framework. Table 3 demonstrates this by comparing the hierarchical and the non-hierarchical frameworks using the bc-LSTM network.

For this reason, we the rest of the analysis only leverages on the hierarchical framework. The non-hierarchical model outperforms the baseline uni-SVM, which confirms that it is the context-sensitive learning paradigm that plays the key role in improving performance over the baseline.

Comparison of Different Network Variants It is to be noted that both sc-LSTM and bc-LSTM perform quite well on the multimodal emotion recognition and sentiment analysis datasets. Since bc-LSTM has access to both the preceding and following information of the utterance sequence, it performs consistently better on all the datasets over sc-LSTM. The usefulness of the dense layer in increasing the performance is evident from the experimental results shown in Table 3. The performance improvement is in the range of 0.3% to 1.5% on MOSI and MOUD datasets. On the IEMOCAP dataset, the performance improvement of bc-LSTM and sc-LSTM over h-LSTM is in the range of 1% to 5%.

Comparison with the Baselines Every LSTM network variant has outperformed the baseline uni-SVM on all the datasets by the margin of 2% to 5% (see Table 3). These results prove our initial hypothesis that modeling the contextual dependencies among utterances (which uni-SVM cannot do) improves the classification. The higher performance improvement on the IEMOCAP dataset indicates the necessity of modeling long-range dependencies among the utterances as continuous emotion recognition is a multiclass sequential problem where a person does not frequently change emotions (Wöllmer et al., 2008). We have implemented and compared with the current state-of-the-art approach proposed by (Poria et al., 2015). In their method, they extracted features from each modality and fed these to a MKL classifier. However, they did not conduct the experiment in a speaker-independent manner and also did not consider the contextual relation among the utterances. In Table 3, the results in bold are statistically significant ($p < 0.05$) in compare to uni-SVM. Experimental results in Table 4 show that the proposed method outperformes (Poria et al., 2015) by a significant margin. For the emotion recognition task, we have compared our method with the current state of the art (Rozgic et al., 2012), who extracted features in a similar fashion to (Poria et al., 2015) (although they used SVM trees (Yuan et al., 2006) for the fusion).

4.3 Importance of the Modalities

As expected, in all kinds of experiments, bimodal and trimodal models have outperformed unimodal models. Overall, audio modality has performed better than visual on all the datasets.

On MOSI and IEMOCAP datasets, the textual classifier achieves the best performance over other unimodal classifiers. On IEMOCAP dataset, the unimodal and multimodal classifiers obtained poor performance to classify *neutral* utterances. The textual modality, combined with non-textual modes, boosts the performance in IEMOCAP by a large margin. However, the margin is less in the other datasets.

On the MOUD dataset, the textual modality performs worse than audio modality due to the noise introduced in translating Spanish utterances to English. Using Spanish word vectors[3] in *text-CNN* results in an improvement of 10%. Nonetheless, we report results using these translated utterances as opposed to utterances trained on Spanish word vectors, in order to make fair comparison with (Poria et al., 2015).

4.4 Generalization of the Models

To test the generalizability of the models, we have trained our framework on complete MOSI dataset and tested on MOUD dataset (Table 5). The performance was poor for audio and textual modality as the MOUD dataset is in Spanish while the model is trained on MOSI dataset, which is in English language. However, notably the visual modality performs better than the other two modalities in this experiment, which means that in cross-lingual scenarios facial expressions carry more generalized, robust information than audio and textual modalities. We could not carry out a similar experiment for emotion recognition as no other utterance-level dataset apart from the IEMOCAP was available at the time of our experiments.

4.5 Qualitative Analysis

The need for considering context dependency (see Section 1) is of prime importance for utterance-level sentiment classification. For example, in the utterance *"What would have been a better name for the movie"*, the speaker is attempting to comment the quality of the movie by giving an appropriate name. However, the sentiment is expressed implicitly and requires the contextual knowledge about the mood of the speaker and his/her general opinion about the film. The baseline unimodal-SVM and state of the art fail to classify this utterance correctly[4].

[3] http://crscardellino.me/SBWCE

[4] RNTN classifies it as neutral. It can be seen here http://nlp.stanford.edu:8080/sentiment/rntnDemo.html

Modality	MOSI					MOUD					IEMOCAP				
	hierarchical (%)				non-hier (%)	hierarchical (%)				non-hier (%)	hierarchical (%)				non-hier (%)
	uni-SVM	h-LSTM	sc-LSTM	bc-LSTM		uni-SVM	h-LSTM	sc-LSTM	bc-LSTM		uni-SVM	h-LSTM	sc-LSTM	bc-LSTM	
T	75.5	77.4	77.6	**78.1**		49.5	50.1	51.3	**52.1**		65.5	68.9	71.4	**73.6**	
V	53.1	55.2	55.6	**55.8**		46.3	48.0	48.2	**48.5**		47.0	52.0	52.6	**53.2**	
A	58.5	59.6	59.9	**60.3**		51.5	56.3	57.5	**59.9**		52.9	54.4	55.2	**57.1**	
T + V	76.7	78.9	79.9	**80.2**	78.5	50.2	50.6	51.3	**52.2**	50.9	68.5	70.3	72.3	**75.4**	73.2
T + A	75.8	78.3	78.8	**79.3**	78.2	53.1	56.9	57.4	**60.4**	55.5	70.1	74.1	75.2	**75.6**	74.5
V + A	58.6	61.5	61.8	**62.1**	60.3	62.8	62.9	64.4	**65.3**	64.2	67.6	67.8	68.2	**68.9**	67.3
T + V + A	77.9	78.1	78.6	**80.3**	78.1	66.1	66.4	67.3	**68.1**	67.0	72.5	73.3	74.2	**76.1**	73.5

Table 3: Comparison of models mentioned in Section 3.2.3. The table reports the accuracy of classification. Legenda: non-hier ← Non-hierarchical bc-lstm. For remaining fusion, hierarchical fusion framework is used (Section 3.3.2).

Modality	Sentiment (%)		Emotion on IEMOCAP (%)			
	MOSI	MOUD	*angry*	*happy*	*sad*	*neutral*
T	78.12	52.17	76.07	78.97	76.23	67.44
V	55.80	48.58	53.15	58.15	55.49	51.26
A	60.31	59.99	58.37	60.45	61.35	52.31
T + V	80.22	52.23	77.24	78.99	78.35	68.15
T + A	79.33	60.39	77.15	79.10	78.10	69.14
V + A	62.17	65.36	68.21	71.97	70.35	62.37
A + V + T	**80.30**	**68.11**	**77.98**	**79.31**	**78.30**	**69.92**
State-of -the-art	73.55[1]	63.25[1]	73.10 [2]	72.40[2]	61.90[2]	58.10[2]

[1]by (Poria et al., 2015),[2]by (Rozgic et al., 2012)

Table 4: Accuracy % on textual (T), visual (V), audio (A) modality and comparison with the state of the art. For the fusion, the hierarchical fusion framework was used.

Modality	MOSI → MOUD			
	uni-SVM	h-LSTM	sc-LSTM	bc-LSTM
T	46.5%	46.5%	46.6%	**46.9%**
V	43.3%	45.5%	48.3%	**49.6%**
A	42.9%	46.0%	46.4%	**47.2%**
T + V	49.8%	49.8%	49.8%	**49.8%**
T + A	50.4%	50.9%	51.1%	**51.3%**
V + A	46.0%	47.1%	49.3%	**49.6%**
T + V + A	51.1%	52.2%	52.5%	**52.7%**

Table 5: Cross-dataset comparison in terms of classification accuracy.

However, information from neighboring utterances, e.g., *"And I really enjoyed it"* and *"The countryside which they showed while going through Ireland was astoundingly beautiful"* indicate its positive context and help our contextual model to classify the target utterance correctly. Such contextual relationships are prevalent throughout the dataset.

In order to have a better understanding of the roles of each modality for the overall classification, we have also done some qualitative analysis. For example, the utterance *"who doesn't have any presence or greatness at all"* was classified as positive by the audio classifier (as "presence and greatness at all" was spoken with enthusiasm). However, the textual modality caught the negation induced by "doesn't" and classified it correctly. The same happened to the utterance "amazing special effects", which presented no jest of enthusiasm in the speaker's voice nor face, but was correctly classified by the textual classifier.

On other hand, the textual classifier classified the utterance "that like to see comic book characters treated responsibly" as positive (for the presence of "like to see" and "responsibly") but the high pitch of anger in the person's voice and the frowning face helps to identify this as a negative utterance. In some cases, the predictions of the proposed method are wrong because of face occlusion or noisy audio. Also, in cases where sentiment is very weak and non contextual, the proposed approach shows some bias towards its surrounding utterances, which further leads to wrong predictions.

5 Conclusion

The contextual relationship among utterances in a video is mostly ignored in the literature. In this paper, we developed a LSTM-based network to extract contextual features from the utterances of a video for multimodal sentiment analysis. The proposed method has outperformed the state of the art and showed significant performance improvement over the baseline.

As future work, we plan to develop a LSTM-based attention model to determine the importance of each utterance and its specific contribution to each modality for sentiment classification.

References

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation* 42(4):335–359.

Erik Cambria, Dipankar Das, Sivaji Bandyopadhyay, and Antonio Feraco. 2017a. *A Practical Guide to Sentiment Analysis*. Springer, Cham, Switzerland.

Erik Cambria, Devamanyu Hazarika, Soujanya Poria, Amir Hussain, and RBV Subramanyam. 2017b. Benchmarking multimodal sentiment aanlysis. In *CICLing*.

Erik Cambria, Soujanya Poria, Rajiv Bajpai, and Björn Schuller. 2016. SenticNet 4: A semantic resource for sentiment analysis based on conceptual primitives. In *COLING*. pages 2666–2677.

Erik Cambria and Bebo White. 2014. Jumping NLP curves: A review of natural language processing research. *IEEE Computational Intelligence Magazine* 9(2):48–57.

Lawrence S Chen, Thomas S Huang, Tsutomu Miyasato, and Ryohei Nakatsu. 1998. Multimodal human emotion/expression recognition. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE, pages 366–371.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Dragos Datcu and L Rothkrantz. 2008. Semantic audio-visual data fusion for automatic emotion recognition. *Euromedia'2008* .

Liyanage C De Silva, Tsutomu Miyasato, and Ryohei Nakatsu. 1997. Facial emotion recognition using multi-modal information. In *Proceedings of ICICS*. IEEE, volume 1, pages 397–401.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Paul Ekman. 1974. Universal facial expressions of emotion. *Culture and Personality: Contemporary Readings/Chicago* .

Florian Eyben, Martin Wöllmer, Alex Graves, Björn Schuller, Ellen Douglas-Cowie, and Roddy Cowie. 2010a. On-line emotion recognition in a 3-d activation-valence-time continuum using acoustic and linguistic cues. *Journal on Multimodal User Interfaces* 3(1-2):7–19.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010b. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*. ACM, pages 1459–1462.

Felix Gers. 2001. *Long Short-Term Memory in Recurrent Neural Networks*. Ph.D. thesis, Universität Hannover.

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

Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 2013. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence* 35(1):221–231.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. pages 1725–1732.

Loic Kessous, Ginevra Castellano, and George Caridakis. 2010. Multimodal emotion recognition in speech-based interaction using facial expression, body gesture and acoustic analysis. *Journal on Multimodal User Interfaces* 3(1-2):33–48.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *COLING*. Osaka, pages 171–180.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. 2017. Deep learning based document modeling for personality detection from text. *IEEE Intelligent Systems* 32(2):74–79.

Angeliki Metallinou, Sungbok Lee, and Shrikanth Narayanan. 2008. Audio-visual emotion recognition using gaussian mixture models for face and voice. In *Tenth IEEE International Symposium on ISM 2008*. IEEE, pages 250–257.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

David Olson. 1977. From utterance to text: The bias of language in speech and writing. *Harvard educational review* 47(3):257–281.

Luca Oneto, Federica Bisio, Erik Cambria, and Davide Anguita. 2016. Statistical learning theory and ELM for big social data analysis. *IEEE Computational Intelligence Magazine* 11(3):45–55.

Verónica Pérez-Rosas, Rada Mihalcea, and Louis-Philippe Morency. 2013. Utterance-level multimodal sentiment analysis. In *ACL (1)*. pages 973–982.

Soujanya Poria, Erik Cambria, Rajiv Bajpai, and Amir Hussain. 2017a. A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion* .

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *Proceedings of EMNLP*. pages 2539–2544.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016a. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems* 108:42–49.

Soujanya Poria, Erik Cambria, D Hazarika, and Prateek Vij. 2016b. A deeper look into sarcastic tweets using deep convolutional neural networks. In *COLING*. pages 1601–1612.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Federica Bisio. 2016c. Sentic LDA: Improving on LDA with semantic similarity for aspect-based sentiment analysis. In *IJCNN*. pages 4465–4473.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016d. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, pages 439–448.

Soujanya Poria, Haiyun Peng, Amir Hussain, Newton Howard, and Erik Cambria. 2017b. Ensemble application of convolutional neural networks and multiple kernel learning for multimodal sentiment analysis. *Neurocomputing* .

Dheeraj Rajagopal, Erik Cambria, Daniel Olsher, and Kenneth Kwok. 2013. A graph-based approach to commonsense concept extraction and semantic similarity detection. In *WWW*. Rio De Janeiro, pages 565–570.

Viktor Rozgic, Sankaranarayanan Ananthakrishnan, Shirin Saleem, Rohit Kumar, and Rohit Prasad. 2012. Ensemble of svm trees for multimodal emotion recognition. In *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*. IEEE, pages 1–4.

Björn Schuller. 2011. Recognizing affect from linguistic information in 3d continuous space. *IEEE Transactions on Affective Computing* 2(4):192–205.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*. pages 1631–1642.

Vee Teh and Geoffrey E Hinton. 2001. Rate-coded restricted boltzmann machines for face recognition. In T Leen, T Dietterich, and V Tresp, editors, *Advances in neural information processing system*. volume 13, pages 908–914.

Martin Wöllmer, Florian Eyben, Stephan Reiter, Björn W Schuller, Cate Cox, Ellen Douglas-Cowie, Roddy Cowie, et al. 2008. Abandoning emotion classes-towards continuous emotion recognition with modelling of long-range dependencies. In *Interspeech*. volume 2008, pages 597–600.

Martin Wollmer, Felix Weninger, Timo Knaup, Bjorn Schuller, Congkai Sun, Kenji Sagae, and Louis-Philippe Morency. 2013. Youtube movie reviews: Sentiment analysis in an audio-visual context. *IEEE Intelligent Systems* 28(3):46–53.

Chung-Hsien Wu and Wei-Bin Liang. 2011. Emotion recognition of affective speech based on multiple classifiers using acoustic-prosodic information and semantic labels. *IEEE Transactions on Affective Computing* 2(1):10–21.

Xun Yuan, Wei Lai, Tao Mei, Xian-Sheng Hua, Xiu-Qing Wu, and Shipeng Li. 2006. Automatic video genre categorization using hierarchical svm. In *Image Processing, 2006 IEEE International Conference on*. IEEE, pages 2905–2908.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems* 31(6):82–88.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *The 54th Annual Meeting of the Association for Computational Linguistics*. pages 207–213.

A Multidimensional Lexicon for Interpersonal Stancetaking

Umashanthi Pavalanathan
Georgia Institute of Technology
Atlanta, GA
umashanthi@gatech.edu

Jim Fitzpatrick
University of Pittsburgh
Pittsburgh, PA
jim.fitzpatrick@gmail.com

Scott F. Kiesling
University of Pittsburgh
Pittsburgh, PA
kiesling@pitt.edu

Jacob Eisenstein
Georgia Institute of Technology
Atlanta, GA
jacobe@gatech.edu

Abstract

The sociolinguistic construct of stancetaking describes the activities through which discourse participants create and signal relationships to their interlocutors, to the topic of discussion, and to the talk itself. Stancetaking underlies a wide range of interactional phenomena, relating to formality, politeness, affect, and subjectivity. We present a computational approach to stancetaking, in which we build a theoretically-motivated lexicon of stance markers, and then use multidimensional analysis to identify a set of underlying stance dimensions. We validate these dimensions intrinsically and extrinsically, showing that they are internally coherent, match pre-registered hypotheses, and correlate with social phenomena.

1 Introduction

What does it mean to be welcoming or standoffish, light-hearted or cynical? Such interactional styles are performed primarily with language, yet little is known about how linguistic resources are arrayed to create these social impressions. The sociolinguistic concept of *interpersonal stancetaking* attempts to answer this question, by providing a conceptual framework that accounts for a range of interpersonal phenomena, subsuming formality, politeness, and subjectivity (Du Bois, 2007).[1] This framework has been applied almost exclusively through qualitative methods, using close readings of individual texts or dialogs to uncover how language is used to position individuals with respect to their interlocutors and readers.

We attempt the first large-scale operationalization of stancetaking through computational methods. Du Bois (2007) formalizes stancetaking as a multi-dimensional construct, reflecting the relationship of discourse participants to (a) the audience or interlocutor; (b) the topic of discourse; (c) the talk or text itself. However, the multidimensional nature of stancetaking poses problems for traditional computational approaches, in which labeled data is obtained by relying on annotator intuitions about scalar concepts such politeness (Danescu-Niculescu-Mizil et al., 2013) and formality (Pavlick and Tetreault, 2016).

Instead, our approach is based on a theoretically-guided application of unsupervised learning, in the form of factor analysis, applied to lexical features. Stancetaking is characterized in large part by an array of linguistic features ranging from discourse markers such as *actually* to backchannels such as *yep* (Kiesling, 2009). We therefore first compile a lexicon of stance markers, combining prior lexicons from Biber and Finegan (1989) and the Switchboard Dialogue Act Corpus (Jurafsky et al., 1998). We then extend this lexicon to the social media domain using word embeddings. Finally, we apply multi-dimensional analysis of co-occurrence patterns to identify a small set of *stance dimensions*.

To measure the internal coherence (construct validity) of the stance dimensions, we use a word

[1] Stancetaking is distinct from the notion of *stance* which corresponds to a position in a debate (Walker et al., 2012). Similarly, Freeman et al. (2014) correlate phonetic features with the *strength* of such argumentative stances.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 884–895
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1082

intrusion task (Chang et al., 2009) and a set of preregistered hypotheses. To measure the utility of the stance dimensions, we perform a series of extrinsic evaluations. A predictive evaluation shows that the membership of online communities is determined in part by the interactional stances that predominate in those communities. Furthermore, the induced stance dimensions are shown to align with annotations of politeness and formality.

Contributions We operationalize the sociolinguistic concept of stancetaking as a multidimensional framework, making it possible to measure at scale. Specifically,

- we contribute a lexicon of stance markers based on prior work and adapted to the genre of online interpersonal discourse;
- we group stance markers into latent dimensions;
- we show that these stance dimensions are internally coherent;
- we demonstrate that the stance dimensions predict and correlate with social phenomena.[2]

2 Related Work

From a theoretical perspective, we build on prior work on interactional meaning in language. Methodologically, our paper relates to prior work on lexicon-based analysis and contrastive studies of social media communities.

2.1 Linguistic Variation and Social Meaning

In computational sociolinguistics (Nguyen et al., 2016), language variation has been studied primarily in connection with macro-scale social variables, such as age (Argamon et al., 2007; Nguyen et al., 2013), gender (Burger et al., 2011; Bamman et al., 2014), race (Eisenstein et al., 2011; Blodgett et al., 2016), and geography (Eisenstein et al., 2010). This parallels what Eckert (2012) has called the "first wave" of language variation studies in sociolinguistics, which also focused on macro-scale variables.

More recently, sociolinguists have dedicated increased attention to situational and stylistic variation, and the *interactional meaning* that such variation can convey (Eckert and Rickford, 2001). This linguistic research can be aligned with computational efforts to quantify phenomena such as subjectivity (Riloff and Wiebe, 2003), sentiment (Wiebe et al., 2005), politeness (Danescu-Niculescu-Mizil et al., 2013), formality (Pavlick and Tetreault, 2016), and power dynamics (Prabhakaran et al., 2012). While linguistic research on interactional meaning has focused largely on qualitative methodologies such as discourse analysis (e.g., Bucholtz and Hall, 2005), these computational efforts have made use of crowdsourced annotations to build large datasets of, for example, polite and impolite text. These annotation efforts draw on the annotators' intuitions about the meaning of these sociolinguistic constructs.

Interpersonal stancetaking represents an attempt to unify concepts such as sentiment, politeness, formality, and subjectivity under a single theoretical framework (Jaffe, 2009; Kiesling, 2009). The key idea, as articulated by Du Bois (2007), is that stancetaking captures the speaker's relationship to (a) the topic of discussion, (b) the interlocutor or audience, and (c) the talk (or writing) itself. Various configurations of these three legs of the "stance triangle" can account for a range of phenomena. For example, epistemic stance relates to the speaker's certainty about what is being expressed, while affective stance indicates the speaker's emotional position with respect to the content (Ochs, 1993).

The framework of stancetaking has been widely adopted in linguistics, particularly in the discourse analytic tradition, which involves close reading of individual texts or conversations (Kärkkäinen, 2006; Keisanen, 2007; Precht, 2003; White, 2003). But despite its strong theoretical foundation, we are aware of no prior efforts to operationalize stancetaking at scale. Since annotators may not have strong intuitions about stance — in the way that they do about formality and politeness — we cannot rely on the annotation methodologies employed in prior work. We take a different approach, performing a multidimensional analysis of the distribution of likely stance markers.

2.2 Lexicon-based Analysis

Our operationalization of stancetaking is based on the induction of lexicons of stance markers. The lexicon-based methodology is related to earlier work from social psychology, such as the General Inquirer (Stone, 1966) and LIWC (Tausczik and Pennebaker, 2010). In LIWC, the basic categories were identified first, based on psychological

[2]Lexicons and stance dimensions are available at https://github.com/umashanthi-research/multidimensional-stance-lexicon

constructs (e.g., positive emotion, cognitive processes, drive to power) and syntactic groupings of words and phrases (e.g., pronouns, prepositions, quantifiers). The lexicon designers then manually contructed lexicons for each category, augmenting their intuitions by using distributional statistics to suggest words that may have been missed (Pennebaker et al., 2015). In contrast, we follow the approach of Biber (1991), using multidimensional analysis to identify latent groupings of markers based on co-occurrence statistics. We then use crowdsourcing and extrinsic comparisons to validate the coherence of these dimensions.

2.3 Multicommunity Studies

Social media platforms such as Reddit, Stack Exchange, and Wikia can be considered *multicommunity environments*, in that they host multiple subcommunities with distinct social and linguistic properties. Such subcommunities can be contrasted in terms of topics (Adamic et al., 2008; Hessel et al., 2014) and social networks (Backstrom et al., 2006). Our work focuses on Reddit, emphasizing community-wide differences in norms for interpersonal interaction. In the same vein, Tan and Lee (2015) attempt to characterize stylistic differences across subreddits by focusing on very common words and parts-of-speech; Tran and Ostendorf (2016) use language models and topic models to measure similarity across threads within a subreddit. One distinction of our approach is that the use of multidimensional analysis gives us interpretable dimensions of variation. This makes it possible to identify the specific interpersonal features that vary across communities.

3 Data

Reddit, one of the internet's largest social media platforms, is a collection of subreddits organized around various topics of interest. As of January 2017, there were more than one million subreddits and nearly 250 million users, discussing topics ranging from politics (*r/politics*) to horror stories (*r/nosleep*).[3] Although Reddit was originally designed for sharing hyperlinks, it also provides the ability to post original textual content, submit comments, and vote on content quality (Gilbert, 2013). Reddit's conversation-like threads are therefore well suited for the study of interpersonal social and linguistic phenomena.

[3] http://redditmetrics.com/

Subreddits	126,789
Authors	6,401,699
Threads	52,888,024
Comments	531,804,658

Table 1: Dataset size

For example, the following are two comments from the subreddit *r/malefashionadvice*, posted in response to a picture posted by a user asking for fashion advise.

U_1: *"I think the beard* ***looks pretty good****.* ***Definitely*** *not the goatee. Clean shaven is always the safe option."*

U_2: *"****Definitely*** *the beard. But keep it trimmed."*

The phrases in **bold** face are markers of stance, indicating a *evaluative* stance. The following example is a part of a thread in the subreddit *r/photoshopbattles* where users discuss an edited image posted by the original poster *OP*. The phrases in **bold** face are markers of stance, indicating an *involved* and *interactional* stance.

U_3: *"****Ha ha*** *awesome!"*
U_4: "are those..... furries?"
OP: *"****yes****, sir. They are!"*
U_4: *"****Oh cool****. That makes sense!"*

We used an archive of 530 million comments posted on Reddit in 2014, retrieved from the public archive of Reddit comments.[4] This dataset consists of each post's textual content, along with metadata that identifies the subreddit, thread, author, and post creation time. More statistics about the full dataset are shown in Table 1.

4 Stance Lexicon

Interpersonal stancetaking can be characterized in part by an array of linguistic features such as hedges (e.g., *might*, *kind of*), discourse markers (e.g., *actually*, *I mean*), and backchannels (e.g., *yep*, *um*). Our analysis focuses on these markers, which we collect into a lexicon.

4.1 Seed lexicon

We began with a seed lexicon of stance markers from Biber and Finegan (1989), who compiled an

[4] https://archive.org/details/2015_reddit_comments_corpus

extensive list by surveying dictionaries, previous studies on stance, and texts in several genres of English. This list includes certainty adverbs (e.g., *actually*, *of course*, *in fact*), affect markers (e.g., *amazing*, *thankful*, *sadly*), and hedges (e.g., *kind of*, *maybe*, *something like*) among other adverbial, adjectival, verbal, and modal markers of stance. In total, this list consists of 448 stance markers.

The Biber and Finegan (1989) lexicon is primarily based on written genres from the pre-social media era. Our dataset — like much of the recent work in this domain — consists of online discussions, which differ significantly from printed texts (Eisenstein, 2013). One difference is that online discussions contain a number of dialog act markers that are characteristic of spoken language, such as *oh yeah*, *nah*, *wow*. We accounted for this by adding 74 dialog act markers from the Switchboard Dialog Act Corpus (Jurafsky et al., 1998). The final seed lexicon consists of 517 unique markers, from these two sources. Note that the seed lexicon also includes markers that contain multiple tokens (e.g. *kind of*, *I know*).

4.2 Lexicon expansion

Online discussions differ not only from written texts, but also from spoken discussions, due to their use of non-standard vocabulary and spellings. To measure stance accurately, these genre differences must be accounted for. We therefore expanded the seed lexicon using automated techniques based on distributional statistics. This is similar to prior work on the expansion of sentiment lexicons (Hatzivassiloglou and McKeown, 1997; Hamilton et al., 2016).

Our lexicon expansion approach used word embeddings to find words that are distributionally similar to those in the seed set. We trained word embeddings on a corpus of 25 million Reddit comments and a vocabulary of 100K most frequent words on Reddit using the structured skip-gram models of both WORD2VEC (Mikolov et al., 2013) and WANG2VEC (Ling et al., 2015) with default parameters. The WANG2VEC method augments WORD2VEC by accounting for word order information. We found the similarity judgments obtained from WANG2VEC to be qualitatively more meaningful, so we used these embeddings to construct the expanded lexicon.[5]

[5]We used the following default parameters: 100 dimensions, a window size of five, a negative sampling size of ten, five-epoch iterations, and a sub-sampling rate of 10^{-4}.

Seed term	Expanded terms
(*Example seeds from Biber and Finegan (1989)*)	
significantly	considerably, substantially, dramatically
certainly	surely, frankly, definitely
incredibly	extremely, unbelievably, exceptionally
(*Example seeds from Jurafsky et al. (1998)*)	
nope	nah, yup, nevermind
great	fantastic, terrific, excellent

Table 2: Stance lexicon: seed and expanded terms.

To perform lexicon expansion, we constructed a dictionary of candidate terms, consisting of all unigrams that occur with a frequency rate of at least 10^{-7} in the Reddit comment corpus. Then, for each single-token marker in the seed lexicon, we identified all terms from the candidate set whose embedding has cosine similarity of at least 0.75 with respect to the seed marker.[6] Table 2 shows examples of seed markers and related terms we extracted from word embeddings. Through this procedure, we identified 228 additional markers based on similarity to items in the seed list from Biber and Finegan (1989), and 112 additional markers based on the seed list of dialog acts. In total, our stance lexicon contains 812 unique markers.

5 Linguistic Dimensions of Stancetaking

To summarize the main axes of variation across the lexicon of stance markers, we apply a multidimensional analysis (Biber, 1992) to the distributional statistics of stance markers across subreddit communities. Each dimension of variation can then be viewed as a spectrum, characterized by the stance markers and subreddits that are associated with the positive and negative extremes. Multidimensional analysis is based on singular value decomposition, which has been applied successfully to a wide range of problems in natural language processing and information retrieval (e.g., Landauer et al., 1998). While Bayesian topic models are an appealing alternative, singular value decomposition is fast and deterministic, with a minimal number of tuning parameters.

[6]We tried different thresholds on the similarity value and the corpus frequency, and the reported values were chosen based on the quality of the resulting related terms. This was done prior to any of the validations or extrinsic analyses described later in the paper.

5.1 Extracting Stance Dimensions

Our analysis is based on the co-occurrence of stance markers and subreddits. This is motivated by our interest in comparisons of the interactional styles of online communities within Reddit, and by the premise that these distributional differences reflect socially meaningful communicative norms. A pilot study applied the same technique to the co-occurrence of stance markers and individual authors, and the resulting dimensions appeared to be less stylistically coherent.

Singular value decomposition is often used in combination with a transformation of the co-occurrence counts by pointwise mutual information (Bullinaria and Levy, 2007). This transformation ensures that each cell in the matrix indicates how much more likely a stance marker is to co-occur with a given subreddit than would happen by chance under an independence assumption. Because negative PMI values tend to be unreliable, we use positive PMI (PPMI), which involves replacing all negative PMI values with zeros (Niwa and Nitta, 1994). Therefore, we obtain stance dimensions by applying singular value decomposition to the matrix constructed as follows:

$$X_{m,s} = \left(\log \frac{\Pr(\text{marker} = m, \text{subreddit} = s)}{\Pr(\text{marker} = m) \Pr(\text{subreddit} = s)} \right)_+ .$$

Truncated singular value decomposition performs the approximate factorization $X \approx U \Sigma V^\top$, where each row of the matrix U is a k-dimensional description of each stance marker, and each row of V is a k-dimensional description of each subreddit. We included the 7,589 subreddits that received at least 1,000 comments in 2014.

5.2 Results: Stance Dimensions

From the SVD analysis, we extracted the six principal latent dimensions that explain the most variation in our dataset.[7] The decision to include only the first six dimensions was based on the strength of the singular values corresponding to the dimensions. Table 3 shows the top five stance markers for each extreme of the six dimensions. The stance dimensions convey a range of concepts, such as involved versus informational language, narrative versus dialogue-oriented writing, standard versus non-standard variation, and positive versus negative affect. Figure 1 shows the distribution of subreddits along two of these dimensions.

[7] Similar to factor analysis, the top few dimensions of SVD explain the most variation, and tend to be most interpretable. A scree plot (Cattell, 1966) showed that the amount of variation explained dropped after the top six dimensions, and qualitative interpretation showed that the remaining dimension were less interpretable.

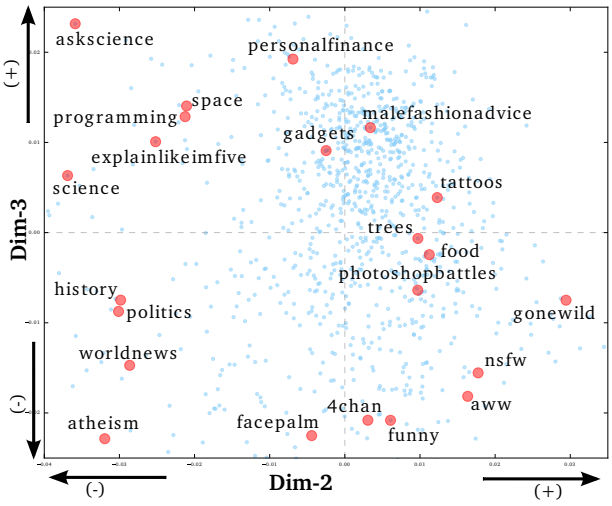


Figure 1: Mapping of subreddits in dimension two and dimension three, highlighting especially popular subreddits. Picture-oriented subreddits *r/gonewild* and *r/aww* map high on dimension two and low on dimension three, indicating involved and informal style of discourse. Subreddits dedicated for knowledge sharing discussions such as *r/askscience* and *r/space* map low on dimension two and high on dimension three indicating informational and formal style.

6 Construct Validity

Evaluating model output against gold-standard annotations is appropriate when there is some notion of a correct answer. As stancetaking is a multidimensional concept, we have taken an unsupervised approach. Therefore, we use evaluation techniques based on the notion of *validity*, which is the extent to which the operationalization of a construct truly captures the intended quantity or concept. Validation techniques for unsupervised content analysis are widely found in the social science literature (Weber, 1990; Quinn et al., 2010) and have also been recently used in the NLP and machine learning communities (e.g., Chang et al., 2009; Murphy et al., 2012; Sim et al., 2013).

We used several methods to validate the stance dimensions extracted from the corpus of Reddit comments. This section describes intrinsic evaluations, which test whether the extracted stance dimensions are linguistically coherent and mean-

		Stance markers	Subreddits
Dim-1	-	beautifully, pleased, thanks, spectacular, delightful	philosophy, history, science
	+	just, even, all, no, so	pcmasterrace, leagueoflegends, gaming
Dim-2	-	suggests that, demonstrates, conclude, demonstrated, demonstrate	philosophy, science, askscience,
	+	lovely, awww, hehe, aww, haha	gonewild, nsfw, aww
Dim-3	-	funnier, hilarious, disturbing, creepy, funny	cringe, creepy, cringepics
	+	thanks, ideally, calculate, estimate, calculation	askscience, personalfinance, space
Dim-4	-	phenomenal, bummed, enjoyed, fantastic, disappointing	movies, television, books
	+	hello, thx, hehe, aww, hi	philosophy, 4chan, atheism
Dim-5	-	lovely, stunning, wonderful, delightful, beautifully	gonewild, aww, tattoos
	+	nvm, cmon, smh, lmao, disappointing	nfl, soccer, cringe
Dim-6	-	stunning, fantastic, incredible, amazing, spectacular	philosophy, gonewild, askscience
	+	anxious, stressed, exhausted, overwhelmed, relieved	relationships, sex, nosleep

Table 3: For each of the six dimensions extracted by our method, we show the five markers and three subreddits (among the 100 most popular subreddits) with the highest loadings.

ingful, thereby testing the *construct* or *content validity* of the proposed stance dimensions (Quinn et al., 2010). Extrinsic evaluations are presented in section 7.

6.1 Word Intrusion Task

A word intrusion task is used to measure the coherence and interpretability of a group of words. Human raters are presented with a list of terms, all but one of which are selected from a target concept; their task is to identify the intruder. If the target concept is internally coherent, human raters should be able to perform this task accurately; if not, their selections should be random. Word intrusion tasks have previously been used to validate the interpretability of topic models (Chang et al., 2009) and vector space models (Murphy et al., 2012).

We deployed a word intrusion task on Amazon Mechanical Turk (AMT), in which we presented the top four stance markers from one end of a dimension, along with an intruder marker selected from the top four markers of the opposite end of that dimension. In this way, we created four word intrusion tasks for each end of each dimension. The main reason for including only the top four words in each dimension is the expense of conducting crowd-sourced evaluations. In the most relevant prior work, Chang et al. (2009) used the top five words from each topic in their evaluation of topic models.

Worker selection We required that the AMT workers ("turkers") have completed a minimum of 1,000 HITs and have at least 95% approval rate Furthermore, because our task is based on analysis of English language texts, we required the turkers to be native speakers of English living in one of the majority English speaking countries. As a further requirement, we required the turkers to obtain a qualification which involves an English comprehension test similar to the questions in standardized English language tests. These requirements are based on best practices identified by Callison-Burch and Dredze (2010).

Task specification Each AMT human intelligence task (HIT) consists of twelve word intrusion tasks, one for each end of the six dimensions. We provided minimal instructions regarding the task, and did not provide any examples, to avoid introducing bias.[8] As a further quality control, each HIT included three questions which ask the turkers to pick the best synonym for a given word from a list of five answers, where one answer was clearly correct; Turkers who gave incorrect answers were to be excluded, but this situation did not arise in practice. Altogether each HIT consists of 15 questions, and was paid US$1.50. Five different turkers performed each HIT.

Results We measured the interrater reliability using Krippendorf's α (Krippendorff, 2007) and the *model precision* metric of Chang et al. (2009). Results on both metrics were encouraging. We obtained a value of $\alpha = 0.73$, on a scale where

[8]The prompt for the word intrusions task was: "Select the intruder word/phrase: you will be given a list of five English words/phrases and asked to pick the word/phrase that is least similar to the other four words/phrases when used in online discussion forums."

$\alpha = 0$ indicates chance agreement and $\alpha = 1$ indicates perfect agreement. The model precision was 0.82; chance precision is 0.20. To offer a sense of typical values for this metric, Chang et al. (2009) report model precisions in the range 0.7–0.83 in their analysis of topic models. Overall, these results indicate that the multi-dimensional analysis has succeeded at identifying dimensions that reflect natural groupings of stance markers.

6.2 Pre-registered Hypotheses

Content validity was also assessed using a set of pre-registered hypotheses. The practice of pre-registering hypotheses before an analysis and testing the correctness is widely used in the social sciences; it was adopted by Sim et al. (2013) to evaluate the induction of political ideological models from text. Before performing the muti-dimensional analysis, we identified two groups of hypotheses that are expected to hold with respect to the latent stancetaking dimensions using our prior linguistic knowledge:

- **Hypothesis I:** Stance markers that are synonyms should not appear on the opposite ends of a stance dimension.
- **Hypothesis II:** If at least one stance marker from a predefined *stance feature group* (defined below) appears on one end of a stance dimension, then other markers from the same feature group will tend not to appear at the opposite end of the same dimension.

6.2.1 Synonym Pairs

For each marker in our stance lexicon, we extracted synonyms from Wordnet, focusing on markers that appear in only one Wordnet synset, and not including pairs in which one term was an inflection of the other.[9] Our final list contains 73 synonym pairs (e.g., *eventually/finally*, *grateful/thankful*, *yea/yeah*). Of these pairs, there were 59 cases in which both terms appeared in either the top or bottom 200 positions of a stance dimension. In 51 of these cases (86%), the two terms appeared on the same side of the dimension. The chance rate would be 50%, so this supports Hypothesis I and further validates the stance dimensions. More details of the results are shown in Table 4. Note that synonym pairs may differ in aspects such as formality (e.g., *said/informed*, *want/desire*), which is one of the main dimensions of stancetaking. Therefore, perfect support for Hypothesis I is not expected.

[9]It is possible that inflections are semantically similar, because by definition they are changes in the form of a word to mark distinctions such as tense, person, or number. However, different inflections of a single word form might be used to mark different stances (e.g., some stances might be associated with the past while others might be associated with the present or future).

Stance Dimension	Number of synonym pairs On same end	On opposite ends
DIMENSION 1	6	3
DIMENSION 2	12	2
DIMENSION 3	2	1
DIMENSION 4	11	0
DIMENSION 5	10	2
DIMENSION 6	10	0
Total	51/59	8/59

Table 4: Results for pre-registered hypothesis that stance dimensions will not split synonym pairs.

6.2.2 Stance Feature Groups

Biber and Finegan (1989) group stance markers into twelve "feature groups", such as certainty adverbs, doubt adverbs, affect expressions, and hedges. Ideally, the stance dimensions should preserve these groupings. To test this, for each of the seven feature groups with at least ten stance markers in the lexicon, we counted the number of terms appearing among the top 200 positions in both ends (high/low) of each dimension. Under the null hypothesis, the stance dimensions are random with respect to the feature groups, so we would expect roughly an equal number of markers on both ends. As shown in Table 5, for five of the seven feature groups, it is possible to reject the null hypothesis at $p < .007$, which is the significance threshold at $\alpha = 0.05$, after correcting for multiple comparisons using the Bonferroni correction. This indicates that the stance dimensions are aligned with predefined stance feature groups.

7 Extrinsic Evaluations

The evaluations in the previous section test internal validity; we now describe evaluations testing whether the stance dimensions are relevant to external social and interactional phenomena.

7.1 Predicting Cross-posting

Online communities can be considered as *communities of practice* (Eckert and McConnell-Ginet, 1992), where members come together to engage in shared linguistic practices. These practices

Feature group	#Stance marker	χ^2	p-value	Reject null?
Certainty adv.	38	16.94	$4.6e^{-03}$	✓
Doubt adv.	23	13.21	$2.2e^{-02}$	×
Certainty verbs	36	48.99	$2.2e^{-09}$	✓
Doubt verbs	55	30.45	$1.2e^{-05}$	✓
Certainty adj.	28	29.73	$1.7e^{-05}$	✓
Doubt adj.	12	14.80	$1.1e^{-02}$	×
Affect exp.	227	97.17	$2.1e^{-19}$	✓

Table 5: Results for preregistered hypothesis that stance dimensions will align with stance feature groups of Biber and Finegan (1989).

evolve simultaneously with membership, coalescing into shared norms. The memberships of multiple subreddits on the same topic (e.g., *r/science* and *r/askscience*) often do not overlap considerably. Therefore we hypothesize that users of Reddit have preferred interactional styles, and that participation in subreddit communities is governed not only by topic interest, but also by these interactional preferences. The proposed stancetaking dimensions provide a simple measure of interactional style, allowing us to test whether it is predictive of community membership decisions.

Classification task We design a classification task, in which the goal is to determine whether a pair of subreddits is *high-crossover* or *low-crossover*. In high-crossover subreddit pairs, individuals are especially likely to participate in both. For the purpose of this evaluation, individuals are considered to participate in a subreddit if they contribute posts or comments. We compute the pointwise mutual information (PMI) with respect to cross-participation among the 100 most popular subreddits. For each subreddit s, we identify the five highest and lowest PMI pairs $\langle s, t \rangle$, and add these to the high-crossover and low-crossover sets, respectively. Example pairs are shown in Table 6. After eliminating redundant pairs, we identify 437 unique high-crossover pairs, and 465 unique low-crossover pairs. All evaluations are based on multiple random training/test splits over this dataset.

Classification approaches A simple classification approach is to predict that subreddits with similar text will have high crossover. We measure similarity using TF-IDF weighted cosine similarity, using two possible lexicons: the 8,000 most frequent words on reddit (BOW), and the stance lexicon (STANCE MARKERS). The similarity threshold between high-crossover and low-crossover pairs was estimated on the training data. We also tested the relevance of multi-dimensional analysis, by applying SVD to both lexicons. For each pair of subreddits, we computed a feature set of the absolute difference across the top six latent dimensions, and applied a logistic regression classifier. Regularization was tuned by internal cross-validation.

Cross-Community Participation	
High-Scoring Pairs	**Low-Scoring Pairs**
r/blog, r/announcements	r/gonewild, r/leagueoflegends
r/pokemon, r/wheredidthesodago	r/soccer, r/nosleep
r/politics, r/technology	r/programming, r/gonewild
r/LifeProTips, r/dataisbeautiful	r/nfl, r/leagueoflegends
r/Unexpected, r/JusticePorn	r/Minecraft, r/personalfinance

Table 6: Examples of subreddit pairs that have large and small amount of overlap of contributing members.

	Cosine	SVD
BOW	66.13%	77.48%
STANCE MARKERS	64.31%	84.93%

Table 7: Accuracy for prediction of subreddit cross-participation.

Results Table 7 shows average accuracies for these models. The stance-based SVD features are considerably more accurate than the BOW-based SVD features, indicating that interactional style does indeed predict cross-posting behavior.[10] Both are considerably more accurate than the bag-of-words models based on cosine similarity.

7.2 Politeness and Formality

The utility of the induced stance dimensions depends on their correlation with social phenomena of interest. Prior work has used crowdsourcing to annotate texts for politeness and formality. We now evaluate the stancetaking properties of these annotated texts.

Data We used the politeness corpus of Wikipedia edit requests from Danescu-Niculescu-Mizil et al. (2013), which includes the textual content of the edit requests, along with scalar annotations of politeness. Following the original

[10] We use BOW+SVD as the most comparable content-based alternative to our stancetaking dimensions. While there may be more accurate discriminative approaches, our goal is a direct comparison of stance and content-based features, not an exhaustive comparison of classification approaches.

authors, we compare the text for the messages ranked in the first and fourth quartiles of politeness scores. For formality, we used the corpus from Pavlick and Tetreault (2016), focusing on the blogs domain, which is most similar to our domain of Reddit. Each sentence in this corpus was annotated for formality levels from -3 to $+3$. We considered only the sentences with mean formality score greater than $+1$ (more formal) and less than -1 (less formal).

Stance dimensions For each document in the above datasets, we compute the stance properties, as follows: for each dimension, we compute the total frequency of the hundred most positive terms and the hundred most negative terms, and then take the difference. Instances containing no terms from either list are excluded. We focus on stance dimensions two and five (summarized in Table 3), because they appeared to be most relevant to politeness and formality. Dimension two contrasts informational and argumentative language against emotional and non-standard language. Dimension five contrasts positive and formal language against non-standard and somewhat negative language.

Results A kernel density plot of the resulting differences is shown in Figure 2. The effect sizes of the resulting differences are quantified using Cohen's d statistic (Cohen, 1988). Effect sizes for all differences are between 0.3 and 0.4, indicating small-to-medium effects — except for the evaluation of formality on dimension five, where the effect size is close to zero. The relatively modest effect sizes are unsurprising, given the short length of the texts. However, these differences lend insight to the relationship between formality and politeness, which may seem to be closely related concepts. On dimension two, it is possible to be polite while using non-standard language such as *hehe* and *awww*, so long as the sentiment expressed is positive; however, these markers are not consistent with formality. On dimension five, we see that positive sentiment terms such as *lovely* and *stunning* are consistent with politeness, but not with formality. Indeed, the distribution of dimension five indicates that both ends of dimension five are consistent only with informal texts.

Overall, these results indicate that interactional phenomena such as politeness and formality are reflected in our stance dimensions, which are induced in an unsupervised manner. Future work may consider the utility of these stance dimensions to predict these social phenomena, particularly in cross-domain settings where lexical classifiers may overfit.

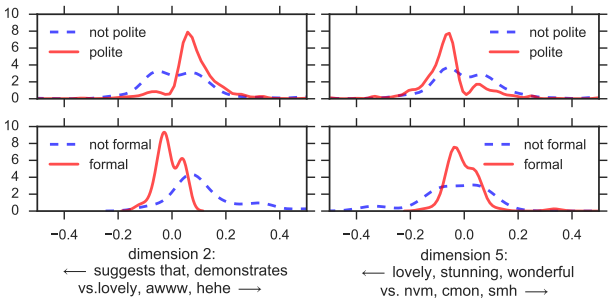


Figure 2: Kernel density distributions for stance dimensions 2 and 5, plotted with respect to annotations of politeness and formality.

8 Conclusion

Stancetaking provides a general perspective on the various linguistic phenomena that structure social interactions. We have identified a set of several hundred stance markers, building on previously-identified lexicons by using word embeddings to perform lexicon expansion. We then used multi-dimensional analysis to group these markers into stance dimensions, which we show to be internally coherent and extrinsically useful. Our hope is that these stance dimensions will be valuable as a convenient building block for future research on interactional meaning.

Acknowledgments Thanks to the anonymous reviewers for their useful and constructive feedback on our submission. This research was supported by Air Force Office of Scientific Research award FA9550-14-1-0379, by National Institutes of Health award R01-GM112697, and by the National Science Foundation awards 1452443 and 1111142. We thank Tyler Schnoebelen for helpful discussions; C.J. Hutto, Tanushree Mitra, and Sandeep Soni for assistance with Mechanical Turk experiments; and Ian Stewart for assistance with creating word embeddings. We also thank the Mechanical Turk workers for performing the word intrusion task, and for feedback on a pilot task.

References

Lada A. Adamic, Jun Zhang, Eytan Bakshy, and Mark S. Ackerman. 2008. Knowledge sharing and yahoo answers: Everyone knows something. In

Proceedings of the Conference on World-Wide Web (WWW). pages 665–674.

Shlomo Argamon, Moshe Koppel, James W. Pennebaker, and Jonathan Schler. 2007. Mining the blogosphere: Age, gender and the varieties of self-expression. *First Monday* 12(9).

Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. 2006. Group formation in large social networks: Membership, growth, and evolution. In *Proceedings of Knowledge Discovery and Data Mining (KDD)*. pages 44–54.

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2014. Gender identity and lexical variation in social media. *Journal of Sociolinguistics* 18(2):135–160.

Douglas Biber. 1991. *Variation across speech and writing*. Cambridge University Press.

Douglas Biber. 1992. The multi-dimensional approach to linguistic analyses of genre variation: An overview of methodology and findings. *Computers and the Humanities* 26(5-6):331–345.

Douglas Biber and Edward Finegan. 1989. Styles of stance in english: Lexical and grammatical marking of evidentiality and affect. *Text* 9(1):93–124.

Su Lin Blodgett, Lisa Green, and Brendan OConnor. 2016. Demographic dialectal variation in social media: A case study of african-american english. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*. pages 1119–1130.

M. Bucholtz and K. Hall. 2005. Identity and interaction: A sociocultural linguistic approach. *Discourse studies* 7(4-5):585–614.

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods* 39(3):510–526.

John D. Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*. pages 1301–1309.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, pages 1–12.

Raymond B Cattell. 1966. The scree test for the number of factors. *Multivariate behavioral research* 1(2):245–276.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Neural Information Processing Systems (NIPS)*. Vancouver, pages 288–296.

Jacob Cohen. 1988. *Statistical power analysis for the behavioral sciences*. Lawrence Earlbaum Associates, Hillsdale, NJ.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the Association for Computational Linguistics (ACL)*. Sophia, Bulgaria, pages 250–259.

John W. Du Bois. 2007. The stance triangle. In Robert Engelbretson, editor, *Stancetaking in discourse*, John Benjamins Publishing Company, Amsterdam/Philadelphia, pages 139–182.

Penelope Eckert. 2012. Three waves of variation study: the emergence of meaning in the study of sociolinguistic variation. *Annual Review of Anthropology* 41:87–100.

Penelope Eckert and Sally McConnell-Ginet. 1992. Think practically and look locally: Language and gender as community-based practice. *Annual review of anthropology* 21:461–490.

Penelope Eckert and John R Rickford. 2001. *Style and sociolinguistic variation*. Cambridge University Press.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*. pages 359–369.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proceedings of the International Conference on Machine Learning (ICML)*. pages 1041–1048.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*. pages 1277–1287.

Valerie Freeman, Richard Wright, Gina-Anne Levow, Yi Luan, Julian Chan, Trang Tran, Victoria Zayats, Maria Antoniak, and Mari Ostendorf. 2014. Phonetic correlates of stance-taking. *The Journal of the Acoustical Society of America* 136(4):2175–2175.

Eric Gilbert. 2013. Widespread underprovision on reddit. In *Proceedings of Computer-Supported Cooperative Work (CSCW)*. pages 803–808.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*. pages 595–605.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the Association for Computational Linguistics (ACL)*. Madrid, Spain, pages 174–181.

Jack Hessel, Chenhao Tan, and Lillian Lee. 2014. Science, askscience, and badscience: On the coexistence of highly related communities. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*. AAAI Publications, Menlo Park, California, pages 171–180.

Alexandra Jaffe. 2009. *Stance: Sociolinguistic Perspectives*. Oxford University Press.

Daniel Jurafsky, Elizabeth Shriberg, Barbara Fox, and Traci Curl. 1998. Lexical, prosodic, and syntactic cues for dialog acts. In *Proceedings of ACL/COLING-98 Workshop on Discourse Relations and Discourse Markers*. pages 114–120.

Elise Kärkkäinen. 2006. Stance taking in conversation: From subjectivity to intersubjectivity. *Text & Talk-An Interdisciplinary Journal of Language, Discourse Communication Studies* 26(6):699–731.

Tiina Keisanen. 2007. Stancetaking as an interactional activity: Challenging the prior speaker. *Stancetaking in discourse: Subjectivity, evaluation, interaction* pages 253–81.

Scott Fabius Kiesling. 2009. Style as stance. *Stance: sociolinguistic perspectives* pages 171–194.

Klaus Krippendorff. 2007. Computing krippendorff's alpha reliability. *Departmental papers (ASC)* page 43.

Thomas Landauer, Peter W. Foltz, and Darrel Laham. 1998. Introduction to latent semantic analysis. *Discource Processes* 25:259–284.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*. Denver, CO, pages 1299–1304.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. pages 3111–3119.

Brian Murphy, Partha Pratim Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of the International Conference on Computational Linguistics (COLING)*. Mumbai, India, pages 1933–1949.

Dong Nguyen, A Seza Doğruöz, Carolyn P Rosé, and Franciska de Jong. 2016. Computational sociolinguistics: A survey. *Computational Linguistics* 42(3):537–593.

Dong Nguyen, Rilana Gravel, Dolf Trieschnigg, and Theo Meder. 2013. "How Old Do You Think I Am?" A Study of Language and Age in Twitter. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*. pages 439–448.

Yoshiki Niwa and Yoshihiko Nitta. 1994. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of the International Conference on Computational Linguistics (COLING)*. Kyoto, Japan, pages 304–309.

Elinor Ochs. 1993. Constructing social identity: A language socialization perspective. *Research on language and social interaction* 26(3):287–306.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics (TACL)* 4:61–74.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of LIWC2015. Technical report.

Vinodkumar Prabhakaran, Owen Rambow, and Mona Diab. 2012. Predicting overt display of power in written dialogs. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*. pages 518–522.

Kristen Precht. 2003. Stance moods in spoken english: Evidentiality and affect in british and american conversation. *Text - Interdisciplinary Journal for the Study of Discourse* 23(2):239–258.

Kevin M Quinn, Burt L Monroe, Michael Colaresi, Michael H Crespin, and Dragomir R Radev. 2010. How to analyze political attention with minimal assumptions and costs. *American Journal of Political Science* 54(1):209–228.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*. pages 105–112.

Yanchuan Sim, Brice Acree, Justin H Gross, and Noah A Smith. 2013. Measuring ideological proportions in political speeches. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Philip J. Stone. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press.

Chenhao Tan and Lillian Lee. 2015. All who wander: On the prevalence and characteristics of multi-community engagement. In *Proceedings of the Conference on World-Wide Web (WWW)*. pages 1056–1066.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology* 29(1):24–54.

Trang Tran and Mari Ostendorf. 2016. Characterizing the language of online communities and its relation to community reception. In *Proceedings of*

Empirical Methods for Natural Language Processing (EMNLP).

Marilyn A Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*. pages 592–596.

Robert Philip Weber. 1990. *Basic content analysis*. 49. Sage.

Peter RR White. 2003. Beyond modality and hedging: A dialogic view of the language of intersubjective stance. *Text - Interdisciplinary Journal for the Study of Discourse* 23(2):259–284.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation* 39(2):165–210.

Tandem Anchoring: a Multiword Anchor Approach for Interactive Topic Modeling

Jeffrey Lund, Connor Cook, Kevin Seppi
Computer Science Department
Brigham Young University
{jefflund,cojoco,kseppi}@byu.edu

Jordan Boyd-Graber
Computer Science Department
University of Colorado Boulder
jordan.boyd.graber@colorado.edu

Abstract

Interactive topic models are powerful tools for understanding large collections of text. However, existing sampling-based interactive topic modeling approaches scale poorly to large data sets. Anchor methods, which use a single word to uniquely identify a topic, offer the speed needed for interactive work but lack both a mechanism to inject prior knowledge and lack the intuitive semantics needed for user-facing applications. We propose combinations of words as anchors, going beyond existing single word anchor algorithms—an approach we call "Tandem Anchors". We begin with a synthetic investigation of this approach then apply the approach to interactive topic modeling in a user study and compare it to interactive and non-interactive approaches. Tandem anchors are faster and more intuitive than existing interactive approaches.

Topic models distill large collections of text into topics, giving a high-level summary of the thematic structure of the data without manual annotation. In addition to facilitating discovery of topical trends (Gardner et al., 2010), topic modeling is used for a wide variety of problems including document classification (Rubin et al., 2012), information retrieval (Wei and Croft, 2006), author identification (Rosen-Zvi et al., 2004), and sentiment analysis (Titov and McDonald, 2008). However, the most compelling use of topic models is to help users understand large datasets (Chuang et al., 2012).

Interactive topic modeling (Hu et al., 2014) allows non-experts to refine automatically generated topics, making topic models less of a "take it or leave it" proposition. Including humans input during training improves the quality of the model and allows users to guide topics in a specific way, custom tailoring the model for a specific downstream task or analysis.

The downside is that interactive topic modeling is slow—algorithms typically scale with the size of the corpus—and requires non-intuitive information from the user in the form of must-link and cannot-link constraints (Andrzejewski et al., 2009). We address these shortcomings of interactive topic modeling by using an interactive version of the *anchor words* algorithm for topic models.

The anchor algorithm (Arora et al., 2013) is an alternative topic modeling algorithm which scales with the number of unique word types in the data rather than the number of documents or tokens (Section 1). This makes the anchor algorithm fast enough for interactive use, even in web-scale document collections.

A drawback of the anchor method is that anchor words—words that have high probability of being in a *single* topic—are not intuitive. We extend the anchor algorithm to use multiple anchor words in tandem (Section 2). Tandem anchors not only improve interactive refinement, but also make the underlying anchor-based method more intuitive.

For interactive topic modeling, tandem anchors produce higher quality topics than single word anchors (Section 3). Tandem anchors provide a framework for fast interactive topic modeling: users improve and refine an existing model through multiword anchors (Section 4). Compared to existing methods such as Interactive Topic Models (Hu et al., 2014), our method is much faster.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 896–905
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1083

1 Vanilla Anchor Algorithm

The anchor algorithm computes the topic matrix $\boldsymbol{A}$, where $\boldsymbol{A}_{v,k}$ is the conditional probability of observing word v given topic k, e.g., the probability of seeing the word "lens" given the camera topic in a corpus of Amazon product reviews. Arora et al. (2012a) find these probabilities by assuming that every topic contains at least one 'anchor' word which has a non-zero probability only in that topic. Anchor words make computing the topic matrix $\boldsymbol{A}$ tractable because the occurrence pattern of the anchor word mirrors the occurrence pattern of the topic itself.

To recover the topic matrix $\boldsymbol{A}$ using anchor words, we first compute a $V \times V$ cooccurrence matrix $\boldsymbol{Q}$, where $\boldsymbol{Q}_{i,j}$ is the conditional probability $p(w_j \mid w_i)$ of seeing word type w_j after having seen w_i in the same document. A form of the Gram-Schmidt process on $\boldsymbol{Q}$ finds anchor words $\{g_1 \dots g_k\}$ (Arora et al., 2013).

Once we have the set of anchor words, we can compute the probability of a topic given a word (the inverse of the conditioning in $\boldsymbol{A}$). This coefficient matrix $\boldsymbol{C}$ is defined row-wise for each word i

$$\boldsymbol{C}^*_{i,\cdot} = \underset{C_{i,\cdot}}{\operatorname{argmin}} D_{KL}\left(\boldsymbol{Q}_{i,\cdot} \middle\| \sum_{k=1}^{K} \boldsymbol{C}_{i,k}\boldsymbol{Q}_{g_k,\cdot}\right), \tag{1}$$

which gives the best reconstruction (based on Kullback-Leibler divergence D_{KL}) of non-anchor words given anchor words' conditional probabilities. For example, in our product review data, a word such as "battery" is a convex combination of the anchor words' contexts ($\boldsymbol{Q}_{g_k,\cdot}$) such as "camera", "phone", and "car". Solving each row of C is fast and is embarrassingly parallel. Finally, we apply Bayes' rule to recover the topic matrix $\boldsymbol{A}$ from the coefficient matrix $\boldsymbol{C}$.

The anchor algorithm can be orders of magnitude faster than probabilistic inference (Arora et al., 2013). The construction of $\boldsymbol{Q}$ has a runtime of $O(DN^2)$ where D is the number of documents and N is the average number of tokens per document. This computation requires only a single pass over the data and can be pre-computed for interactive use-cases. Once $\boldsymbol{Q}$ is constructed, topic recovery requires $O(KV^2 + K^2VI)$, where K is the number of topics, V is the vocabulary size, and I is the average number of iterations (typically 100-1000). In contrast, traditional topic model inference typically requires multiple passes over the entire data. Techniques such as Online LDA (Hoffman et al., 2010) or Stochastic Variation Inference (Hoffman et al., 2013) improves this to a single pass over the entire data. However, from Heaps' law (Heaps, 1978) it follows that $V^2 \ll DN$ for large datasets, leading to much faster inference times for anchor methods compared to probabilistic topic modeling. Further, even if online were to be adapted to incorporate human guidance, a single pass is not tractable for interactive use.

Anchor	Top Words in Topics
backpack	backpack camera lens bag room carry fit cameras equipment comfortable
camera	camera lens pictures canon digital lenses batteries filter mm photos
bag	bag camera diaper lens bags genie smell room diapers odor

Table 1: Three separate attempts to construct a topic concerning camera bags in Amazon product reviews with single word anchors. This example is drawn from preliminary experiments with an author as the user. The term "backpack" is a good anchor because it uniquely identifies the topic. However, both "camera" and "bag" are poor anchors for this topic.

2 Tandem Anchor Extension

Single word anchors can be opaque to users. For an example of bewildering anchor words, consider a camera bag topic from a collection of Amazon product reviews (Table 1). The anchor word "backpack" may seem strange. However, this dataset contains nothing about regular backpacks; thus, "backpack" is unique to camera bags. Bizarre, low-to-mid frequency words are often anchors because anchor words must be *unique* to a topic; intuitive or high-frequency words cannot be anchors if they have probability in *any other topic*.

The anchor selection strategy can mitigate this problem to some degree. For example, rather than selecting anchors using an approximate convex hull in high-dimensional space, we can find an exact convex hull in a low-dimensional embedding (Lee and Mimno, 2014). This strategy will produce more salient topics but still makes it difficult for users to manually choose unique anchor words for interactive topic modeling.

If we instead ask users to give us representative

words for this topic, we would expect combinations of words like "camera" and "bag." However, with single word anchors we must choose a single word to anchor each topic. Unfortunately, because these words might appear in multiple topics, individually they are not suitable as anchor words. The anchor word "camera" generates a general camera topic instead of camera bags, and the topic anchored by "bag" includes bags for diaper pails (Table 1).

Instead, we need to use sets of representative terms as an interpretable, parsimonious description of a topic. This section discusses strategies to build anchors from multiple words and the implications of using multiword anchors to recover topics. This extension not only makes anchors more interpretable but also enables users to manually construct effective anchors in interactive topic modeling settings.

2.1 Anchor Facets

We first need to turn words into an anchor. If we interpret the anchor algorithm geometrically, each row of $\boldsymbol{Q}$ represents a word as a point in V-dimensional space. We then model each point as a convex combination of anchor words to reconstruct the topic matrix $\mathbf{A}$ (Equation 1). Instead of individual anchor words (one anchor word per topic), we use anchor **facets**, or sets of words that describe a topic. The facets for each anchor form a new **pseudoword**, or an invented point in V-dimensional space (described in more detail in Section 2.2).

While these new points do not correspond to words in the vocabulary, we can express non-anchor words as convex combinations of pseudowords. To construct these pseudowords from their facets, we combine the co-occurrence profiles of the facets. These pseudowords then augment the original cooccurrence matrix $\boldsymbol{Q}$ with K additional rows corresponding to synthetic pseudowords forming each of K multiword anchors. We refer to this augmented matrix as $\boldsymbol{S}$. The rest of the anchor algorithm proceeds unmodified.

Our augmented matrix $\boldsymbol{S}$ is therefore a $(V + K) \times V$ matrix. As before, V is the number of token types in the data and K is the number of topics. The first V rows of $\boldsymbol{S}$ correspond to the V token types observed in the data, while the additional K rows correspond to the pseudowords constructed from anchor facets. Each entry of $\boldsymbol{S}$ encodes conditional probabilities so that $S_{i,j}$ is equal to $p(w_i \mid w_j)$. For the additional K rows, we invent a cooccurrence pattern that can effectively explain the other words' conditional probabilities.

This modification is similar in spirit to supervised anchor words (Nguyen et al., 2015). This supervised extension of the anchor words algorithm adds columns corresponding to conditional probabilities of metadata values after having seen a particular word. By extending the vector-space representation of each word, anchor words corresponding to metadata values can be found. In contrast, our extension does not add dimensions to the representation, but simply places additional points corresponding to pseudoword words in the vector-space representation.

2.2 Combining Facets into Pseudowords

We now describe more concretely how to combine an anchor facets to describe the cooccurrence pattern of our new pseudoword anchor. In tandem anchors, we create vector representations that combine the information from anchor facets. Our anchor facets are $\mathcal{G}_1 \dots \mathcal{G}_K$, where $\mathcal{G}_k$ is a set of anchor facets which will form the kth pseudoword anchor. The pseudowords are $g_1 \dots g_K$, where g_k is the pseudoword from $\mathcal{G}_k$. These pseudowords form the new rows of $\boldsymbol{S}$. We give several candidates for combining anchors facets into a single multiword anchor; we compare their performance in Section 3.

Vector Average An obvious function for computing the central tendency is the vector average. For each anchor facet,

$$\boldsymbol{S}_{g_k,j} = \sum_{i \in \mathcal{G}_k} \frac{\boldsymbol{S}_{i,j}}{|\mathcal{G}_k|}, \tag{2}$$

where $|\mathcal{G}_k|$ is the cardinality of $\mathcal{G}_k$. Vector average makes the pseudoword $\boldsymbol{S}_{g_k,j}$ more central, which is intuitive but inconsistent with the interpretation from Arora et al. (2013) that anchors should be extreme points whose linear combinations explain more central words.

Or-operator An alternative approach is to consider a cooccurrence with *any* anchor facet in $\mathcal{G}_k$. For word j, we use De Morgan's laws to set

$$\boldsymbol{S}_{g_k,j} = 1 - \prod_{i \in \mathcal{G}_k} (1 - \boldsymbol{S}_{i,j}). \tag{3}$$

Unlike the average, which pulls the pseudoword inward, this or-operator pushes the word outward,

increasing each of the dimensions. Increasing the volume of the simplex spanned by the anchors explains more words.

Element-wise Min Vector average and or-operator are both sensitive to outliers and cannot account for polysemous anchor facets. Returning to our previous example, both "camera" and "bag" are bad anchors for camera bags because they appear in documents discussing other products. However, if both "camera" and "bag" are anchor facets, we can look at an *intersection* of their contexts: words that appear with both. Using the intersection, the cooccurrence pattern of our anchor facet will only include terms relevant to camera bags.

Mathematically, this is an element-wise min operator,

$$\boldsymbol{S}_{g_k,j} = \min_{i \in \mathcal{G}_k} \boldsymbol{S}_{i,j}. \tag{4}$$

This construction, while perhaps not as simple as the previous two, is robust to words which have cooccurrences which are not unique to a single topic.

Harmonic Mean Leveraging the intuition that we should use a combination function which is both centralizing (like vector average) and ignores large outliers (like element-wise min), the final combination function is the element-wise harmonic mean. Thus, for each anchor facet

$$\boldsymbol{S}_{g_k,j} = \sum_{i \in \mathcal{G}_k} \left(\frac{\boldsymbol{S}_{i,j}^{-1}}{|\mathcal{G}_k|} \right)^{-1}. \tag{5}$$

Since the harmonic mean tends towards the lowest values in the set, it is not sensitive to large outliers, giving us robustness to polysemous words.

2.3 Finding Topics

After constructing the pseudowords of $\boldsymbol{S}$ we then need to find the coefficients $\boldsymbol{C}_{i,k}$ which describe each word in our vocabulary as a convex combination of the multiword anchors. Like standard anchor methods, we solve the following for each token type:

$$\boldsymbol{C}^*_{i,\cdot} = \underset{\boldsymbol{C}_{i,\cdot}}{\operatorname{argmin}} D_{KL} \left(\boldsymbol{S}_{i,\cdot} \middle\| \sum_{k=1}^{K} \boldsymbol{C}_{i,k} \boldsymbol{S}_{g_k,\cdot} \right). \tag{6}$$

Finally, we appeal to Bayes' rule, we recover the topic-word matrix $\boldsymbol{A}$ from the coefficients of $\boldsymbol{C}$.

The correctness of the topic recovery algorithm hinges upon the assumption of separability. Separability means that the occurrence pattern across documents of the anchor words across the data mirrors that of the topics themselves. For single word anchors, this has been observed to hold for a wide variety of data (Arora et al., 2012b). With our tandem anchor extension, we make similar assumptions as the vanilla algorithm, except with pseudowords constructed from anchor facets. So long as the occurrence pattern of our tandem anchors mirrors that of the underlying topics, we can use the same reasoning as Arora et al. (2012a) to assert that we can provably recover the topic-word matrix $\boldsymbol{A}$ with all of the same theoretical guarantees of complexity and robustness. Furthermore, we runtime analysis given by Arora et al. (2013) applies to tandem anchors.

If desired, we can also add further robustness and extensibility to tandem anchors by adding regularization to Equation 6. Regularization allows us to add something which is mathematically similar to priors, and has been shown to improve the vanilla anchor word algorithm (Nguyen et al., 2014). We leave the question of the best regularization for tandem anchors as future work, and focus our efforts on solving the problem of interactive topic modeling.

3 High Water Mark for Tandem Anchors

Before addressing interactivity, we apply tandem anchors to real world data, but with anchors gleaned from metadata. Our purpose is twofold. First, we determine which combiner from Section 2.2 to use in our interactive experiments in Section 4 and second, we confirm that well-chosen tandem anchors can improve topics. In addition, we examine the runtime of tandem anchors and compare to traditional model-based interactive topic modeling techniques. We cannot assume that we will have metadata available to build tandem anchors, but we use them here because they provide a high water mark without the variance introduced by study participants.

3.1 Experimental Setup

We use the well-known 20 Newsgroups dataset (20NEWS) used in previous interactive topic modeling work: 18,846 Usenet postings from 20 different newgroups in the early 1990s.[1] We remove the newsgroup headers from each message, which contain the newsgroup names, but otherwise left messages intact with any footers or quotes. We

[1] http://qwone.com/~jason/20Newsgroups/

then remove stopwords and words which appear in fewer than 100 documents or more than 1,500 documents.

To seed the tandem anchors, we use the titles of newsgroups. To build each multiword anchor facet, we split the title on word boundaries and expand any abbreviations or acronyms. For example, the newsgroup title 'comp.os.ms-windows.misc' becomes {"computer", "operating", "system", "microsoft", "windows", "miscellaneous"}. We do not fully specify the topic; the title gives some intuition, but the topic modeling algorithm must still recover the complete topic-word distributions. This is akin to knowing the names of the categories used but nothing else. Critically, the topic modeling algorithm has no knowledge of document-label relationships.

3.2 Experimental Results

Our first evaluation is a classification task to predict documents' newsgroup membership. Thus, we do not aim for state-of-the-art accuracy,[2] but the experiment shows title-based tandem anchors yield topics closer to the underlying classes than Gram-Schmidt anchors. After randomly splitting the data into test and training sets we learn topics from the test data using both the title-based tandem anchors and the Gram-Schmidt single word anchors.[3] For multiword anchors, we use each of the combiner functions from Section 2.2. The anchor algorithm only gives the topic-word distributions and not word-level topic assignments, so we infer token-level topic assignments using LDA Latent Dirichlet Allocation (Blei et al., 2003) with *fixed* topics discovered by the anchor method. We use our own implementation of Gibbs sampling with fixed topics and a symmetric document-topic Dirichlet prior with concentration $\alpha = .01$. Since the topics are fixed, this inference is very fast and can be parallelized on a per-document basis. We then train a hinge-loss linear classifier on the newsgroup labels using Vowpal Wabbit[4] with topic-word pairs as features. Finally, we infer topic assignments in the test data and evaluate the classification using those topic-word features. For both training and test, we exclude words outside the LDA vocabulary.

The topics created from multiword anchor facets are more accurate than Gram-Schmidt topics (Figure 1). This is true regardless of the combiner function. However, harmonic mean is more accurate than the other functions.[5]

Since 20NEWS has twenty classes, accuracy alone does not capture confusion between closely related newsgroups. For example, accuracy penalizes a classifier just as much for labeling a document from 'rec.sport.baseball' with 'rec.sport.hockey' as with 'alt.atheism' despite the similarity between sports newsgroups. Consequently, after building a confusion matrix between the predicted and true classes, external clustering metrics reveal confusion between classes.

The first clustering metric is the adjusted Rand index (Yeung and Ruzzo, 2001), which is akin to accuracy for clustering, as it gives the percentage of correct pairing decisions from a reference clustering. Adjusted Rand index (ARI) also accounts for chance groupings of documents. Next we use F-measure, which also considers pairwise groups, balancing the contribution of false negatives, but without the true negatives. Finally, we use variation of information (VI). This metric measures the amount of information lost by switching from the gold standard labels to the predicted labels (Meilă, 2003). Since we are measuring the amount of information lost, lower variation of information is better.

Based on these clustering metrics, tandem anchors can yield superior topics to those created using single word anchors (Figure 1). As with accuracy, this is true regardless of which combination function we use. Furthermore, harmonic mean produces the least confusion between classes.[5]

The final evaluation is topic coherence by Newman et al. (2010), which measures whether the topics make sense, and correlates with human judgments of topic quality. Given V, the set of the n most probable words of a topic, coherence is

$$\sum_{v_1, v_2 \in V} log \frac{D(v_1, v_2) + \epsilon}{D(v_2)} \quad (7)$$

where $D(v_1, v_2)$ is the co-document frequency of

[2] The best system would incorporate topic features with other features, making it harder to study and understand the topical trends in isolation.

[3] With fixed anchors and data the anchor algorithm is deterministic, so we use random splits instead of the standard train/test splits so that we can compute variance.

[4] http://hunch.net/~vw/

[5] Significant at $p < 0.01/4$ when using two-tailed t-tests with a Bonferroni correction. For each of our evaluations, we verify the normality of our data (D'Agostino and Pearson, 1973) and use two-tailed t-tests with Bonferroni correction to determine whether the differences between the different methods are significant.

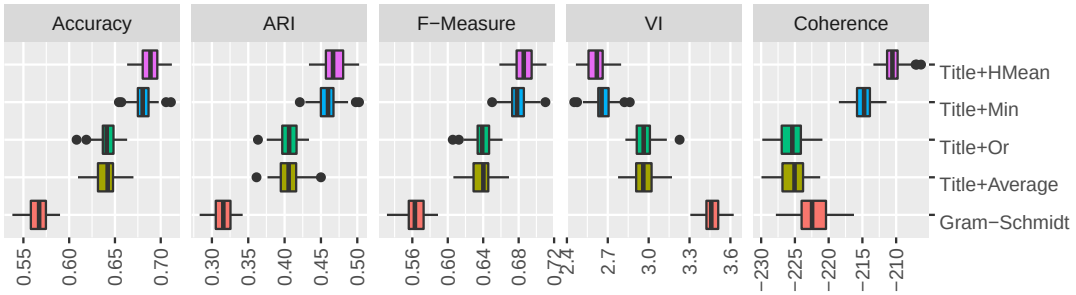


Figure 1: Using metadata can improve anchor-based topic models. For all metrics, the unsupervised Gram-Schmidt anchors do worse than creating anchors based on Newsgroup titles (for all metrics except VI, higher is better). For coherence, Gram-Schmidt does better than two functions for combining anchor words, but not the element-wise min or harmonic mean.

word types v_1 and v_2, and $D(v_2)$ is the document frequency of word type v_2. A smoothing parameter ϵ prevents zero logarithms.

Figure 1 also shows topic coherence. Although title-based anchor facets produce better classification features, topics from Gram-Schmidt anchors have better coherence than title-based anchors with the vector average or the or-operator. However, when using the harmonic mean combiner, title-based anchors produce the most human interpretable topics.[6]

Harmonic mean beats other combiner functions because it is robust to ambiguous or irrelevant term cooccurrences an anchor facet. Both the vector average and the or-operator are swayed by large outliers, making them sensitive to ambiguous terms in an anchor facet. Element-wise min also has this robustness, but harmonic mean is also able to better characterize anchor facets as it has more centralizing tendency than the min.

3.3 Runtime Considerations

Tandem anchors will enable users to direct topic inference to improve topic quality. However, for the algorithm to be interactive we must also consider runtime. Cook and Thomas (2005) argue that for interactive applications with user-initiated actions like ours the response time should be less than ten seconds. Longer waits can increase the cognitive load on the user and harm the user interaction.

[6]Significant at $p < 0.01/4$ when using two-tailed t-tests with a Bonferroni correction. For each of our evaluations, we verify the normality of our data (D'Agostino and Pearson, 1973) and use two-tailed t-tests with Bonferroni correction to determine whether the differences between the different methods are significant.

Fortunately, the runtime of tandem anchors is amenable to interactive topic modeling. On 20NEWS, interactive updates take a median time of 2.13 seconds. This result was obtained using a single core of an AMD Phemon II X6 1090T processor. Furthermore, larger datasets typically have a sublinear increase in distinct word types, so we can expect to see similar run times, even on much larger datasets.

Compared to other interactive topic modeling algorithms, tandem anchors has a very attractive run time. For example, using an optimized version of the sampler for the Interactive Topic Model described by Hu and Boyd-Graber (2012), and the recommended 30 iterations of sampling, the Interactive Topic Model updates with a median time of 24.8 seconds (Hu and Boyd-Graber, 2012), which is well beyond our desired update time for interactive use and an order of magnitude slower than tandem anchors.

Another promising interactive topic modeling approach is Utopian (Choo et al., 2013), which uses non-negative factorization, albeit without the benefit of anchor words. Utopian is much slower than tandem anchors. Even on the small InfoVis-VAST dataset which contains only 515 documents, Utopian takes 48 seconds to converge. While the times are not strictly comparable due to differing datasets, Utopian scales linearly with the size of the data, we can intuit that even for moderately sized datasets such as 20NEWS, Utopian is infeasible for interactive topic modeling due to run time.

While each of these interactive topic modeling algorithms do achieve reasonable topics, only our algorithm fits the run time requirements for inter-

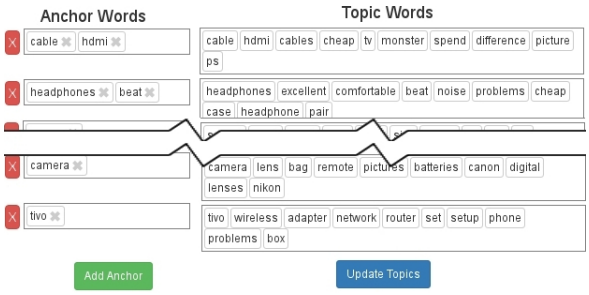


Figure 2: Interface for user study with multiword anchors applied to interactive topic modeling.

activity. Furthermore, since tandem anchors scales with the size of the vocabulary rather than the size of the data, this trend will only become more pronounced as we increase the amount of data.

4 Interactive Anchor Words

Given high quality anchor facets, the tandem anchor algorithm can produce high quality topic models (particularly when the harmonic mean combiner is used). Moreover, the tandem anchor algorithm is fast enough to be interactive (as opposed to model-based approaches such as the Interactive Topic Model). We now turn our attention to our main experiment: tandem anchors applied to the problem of interactive topic modeling. We compare both single word and tandem anchors in our study. We do not include the Interactive Topic Model or Utopian, as their run times are too slow for our users.

4.1 Interface and User Study

To show that interactive tandem anchor words are fast, effective, and intuitive, we ask users to understand a dataset using the anchor word algorithm. For this user study, we recruit twenty participants drawn from a university student body. The student median age is twenty-two. Seven are female, and thirteen are male. None of the students had any prior familiarity with topic modeling or the 20NEWS dataset.

Each participant sees a simple user interface (Figure 2) with topic given as a row with two columns. The left column allows users to view and edit topics' anchor words; the right column lists the most probable words in each topic.[7] The user can remove an anchor word or drag words from the topic word lists (right column) to become an anchor word. Users can also add additional topics by clicking the "Add Anchor" to create additional anchors. If the user wants to add a word to a tandem anchor set that does not appear in the interface, they manually type the word (restricted to the model's vocabulary). When the user wants to see the updated topics for their newly refined anchors, they click "Update Topics".

We give each a participant a high level overview of topic modeling. We also describe common problems with topic models including intruding topic words, duplicate topics, and ambiguous topics. Users are instructed to use their best judgement to determine if topics are useful. The task is to edit the anchor words to improve the topics. We asked that users spend at least twenty minutes, but no more than thirty minutes. We repeat the task twice: once with tandem anchors, and once with single word anchors.[8]

4.2 Quantitative Results

We now validate our main result that for interactive topic modeling, tandem anchors yields better topics than single word anchors. Like our title-based experiments in Section 3, topics generated from users become features to train and test a classifier for the 20NEWS dataset. We choose this dataset for easier comparison with the Interactive Topic Modeling result of Hu et al. (2014). Basedsie on our results with title-based anchors, we use the harmonic mean combiner in our analysis. As before, we report not only accuracy, but also multiple clustering metrics using the confusion matrix from the classification task. Finally, we report topic coherence.

Figure 3 summarizes the results of our quantitative evaluation. While we only compare user generated anchors in our analysis, we include the unsupervised Gram-Schmidt anchors as a baseline. Some of the data violate assumptions of normality. Therefore, we use Wilcoxon's signed-rank test (Wilcoxon, 1945) to determine if the differences between multiword anchors and single word anchors are significant.

Topics from user generated multiword anchors yield higher classification accuracy (Figure 3). Not only is our approach more scalable than the Interactive Topic Model, but we also achieve

[7] While we use topics generated using harmonic mean for our final analysis, users were shown topics generated using the min combiner. However, this does not change our result.

[8] The order in which users complete these tasks is counterbalanced.

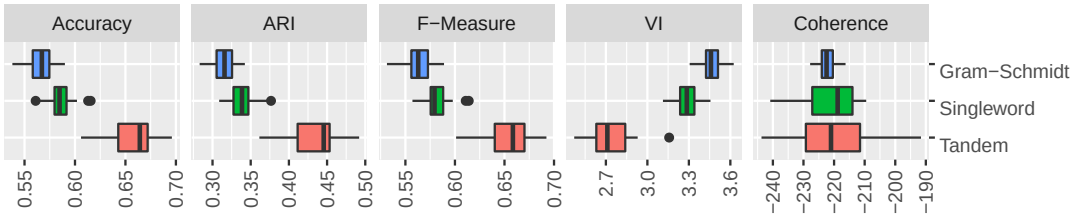


Figure 3: Classification accuracy and coherence using topic features gleaned from user provided multiword and single word anchors. Grahm-Schmidt anchors are provided as a baseline. For all metrics except VI, higher is better. Except for coherence, multiword anchors are best.

higher classification accuracy than Hu et al. (2014).[9] Tandem anchors also improve clustering metrics.[10]

While user selected tandem anchors produce better classification features than single word anchors, users selected single word anchors produce topics with similar topic coherence scores.[11]

To understand this phenomenon, we use quality metrics (AlSumait et al., 2009) for ranking topics by their correspondence to genuine themes in the data. Significant topics are likely skewed towards a few related words, so we measure the distance of each topic-word distribution from the **uniform** distribution over words. Topics which are close to the underlying word distribution of the entire data are likely to be **vacuous**, so we also measure the distance of each topic-word distribution from the underlying word distribution. Finally, **background** topics are likely to appear in a wide range of documents, while meaningful topics will appear in a smaller subset of the data.

Figure 4 reports our topic significance findings. For all three significance metrics, multiword anchors produce more significant topics than single word anchors.[10] Topic coherence is based solely on the top n words of a topic, while both accuracy and topic significance depend on the entire topic-word distributions. With single word anchors, topics with good coherence may still be too general. Tandem anchors enables users to produce topics with more specific word distributions which are better features for classification.

Anchor	Top Words in Topic
Automatic Gram Schmidt	
love	love god evolution romans heard car
game	game games team hockey baseball heard
Interactive Single-word	
evolution	evolution theory science faith quote facts
religion	religion god government state jesus israel
baseball	baseball games players word teams car
hockey	hockey team play games season players
Interactive Tandem	
atheism god exists prove	god science evidence reason faith objective
christian jesus	jesus christian christ church bible christians
jew israel	israel jews jewish israeli state religion
baseball bat ball	hit baseball ball player games call
hockey nhl	team hockey player nhl win play

Table 2: Comparison of topics generated for 20NEWS using various types of anchor words. Users are able to combine words to create more specific topics with tandem anchors.

4.3 Qualitative Results

We examine the qualitative differences between how users select multiword anchor facets versus single word anchors. Table 2 gives examples of topics generated using different anchor strategies. In a follow-up survey with our users, 75% find it easier to affect individual changes in the topics using tandem anchors compared to single word anchors. Users who prefer editing multiword anchors over single word anchors often report that

[9] However, the values are not strictly comparable, as Hu et al. (2014) use the standard chronological test/train fold, and we use random splits.

[10] Significant at $p < 0.01$ when using Wilcoxon's signed-rank test.

[11] The difference between coherence scores was *not* statistically significant using Wilcoxon's signed-rank test.

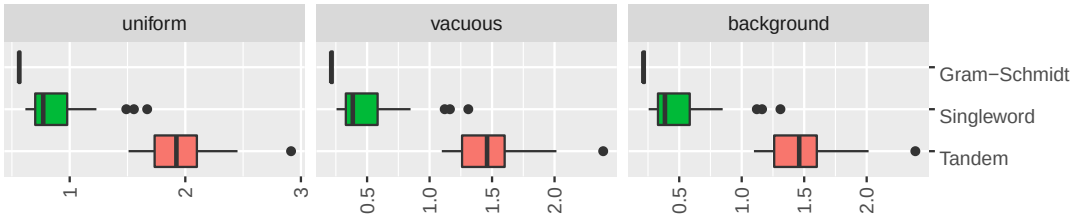


Figure 4: Topic significance for both single word and multiword anchors. In all cases higher is better. Multiword anchors produce topics which are more significant than single word anchors.

multiword anchors make it easier to merge similar topics into a single focused topic by combining anchors. For example, by combining multiple words related to Christianity, users were able to create a topic which is highly specific, and differentiated from general religion themes which included terms about Atheism and Judaism.

While users find that use tandem anchors is easier, only 55% of our users say that they prefer the final topics produced by tandem anchors compared to single word anchors. This is in harmony with our quantitative measurements of topic coherence, and may be the result of our stopping criteria: when users judged the topics to be useful.

However, 100% of our users feel that the topics created through interaction were better than those generated from Gram-Schmidt anchors. This was true regardless of whether we used tandem anchors or single word anchors.

Our participants also produce fewer topics when using multiword anchors. The mean difference between topics under single word anchors and multiple word anchors is 9.35. In follow up interviews, participants indicate that the easiest way to resolve an ambiguous topic with single word anchors was to create a new anchor for each of the ambiguous terms, thus explaining the proliferation of topics for single word anchors. In contrast, fixing an ambiguous tandem anchor is simple: users just add more terms to the anchor facet.

5 Conclusion

Tandem anchors extend the anchor words algorithm to allow multiple words to be combined into anchor facets. For interactive topic modeling, using anchor facets in place of single word anchors produces higher quality topic models and are more intuitive to use. Furthermore, our approach scales much better than existing interactive topic modeling techniques, allowing interactivity on large datasets for which interactivity was previous impossible.

Acknowledgements

This work was supported by the collaborative NSF Grant IIS-1409287 (UMD) and IIS- 1409739 (BYU). Boyd-Graber is also supported by NSF grants IIS-1320538 and NCSE-1422492.

References

Loulwah AlSumait, Daniel Barbará, James Gentle, and Carlotta Domeniconi. 2009. Topic significance ranking of LDA generative models. In *Proceedings of European Conference of Machine Learning*.

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of the International Conference of Machine Learning*.

Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2013. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of the International Conference of Machine Learning*.

Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. 2012a. Computing a nonnegative matrix factorization–provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*.

Sanjeev Arora, Rong Ge, and Ankur Moitra. 2012b. Learning topic models–going beyond svd. In *Fifty-Third IEEE Annual Symposium on Foundations of Computer Science*.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Jaegul Choo, Changhyun Lee, Chandan K Reddy, and Heejung Park. 2013. Utopian: User-driven topic modeling based on interactive nonnegative matrix factorization. *Visualization and Computer Graphics, IEEE Transactions on* 19(12):1992–2001.

Jason Chuang, Christopher D Manning, and Jeffrey Heer. 2012. Termite: Visualization techniques for assessing textual topic models. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*.

Kristin A. Cook and James J. Thomas. 2005. Illuminating the path: The research and development agenda for visual analytics. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (US).

Ralph D'Agostino and Egon S Pearson. 1973. Tests for departure from normality. empirical results for the distributions of b2 and b1. *Biometrika* 60(3):613–622.

Matthew J Gardner, Joshua Lutes, Jeff Lund, Josh Hansen, Dan Walker, Eric Ringger, and Kevin Seppi. 2010. The topic browser: An interactive tool for browsing topic models. In *NIPS Workshop on Challenges of Data Visualization*.

Harold Stanley Heaps. 1978. *Information retrieval: Computational and theoretical aspects*, Academic Press, Inc., pages 206–208.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*.

Matthew D Hoffman, David M Blei, Chong Wang, and John William Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research* 14(1):1303–1347.

Yuening Hu and Jordan Boyd-Graber. 2012. Efficient tree-based topic modeling. In *Proceedings of the Association for Computational Linguistics*.

Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2014. Interactive topic modeling. *Machine Learning* 95(3):423–469.

Moontae Lee and David Mimno. 2014. Low-dimensional embeddings for interpretable anchor-based topic inference. In *Proceedings of Empirical Methods in Natural Language Processing*.

Marina Meilă. 2003. Comparing clusterings by the variation of information. In *Learning theory and kernel machines*.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Proceedings of the Association for Computational Linguistics*.

Thang Nguyen, Jordan Boyd-Graber, Jeffrey Lund, Kevin Seppi, and Eric Ringger. 2015. Is your anchor going up or down? Fast and accurate supervised topic models. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Thang Nguyen, Yuening Hu, and Jordan L Boyd-Graber. 2014. Anchors regularized: Adding robustness and extensibility to scalable topic-modeling algorithms. In *Proceedings of the Association for Computational Linguistics*.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of Uncertainty in Artificial Intelligence*.

Timothy Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. 2012. Statistical topic models for multi-label document classification. *Machine Learning* 1(88):157–208.

Ivan Titov and Ryan T McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the Association for Computational Linguistics*.

Xing Wei and W Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin* 1(6):80–83.

Ka Yee Yeung and Walter L Ruzzo. 2001. Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. *Bioinformatics* 17(9):763–774.

Apples to Apples: Learning Semantics of Common Entities Through a Novel Comprehension Task

Omid Bakhshandeh
University of Rochester
omidb@cs.rochester.edu

James F. Allen
University of Rochester
Institute for Human and Machine Cognition
james@cs.rochester.edu

Abstract

Understanding common entities and their attributes is a primary requirement for any system that comprehends natural language. In order to enable learning about common entities, we introduce a novel machine comprehension task, GuessTwo: given a short paragraph comparing different aspects of two real-world semantically-similar entities, a system should guess what those entities are. Accomplishing this task requires deep language understanding which enables inference, connecting each comparison paragraph to different levels of knowledge about world entities and their attributes. So far we have crowdsourced a dataset of more than 14K comparison paragraphs comparing entities from a variety of categories such as fruits and animals. We have designed two schemes for evaluation: open-ended, and binary-choice prediction. For benchmarking further progress in the task, we have collected a set of paragraphs as the test set on which human can accomplish the task with an accuracy of 94.2% on open-ended prediction. We have implemented various models for tackling the task, ranging from semantic-driven to neural models. The semantic-driven approach outperforms the neural models, however, the results indicate that the task is very challenging across the models.

1 Introduction

In the past few years, there has been great progress on core NLP tasks (e.g., parsing and part of speech tagging) which has renewed interest in primary language learning tasks which require text understanding and reasoning, such as machine comprehension (Schoenick et al., 2016; Hermann et al., 2015; Rajpurkar et al., 2016; Mostafazadeh et al., 2016). Our question is *how far have we got in learning basic concepts of the world through language comprehension*. If we look at the large body of work on extracting knowledge from unstructured corpora, we will see that they often lack some very basic pieces of information. For example, let us focus on the basic concept of *apple*, the fruit. What do the state-of-the-art systems and resources know about an *apple*? None of the state-of-the-art knowledge bases (Speer and Havasi, 2012; Carlson et al., 2010; Fader et al., 2011) include much precise information about the fact that apples have an edible skin, vary from sweet to sour, are round, and relatively the same size of a fist. Moreover, there is no clear approach on how to extract such information, if any, from trained word embeddings. This paper focuses on how we can automatically learn about various attributes of such generic entities in the world.

A key observation motivating this work is that we can learn more detail about objects when they are compared to other similar objects. When we compare things we often contrast, that is, we count their similarities along with their dissimilarities. This results in covering the primary attributes and aspects of objects. As humans, we tend to recall and mention the difference between things (say *green skin* vs. *red skin* in apples) as opposed to absolute measures (say the existence of skin). Interestingly, there is evidence that human knowledge is structured by *semantic similarity* and the relations among objects are defined by their relative perceptual and conceptual properties, such as their form, function, behavior, and environment (Collins and Loftus, 1975; Tversky and Gati, 1978; Cree and Mcrae, 2003). Our idea is to leverage comparison as a way of naturally learning

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 906–916
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1084

about common world concepts and their specific attributes.

Comparison, where we name the similarities and differences between things, is a unique cognitive ability in humans[1] which requires memorizing facts, experiencing things and integration of concepts of the world (Hazlitt, 1933). It is clear that developing AI systems that are capable of comprehending comparison is crucial. In this paper, in order to enable learning through comparison, we introduce a new language comprehension task which requires understanding different attributes of basic entities that are being compared.

The contributions of this paper are as follows: (1) To equip learning about common entities through comparison comprehension, we have crowdsourced a dataset of more than 14K comparison paragraphs comparing entities from nine broad categories (Section 2). This resource will be expanded over time and will be released to the public. (2) We introduce a novel task called GuessTwo, in which given a short paragraph comparing two entities, a system should guess what the two things are. (Section 3). To make systematic benchmarking on the task possible, we vet a collection of comparison paragraphs to obtain a test set on which human performs with an accuracy 94.2%. (3) We present a host of neural approaches and a novel semantic-driven model for tackling the GuessTwo task (Sections 4, 5). Our experiments show that the semantic approach outperforms the neural models. The results strongly suggest that closing the gap between system and human performances requires richer semantic processing (Section 6). We hope that this work will establish a new base for a machine comprehension test that requires systems to go beyond information extraction and towards levels of performing basic reasoning.

2 Data Collection

To enable learning about common entities, we aimed to create a dataset which meets the following goals:

1. The dataset should be a collection of high-quality documents which are rich in comparing and contrasting entities using their various attributes and aspects.

2. The comparisons in the dataset should involve everyday non-technical concepts, making their comprehension easy and commonsense for a human.

After many experiments with scraping existing Web resources, we decided to crowdsource the comparison paragraphs using Amazon Mechanical Turk[2] (Mturk). We prompt the crowd workers as follows: "Your task is to compare two given items in one simple language paragraph so that a knowledgeable person who reads it can guess what the two things are". The workers were instructed to compare only the major and well-known aspects of the two entities. We also asked them to use $\mathbb{X}$ and $\mathbb{Y}$ for anonymously referring to the two entities. Table 1 shows three examples of our crowdsourced comparison paragraphs. As these examples show, the paragraphs are very contentful and rich in comparison which meets our initial goals in the dataset creation.

Entity Pair Selection. The choice of the two entities which should be compared against each other plays a key role in the quality of the collected dataset. It is evident that naturally, we compare two things which are semantically similar, yet have some dissimilarities[3], such as *jam* and *jelly*. Given the goals of our task, we experimented with concrete nouns which share a common taxonomy class. We choose semantic classes which have at least five well-known entities. So far, we have covered nine broad categories as shown in Figure 2, with 21 subcategories shown in Figure 3. We use Wikipedia item categories and the WordNet (Miller, 1995) ontology for identifying entities from each subcategory. Then, we choose the most common entities by looking up their frequency on Google Web 1T N-grams[4]. We manually inspected the frequency-filtered list to make sure that the entities are rather easy to describe without getting technical. Given the list of entities, we paired each entity with at most five and at least three other entities from the same subcategory. We also include inter-subcategory compar-

[1] It has been suggested (Hazlitt, 1933) that children under seven years old cannot name differences between simple things such as *peach* and *apple*. This further shows that the ability for comparison develops at a later age and is cognitively complex.

[2] www.mturk.com

[3] Tversky's (1978) analysis of similarity suggests that similarity statements compare objects that belong to the same class of things.

[4] https://catalog.ldc.upenn.edu/ldc2006t13

Comparison Paragraph	Entity 𝕏	Entity 𝕐
Both 𝕏 and 𝕐 are fruits and a variety of apples. 𝕏 and 𝕐 are generally similar in size. 𝕏 are dark red in color when ripe, while 𝕐 are a bright green color. 𝕏 is sweeter and softer than 𝕐 in taste and texture, sometimes starchy. 𝕐 are tart and somewhat stringy. 𝕐 is often used in cooking, whereas 𝕏 is not.	*Red Delicious* Apple Fruit	*Granny Smith* Apple Fruit
The 𝕏 and 𝕐 are two types of vehicles. 𝕏 is a smaller vehicle than 𝕐. The 𝕏 has two wheels while 𝕐 has none. The 𝕏 travels on roadways and smooth surfaces, whereas 𝕐 is capable of flying. Only one or two people are able to ride on 𝕏 at once, while 𝕐 can carry more people.	*Motorcycle* Motor Vehicle Vehicle	*Helicopter* Aircraft Vehicle
𝕏 and 𝕐 are both types of world cuisines. 𝕏 incorporates a lot of pasta dishes and sauces, with basil, tomato, and cheese being major ingredients. 𝕐 consists of many curries and stir fried dishes, with coconut and lemongrass being used often. 𝕐 is generally spicier and more aromatic than 𝕏. 𝕏 is a European cuisine, while 𝕐 is an Asian cuisine.	*Italian Cuisine* Cuisine Cuisine	*Thai Cuisine* Cuisine Cuisine

Table 1: Examples from the GuessTwo comprehension dataset. Also provided with the dataset is the subcategory and the broad category of the entities which are listed below the entity names in this Table.

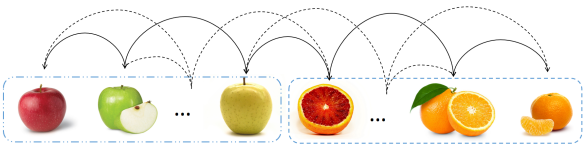


Figure 1: An example illustrating the entity pair matching process.

ison for a handful of entities at the boundaries. Figure 1 illustrates our entity pair matching process with an example on subcategories 'apple' and 'citrus'.

Data Quality Control. Our task of free-form writing is trickier than many other tasks such as tagging on Mturk. To instruct the non-expert workers, we designed a qualification test on Mturk in which the workers had to judge whether or not a given paragraph is acceptable according to our criteria. We used three carefully selected paragraphs to be a part of the qualification test. Moreover, to further ensure the quality of the submissions, one of our team members qualitatively browsed through the submissions and gave the workers detailed feedback before approving their paragraphs.

For each pair of entities, we collected eight comparison paragraphs from different workers. Given that different workers have different perspectives on what the major aspects to be compared are, collecting multiple paragraphs helps further enriching our dataset. We constrained the paragraphs to be at least 250 characters and at most 850 characters. Table 2 shows the basic statistics of our dataset. In this Table, we also included the median number of adjectives (including comparatives) per paragraph as a measure of descriptiveness of the comparison paragraphs. As a point of reference, the median number of adjectives in a random Wikipedia paragraph of the same length is 5.

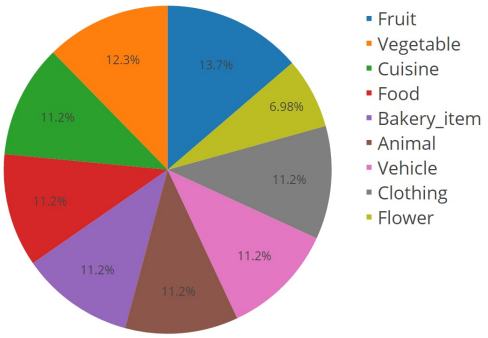


Figure 2: Distribution of broad category of the entities.

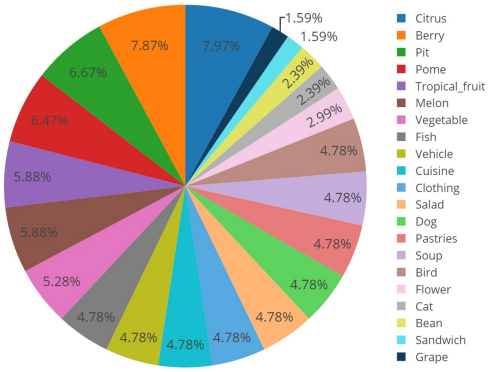


Figure 3: Distribution of subcategory of the entities.

Given the quality control we have in place, our data collection is going slowly. So far we have collected 14,142 paragraphs; however, we are aiming

Number of total approved paragraphs	**14,142**
Number of workers participated	649
Average number of paragraphs by one worker	21.7
Average work time among workers (minutes)	17.3
Median work time among workers (minutes)	6.4
Payment per paragraph (cents)	50
Number of broad entity categories	9
Number of entity sub-categories	24
Number of unique entities	920
Number of unique pairs compared	1974
Median number of sentences per paragraph	7
Median number of tokens per paragraph	70
Median number of adjectives per paragraph	7

Table 2: Statistics of the GuessTwo dataset as of April 2017.

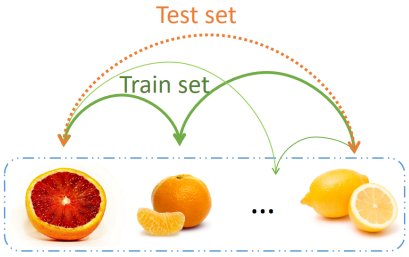


Figure 4: An example showing the entity pairs in the test and training sets.

to expand the resource over time.

Test Set Creation. In order to enable benchmarking on the task, we assessed the quality of a random sample of GuessTwo paragraphs as follows: we show the paragraph to three human workers on Mturk and ask them to guess what the two things are. Then, we choose 520 paragraphs for which all three workers have made exactly correct guesses for both entities. The test set will also be expanded along with the further data collection.

We divided the rest of the GuessTwo dataset into training and validation sets, with a 90%/10% split. To ensure that the test set requires some level of basic reasoning, our training set does not share any exact entity pairs with the validation or test set. This further enforces systems to learn about entities indirectly by processing across paragraphs. For instance, as shown in Figure 4, at test time, a system should be able to guess a comparison involving the entities *blood orange vs. lemon* by having seen comparisons of *blood orange vs. tangerine* and *tangerine vs. lemon*.

Our dataset will be released to the public through `https://omidb.github.io/guesstwo/`.

3 The GuessTwo Task Definition

We define the following two different schemes for the GuessTwo task:

• **Open-ended GuessTwo.** Given a short paragraph P which compares two entities $\mathbb{X}$ and $\mathbb{Y}$, guess what the two entities are. The scope of this prediction is the set of all entities appearing in the training dataset.

• **Binary Choice GuessTwo.** Given a short paragraph P which compares two entities $\mathbb{X}$ and $\mathbb{Y}$, and two nominals n_1 and n_2, choose 0 if $n_1 = \mathbb{X}$ and $n_2 = \mathbb{Y}$, choose 1 otherwise.

We speculate that system which can successfully tackle the GuessTwo task, has achieved two major objectives: (1) Has successfully learned the knowledge about entities stored in any form (e.g., continuous-space representation or symbolic) (2) Has a basic natural language understanding capability, using which, it can comprehend a paragraph and access its knowledge. We predict that our training dataset has enough detailed information about entities for learning the required knowledge for tackling the task. Given the design of our dataset, at test time, a system should perform some level of reasoning to go beyond understanding only one paragraph.

4 Neural Models

In this Section we present various end-to-end neural models for tackling the task of GuessTwo.

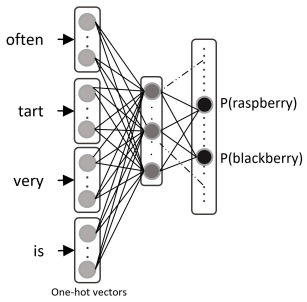
Continuous Bag-of-words Language Model. This model computes the probability of a sequence of consecutive words in context. The premise is that the probability of a paragraph with the correct realization of $\mathbb{X}$ and $\mathbb{Y}$ should be higher than the a paragraph with incorrect realizations. In order to compute the probability of a word given a context we use Continuous Bag-of-words (CBOW) (Mikolov et al., 2013a) which models the following conditional probability:

$$p(w|C(w), \theta) \quad (1)$$

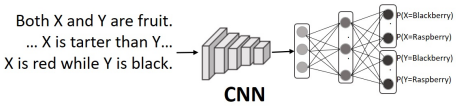
here, $C(w)$ is the context of the word w and θ is the model parameters. Then, the probability of a sequence of words (in a paragraph) is computed as follows:

$$\prod_{i=1}^{n} p(w_i|C(w_i), \theta) \quad (2)$$

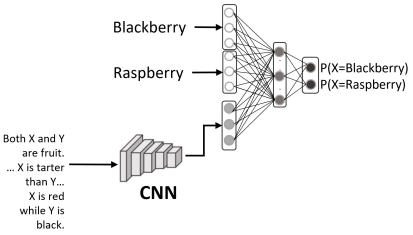
We define context to be a window of five words. Figure 5a summarizes this model. We train this



(a) The CBOW model.



(b) The CNN open-ended model.



(c) The CNN binary-choice model.

Figure 5: Various neural models for tackling the task of GuessTwo.

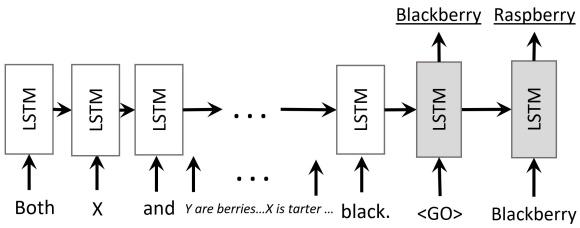


Figure 6: The Encoder-Decoder model.

model on two datasets: (1) A collection[5] of processed Wikipedia articles. Wikipedia articles often include definitions and descriptions of variety of items, which can provide a reasonable resource for our task. (2) the GuessTwo training dataset. We call these models *CBOW-Wikipedia* and *CBOW-GuessTwo* respectively.

At test time, for open-ended prediction we find the two nominals which maximize the following probability:

$$\underset{x,y}{\text{argmax}} \prod_{i=1}^{n} p(w_i | C(w_i)_{x,y}, \theta) \qquad (3)$$

where $C(w_i)_{x,y}$ indicates the context in which any occurrences of $\mathbb{X}$ have been replaced with x and $\mathbb{Y}$'s have been replaced with y. For binary choice classification, we use the same modeling except that we only consider $x = n_1, y = n_2$ and $x = n_2, y = n_1$.

Encoder-Decoder Recurrent Neural Net (RNN). This model is a sequence-to-sequence generation model (Cho et al., 2014; Sutskever et al., 2014) that maps an input sequence to an output sequence using an encoder-decoder RNN with attention (Bahdanau et al., 2014). The encoder RNN processes the comparison paragraph and the decoder generates the first item followed by the second item (Figure 6). The paragraph is encoded into a state vector of size 512. This vector is then set as the initial recurrent state of the decoder. We tune the model parameters on the validation set, where we set the number of layers to 2. The model is trained end-to-end, using Stochastic Gradient Descent with early stopping.

For open-ended prediction, we use beam search with beam-width = 25 and then output the two tokens with the highest probability. For binary choice classification, we use the same model where we set the encoder RNN inputs to the input paragraph tokens, then, we set the input of the decoder RNN once to $[n_1, n_2]$ and next to $[n_2, n_1]$. After running the network forward, we take the probability of the decoder logits and choose the ordering which has the highest probability.

Convolutional Neural Network (CNN) Encoder. As shown in the Figure 5b, this model first uses a Convolutional Neural Network (CNN) (LeCun and Bengio, 1998) for encoding the paragraph (Kim, 2014). We train a simple CNN with one layer of convolution on top of pre-trained word vectors. Here we use the word vectors trained by

[5] http://mattmahoney.net/dc/text8.zip

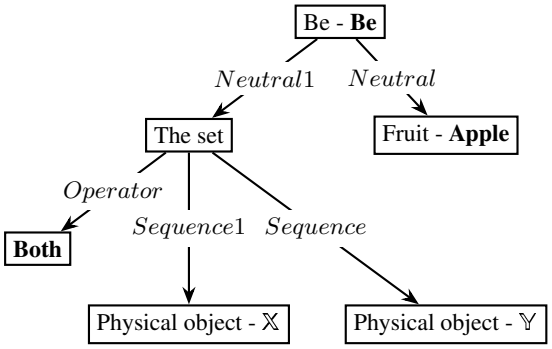


Figure 7: Semantic parsing for the sentence *Both* 𝕏 *and* 𝕐 *are apples.*

Skip-gram model (Mikolov et al., 2013b) on 100 billion words of Google News[6]. For open-ended prediction, the output of CNN is fed forward and transformed into a 300 dimension vector. Then, we use a softmax layer to get the probability of each of the possible nominals for 𝕏 and 𝕐. For binary choice classification, we use the same architecture and settings as above. Additionally, we encode each nominal into a 300-dimensional vector, which then gets concatenated with the paragraph vector. Figure 5c shows this model.

5 Semantic-driven Model

In this Section we present a semantic-driven approach which models the comparison paragraph using semantic features and is capable of performing basic reasoning across paragraphs.

5.1 Representing Paragraphs

The question is, given a comparison paragraph, what is the best representation which can enable further reasoning? The comparison paragraphs often have complex syntactic and semantic structures, which might be challenging for many off-the-shelf NLP tools to process. For instance, consider the sentence 𝕏 *is much sweeter in taste than* 𝕐. Although a dependency parser provides a lot of information regarding how the individual words relate grammatically, it does not give us any information regarding how 𝕐's sweetness (which is elided from the sentence and is implicit) relates to 𝕏's. As another processing technique, if we use the standard information extraction methods for extracting and representing syntactic triplets (*argument1, relation, argument2*) (Fader et al., 2014; Etzioni et al., 2011), we will extract a triplet such as 𝕏 *is sweeter* which shares the same shortcomings.

Our approach for better representation of comparison paragraphs starts with a broad-coverage semantic parser (Banarescu et al., 2013; Bos, 2008; Allen et al., 2008). A semantic parser maps an input sentence to its formal meaning representation, operating at the generic natural language level. Here we use the TRIPS[7] (Allen et al., 2008) broad-coverage semantic parser. TRIPS provides a very rich semantic structure; mainly it provides sense disambiguated deep structures augmented with semantic ontology types. Figure 7 shows an example TRIPS semantic parse. In this graph representation, each node specifies a word in bold along with its corresponding ontology type on its left. The edges in the graph are semantic roles[8]. As you can see, this semantic parse represents the sentence by decoupling the token 'both' and attributing the property of 'be apple' to both 𝕏 and 𝕐.

In our comparison paragraphs there are two major types of sentences:

• **Sentences with Absolute Information.** These sentences contain direct information about the entities, such as 𝕏 *is red* or *Both* 𝕏 *and* 𝕐 *are very sweet*. From each absolute sentence, we extract frames which describe the absolute attributes of the corresponding entity. We define a frame to be a subgraph of a semantic parse which involves exactly one entity and all of its semantic roles. Relying on the deep semantic features offered by the semantic parser, we perform negation propagation[9] and sequence decoupling, among others features. For example, given a sentence which has a sequence, as the one depicted in Figure 7, we perform sequence decoupling and extract the two frames *[*𝕏 *Be Apple]* and *[*𝕐 *Be Apple]*.

• **Sentences with Relative Information.** These sentences contain relative information about the two entities, for instance, 𝕏 *is somewhat sweeter than* 𝕐. As opposed to the sentences with absolute information, we cannot extract frames from sentences with comparisons directly. Various properties of entities can be associated with an abstract scale, such as 'size' or 'sweetness', on which dif-

[6] https://code.google.com/archive/p/word2vec/

[7] http://trips.ihmc.us/parser/cgi/parse

[8] Refer to http://trips.ihmc.us/parser/LFDocumentation.pdf for the full list of semantic roles in TRIPS parser.

[9] A common construction which needs negation propagation is *Neither* 𝕏 *nor* 𝕐 *are ...* .

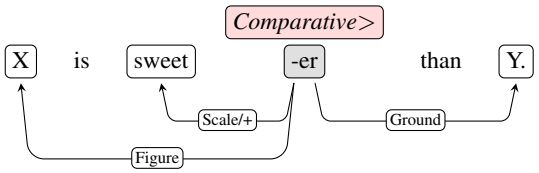


Figure 8: The comparison construction predicted for the sentence *X is sweeter than Y.*

ferent entities can be compared. In order to extract such scales and the relative standing of items on them we use the structured prediction model presented in Bakhshandeh et al. (2016), which given a sentence predicts its comparison structures. Figure 8 shows an example predicate-argument structure that is predicted by this model. We use pretrained model on the annotated corpus (Bakhshandeh et al., 2016) of comparison structures.

Given a comparison structure such as the one presented in Figure 8, we can extract the information that on the scale of 'sweetness' $\mathbb{X}$ is higher than $\mathbb{Y}$. It is clear that one can build a large knowledge base of such relations by reading large collections of comparison paragraphs. We populate our knowledge base of relative information about entities as follows: First, we predict the comparison structure of each sentence and then extract a binary relation $\prec_s$ which shows the relation on the scale of s. Second, for any scale s, we apply transitivity on its entities. As shown in equation 4, the binary relation $\prec_s$ is transitive over the set of all entities, A. This process, called closure, enables us do basic reasoning and derives implicit relations on scales from explicit relations.

$$\forall s \in S\ \forall x, y, z \in A : (x \prec_s y\ \wedge\ y \prec_s z) \\ \implies x \prec_s z \tag{4}$$

The product of this step is a structured knowledge base on entity ordering which we call the *ordering lattice*. Figure 9 shows an example partial ordering lattice inferred by our model, where the sweetness of *Golden Delicious* can be compared to *Granny Smith* through their direct link with *Red Delicious*.

5.2 Modeling

Given a paragraph P, we first extract the set of all the absolute information frames for $\mathbb{X}$ and $\mathbb{Y}$ (as described above), called $F_{\mathbb{X}}(P)$ and $F_{\mathbb{Y}}(P)$. Second, for the sentences with relative information,

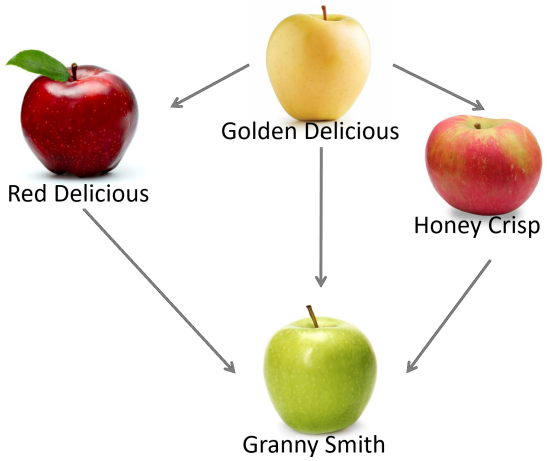


Figure 9: The inferred partial ordering lattice comparing the *sweetness* of different apples.

we extract all the binary relations $\prec_s \in R(P)$ that should hold between $\mathbb{X}$ and $\mathbb{Y}$. Then, our objective is to find two realizations for $\mathbb{X}$ and $\mathbb{Y}$ that maximize the following:

$$\underset{x,y}{\text{argmax}}\ p(x|F_{\mathbb{X}}(P)) + p(y|F_{\mathbb{Y}}(P)) \\ \text{s.t. } \forall \prec_s \in R(P) : x \prec_s y \tag{5}$$

In order to compute the $p(x|F_{\mathbb{X}}(P))$ and $p(y|F_{\mathbb{Y}}(P))$ scores we used Regularized Gradient Boosting (XGBoost) classifier (Friedman, 2000), which uses a regularized model formulation to limit overfitting. We directly use each frame in the $F_{\mathbb{X}}(P)$ and $F_{\mathbb{Y}}(P)$ sets as the classifier features. We use Integer Linear Programming (ILP) for formulating the constraints as follows: for each relation $r \in R$ on the scale s, we lookup the scale s in the ordering lattice and make the blacklist $B(P)$ containing each pair of entities which do not satisfy the relation r. Our ordering lattice does not have perfect complete information, hence, we have Open World Assumption and only prune our search space not to include the already observed pairs which violate the relation. our ILP objective function will be the following:

$$\underset{b,b'}{\text{argmax}} \sum_{x \in N} b_x\ p(x|F_{\mathbb{X}}(P)) + \\ \sum_{y \in N} b'_y\ p(y|F_{\mathbb{Y}}(P)) \\ \text{s.t. } \forall\ (j, j') \in B(P) : b_j + b'_{j'} \leq 1 \tag{6}$$

where N is the set of all possible realizations and b and b' are the binary indicator variables, so $b_x = 1$ indicates the realization of x for $\mathbb{X}$.

In the case of open-ended prediction, the maximization presented in Equation 6 is carried out on the set N. In the case of binary choice classification, however, only the two choices of n_1 and n_2 are considered in the maximization.

6 Results

We evaluate all the models presented in Sections 4 and 5 using the following accuracy measure:

$$\frac{\#\text{correct predictions of both entities}}{\#\text{test cases}} \quad (7)$$

As for the open-ended prediction we compute the nominator of the accuracy measure using three various matching methods on both entities: (1) exact-match, (2) subcategory match, (3) broad category match.

As Table 3 shows, the semantic model outperforms all the neural models. Moreover, the ILP constraints have been very effective in directing the system in the correct search space. Among the neural models, the Encoder-Decoder RNN model performs noticeably better than other models when matching the subcategory and broad category. According to the exact-matching, neither of the CBOW models could guess any of the two test entities correctly. Overall, it is evident that the end-to-end neural models have not been able to generalize well and learn about the attributes of entities across various training paragraphs. This can be partly due to not being trained on large enough comparison training dataset. The semantic model, however, could outperform the neural models using the same amount of data. To a degree, this is because the semantic model leverages the basic language understanding capabilities offered by the semantic parser.

It is also important to note that our semantic approach is not only capable of binary and open-ended prediction, but it also offers two by-products that can be used as knowledge in a variety of other tasks: (1) a set of the most important absolute information frames which can be chosen based on feature importance in the classification, (2) the partial ordering lattice of entities. Overall, the results strongly suggest that the GuessTwo task is challenging, with the open-ended scheme being the most challenging. There is a wide gap between human and system performance on this task, which makes it a very promising task for the community to pursue.

Model	**Binary**	**Open-ended**	
		Exact.	**Subcat.**
Human	100.0	94.2	100.0
CBOW-Wikipedia	51.9	0.0	1.5
CBOW-GuessTwo	51.7	0.0	1.1
Encoder-Decoder RNN	58.8	2.9	6.8
CNN	57.6	1.9	2.5
Semantic (no constraints)	61.5	10.5	38.5
Semantic (with ILP constraints)	**69.2**	**11.7**	**40.4**

Table 3: System accuracy results on the GuessTwo test set. A random baseline on binary choice task achieves 51%. The open-ended evaluation has two columns: exact-match (exact) and subcategory match (subcat), respectively.

7 Related Work

The task of Machine Comprehension (MC) has gained a significant attention over the past few years. The major driver for MC has been the publicly available benchmarking datasets. A variety of MC tasks have been introduced in the community (Richardson et al.; Hermann et al., 2015; Rajpurkar et al., 2016; Hill et al., 2015), in which the system reads a short text and answers a few multiple-choice questions. The reading comprehension involved in these tests ranges from reading a short fictional story (Richardson et al.) to reading a short news article (Hermann et al., 2015). In comparison, in the GuessTwo task the reading comprehension involves reading a short comparison paragraph and one can say the multiple-choice question is the constant *What are* $\mathbb{X}$ *and* $\mathbb{Y}$*?*

The CNN/DailyMail dataset consists of more than 100K short news articles with the questions automatically created from the bullet-point summaries of the original article. This dataset uses fill-in-the-blank-style questions such as 'Producer *X* will not press charges against Jeremy Clarkson' where the system should choose among all the anonymized entities in the corresponding paragraph to fill in *X*. The Stanford Question Answering (SQuAD) dataset is another recent machine comprehension test with over 500 Wikipedia articles and +100,000 crowdsourced questions. The answer to every question in this dataset is a span of text from the corresponding reading passage.

Human accuracy on CNN/DailyMail is estimated to be around 75% (Chen et al., 2016) with the current state-of-the-art at 76.1 on CNN (Sordoni et al., 2016), and 75.8 on DailyMail (Chen et al., 2016). The human F1 score on SQuAD

dataset is reported to be at 86.8%, with the current state-of-the-art achieving 82.9%. Given these statistics, neither of these datasets leave enough room for further research. Given that in both these tasks the answer to the question is directly found in the provided passage, we argue that the community requires a more challenging MC task which goes beyond matching and needs some level of inference across passages. The GuessTwo task requires basic reasoning and inference across paragraphs for comprehending various aspects of entities relative to one another.

Another interesting task is MCTest (Richardson et al.), which is a reading comprehension test with 660 fictional stories as the passage and four questions per story. The human-level performance on MCTest is estimated to be around 90%, with the state-of-the-art achieving an accuracy of 70% (Wang et al., 2015). MCTest is also proven to be challenging, however, given its very limited training data, further progress on the task has been hindered. Yet another relevant QA task is the Allen AI Science Challenge (Clarke et al., 2010; Schoenick et al., 2016), which is a dataset of multiple-choice questions and answers from a standardized 8th grade science exam. The questions can range from simple fact lookup to complex ones which require extensive world knowledge and commonsense reasoning. This task requires machine reading of a variety of resources such as textbooks and goes beyond reading a couple of passages.

8 Conclusion

We introduced the novel task of GuessTwo, in which given a short paragraph comparing two common entities, a system should guess what the two entities are. The comparison paragraphs often have complex semantic structures which make this comprehension task demanding. Furthermore, guessing the two entities requires a system to go beyond only understanding one given passage and requires reasoning across paragraphs, which is one of the most under-explored, yet crucial, capabilities of an intelligent agent.

So far, we have crowdsourced a dataset of more than 14K comparison paragraphs comparing entities from nine major categories. For benchmarking the progress, we filter a collection of these paragraphs to create a test set, on which humans perform with an accuracy of 94.2%. For continuing our data collection, we would like to have a targeted entity pair selection where we particularly collect the missing relations in our partial ordering lattice. We believe that this process can help developing more effective systems. For the most recent statistics of the dataset and the best performing systems please check this website.

We presented a host of neural models and a novel semantic-driven approach for tackling the task of GuessTwo. Our experiments show that the semantic approach outperforms the neural models by a large margin. The poor performance of the neural models we experimented with can motivate designing new architectures which are capable of performing basic reasoning across paragraphs. The results strongly suggest that bridging the gap between system and human performance on this task requires models with richer language representation and reasoning capabilities. As a future work, we would like to explore the feasibility of marrying our semantic and neural models to exploit the benefits that each of them has to offer.

9 Acknowledgments

This work was supported in part by Grant W911NF-15-1-0542 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO). We would like to thank Linxiuzhi Yang for her help in the data collection and anonymous reviewers for their insightful comments on this work. We specially thank William de Beaumont for his invaluable feedback on this paper. We also thank the inputs from Steven Piantadosi, Brad Mahon, and Gregory Carlson on cognitive aspects of comparison.

References

James F. Allen, Mary Swift, and Will de Beaumont. 2008. Deep semantic analysis of text. In *Proceedings of the 2008 Conference on Semantics in Text Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, STEP '08, pages 343–354. http://dl.acm.org/citation.cfm?id=1626481.1626508.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Omid Bakhshandeh, Alexis Cornelia Wellwood, and James Allen. 2016. Learning to jointly predict ellipsis and comparison structures. In *Proceedings of The 20th SIGNLL Conference on Computational*

Natural Language Learning. Association for Computational Linguistics, Berlin, Germany, pages 62–74. http://www.aclweb.org/anthology/K16-1007.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, pages 178–186. http://www.aclweb.org/anthology/W13-2322.

Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*. College Publications, Research in Computational Semantics, pages 277–286.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *In AAAI*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/-daily mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, CoNLL '10, pages 18–27. http://dl.acm.org/citation.cfm?id=1870568.1870571.

Allan M. Collins and Elizabeth F. Loftus. 1975. A spreading-activation theory of semantic processing. *Psychological Review* 82(6):407 – 428.

George S. Cree and Ken Mcrae. 2003. Analyzing the factors underlying the structure and computation of the meaning of chipmunk, cherry, chisel, cheese, and cello (and many other such concrete nouns). *Journal of Experimental Psychology: General* 132(2):163–201+.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*. AAAI Press, IJCAI'11, pages 3–10. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-012.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 1535–1545. http://dl.acm.org/citation.cfm?id=2145432.2145596.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '14, pages 1156–1165. https://doi.org/10.1145/2623330.2623677.

Jerome H. Friedman. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29:1189–1232.

V. Hazlitt. 1933. *The psychology of infancy*. E.P. Dutton and company, inc. https://books.google.com/books?id=I8svAAAAYAAJ.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *International Conference on Learning Representations (ICLR)* .

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, pages 1746–1751. http://aclweb.org/anthology/D/D14/D14-1181.pdf.

Yann LeCun and Yoshua Bengio. 1998. The handbook of brain theory and neural networks. MIT Press, Cambridge, MA, USA, chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. http://dl.acm.org/citation.cfm?id=303568.303704.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information*

Processing Systems 26, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11):39–41. https://doi.org/10.1145/219717.219748.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 839–849. http://www.aclweb.org/anthology/N16-1098.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text .

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. ???? Mctest: A challenge dataset for the open-domain machine comprehension of text. pages 193–203.

Carissa Schoenick, Peter Clark, Oyvind Tafjord, Peter D. Turney, and Oren Etzioni. 2016. Moving beyond the turing test with the allen AI science challenge. *CoRR* abs/1604.04315. http://arxiv.org/abs/1604.04315.

Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *CoRR* abs/1606.02245. http://arxiv.org/abs/1606.02245.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uur Doan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Amos Tversky and Itamar Gati. 1978. Studies of similarity. *Cognition and categorization* 1(1978):79–98.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*. The Association for Computer Linguistics, pages 700–706. http://aclweb.org/anthology/P/P15/P15-2115.pdf.

Going out on a limb: Joint Extraction of Entity Mentions and Relations without Dependency Trees

Arzoo Katiyar and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY, 14853, USA
arzoo, cardie@cs.cornell.edu

Abstract

We present a novel attention-based recurrent neural network for joint extraction of entity mentions and relations. We show that attention along with long short term memory (LSTM) network can extract semantic relations between entity mentions without having access to dependency trees. Experiments on Automatic Content Extraction (ACE) corpora show that our model significantly outperforms feature-based joint model by Li and Ji (2014). We also compare our model with an end-to-end tree-based LSTM model (SPTree) by Miwa and Bansal (2016) and show that our model performs within 1% on entity mentions and 2% on relations. Our fine-grained analysis also shows that our model performs significantly better on AGENT-ARTIFACT relations, while SPTree performs better on PHYSICAL and PART-WHOLE relations.

1 Introduction

Extraction of entities and their relations from text belongs to a very well-studied family of structured prediction tasks in NLP. There are several NLP tasks such as fine-grained opinion mining (Choi et al., 2006), semantic role labeling (Gildea and Jurafsky, 2002), etc., which have a similar structure; thus making it an important and a challenging task.

Several methods have been proposed for entity mention and relation extraction at the sentence-level. These can be broadly categorized into – 1) pipeline models that treat the identification of entity mentions (Nadeau and Sekine, 2007) and relation classification (Zhou et al., 2005) as two separate tasks; and 2) joint models, also the more recent, which simultaneously identify the entity mention and relations (Li and Ji, 2014; Miwa and Sasaki, 2014). Joint models have been argued to perform better than the pipeline models as knowledge of the typed relation can increase the confidence of the model on entity extraction and vice versa.

Recurrent networks (RNNs) (Elman, 1990) have recently become very popular for sequence tagging tasks such as entity extraction that involves a set of contiguous tokens. However, their ability to identify relations between *non-adjacent* tokens in a sequence, e.g., the head nouns of two entities, is less explored. For these tasks, RNNs that make use of tree structures have been deemed more suitable. Miwa and Bansal (2016), for example, propose an RNN comprised of a sequence-based long short term memory (LSTM) for entity identification and a separate tree-based dependency LSTM layer for relation classification using shared parameters between the two components. As a result, their model depends critically on access to dependency trees, restricting it to sentence-level extraction and to languages for which (good) dependency parsers exist. Also, their model does not jointly extract entities and relations; they first extract all entities and then perform relation classification on all pairs of entities in a sentence.

In our previous work (Katiyar and Cardie, 2016), we address the same task in an opinion extraction context. Our LSTM-based formulation explicitly encodes distance between the head of entities into opinion relation labels. The output space of our model is quadratic in size of the entity and relation label set and we do not specifically identify the relation type. Unfortunately, adding relation type makes the output label space very sparse, making it difficult for the model to learn.

In this paper, we propose a novel RNN-based model for the joint extraction of entity mentions

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 917–928
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1085

and relations. Unlike other models, our model does not depend on any dependency tree information. Our RNN-based model is a multi-layer bi-directional LSTM over a sequence. We encode the output sequence from left-to-right. At each time step, we use an attention-like model on the previously decoded time steps, to identify the tokens in a specified relation with the current token. We also add an additional layer to our network to encode the output sequence from right-to-left and find significant improvement on the performance of relation identification using bi-directional encoding.

Our model significantly outperforms the feature-based structured perceptron model of Li and Ji (2014), showing improvements on both entity and relation extraction on the ACE05 dataset. In comparison to the dependency tree-based LSTM model of Miwa and Bansal (2016), our model performs within 1% on entities and 2% on relations on ACE05 dataset. We also find that our model performs significantly better than their tree-based model on the AGENT-ARTIFACT relation, while their tree-based model performs better on PHYSICAL and PART-WHOLE relations; the two models perform comparably on all other relation types. The very competitive performance of our non-tree-based model bodes well for relation extraction of non-adjacent entities in low-resource languages that lack good parsers.

In the sections that follow, we describe related work (Section 2); our bi-directional LSTM model with attention (Section 3); the training (Section 4); the experiments on ACE dataset (Section 5); results (Section 6); error analysis (Section 7) and conclusion (Section 8).

2 Related Work

RNNs (Hochreiter and Schmidhuber, 1997) have been recently applied to many sequential modeling and prediction tasks, such as machine translation (Bahdanau et al., 2015; Sutskever et al., 2014), named entity recognition (NER) (Hammerton, 2003), opinion mining (Irsoy and Cardie, 2014). Variants such as adding CRF-like objective on top of LSTMs have been found to produce state-of-the-art results on several sequence prediction NLP tasks (Collobert et al., 2011; Huang et al., 2015; Katiyar and Cardie, 2016). These models assume conditional independence at the output layer whereas the model we propose in this paper does not assume any conditional independence at the output layer, allowing it to model an arbitrary distribution over output sequences.

Relation classification has been widely studied as a stand-alone task, assuming that the arguments of the relations are known in advance. There have been several models proposed including feature-based models (Bunescu and Mooney, 2005; Zelenko et al., 2003) and neural network based models (Socher et al., 2012; dos Santos et al., 2015; Hashimoto et al., 2015; Xu et al., 2015a,b).

For joint-extraction of entities and relations, feature-based structured prediction models (Li and Ji, 2014; Miwa and Sasaki, 2014), joint inference integer linear programming models(Yih and Roth, 2007; Yang and Cardie, 2013), card-pyramid parsing (Kate and Mooney, 2010) and probabilistic graphical models (Yu and Lam, 2010; Singh et al., 2013) have been proposed. In contrast, we propose a neural network model which does not depend on the availability of any features such as part of speech (POS) tags, dependency trees, etc.

Recently, Miwa and Bansal (2016) proposed an end-to-end LSTM based sequence and tree-structured model. They extract entities via a sequence layer and relations between the entities via the shortest path dependency tree network. In this paper, we try to investigate recurrent neural networks with attention for extracting semantic relations between entity mentions without using any dependency parse tree features. We also present the *first* neural network based joint model that can extract entity mentions and relations along with the relation type. In our previous work (Katiyar and Cardie, 2016), as explained earlier, we proposed a LSTM-based model for joint extraction of opinion entities and relations, but no relation types. This model cannot be directly extended to include relation types as the output space becomes sparse making it difficult for the model to learn.

Recent advances in recurrent neural network has seen the application of attention on recurrent neural networks to obtain a representation weighted by the importance of tokens in the sequence model. Such models have been very frequently used in question-answering tasks (for recent examples, see Chen et al. (2016) and Lee et al. (2016)), machine translation (Luong et al., 2015; Bahdanau et al., 2015), and many other NLP applications. Pointer networks (Vinyals et al., 2015), an adaptation of attention models, use these token-level weights as pointers to the input elements.

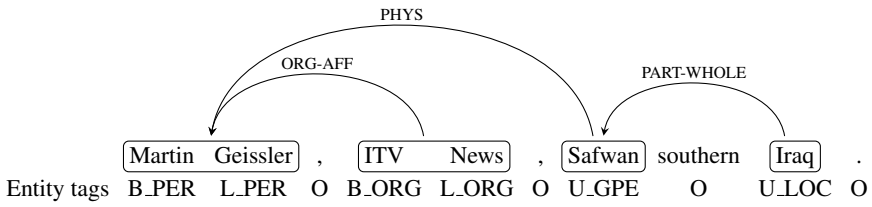


Figure 1: Gold standard annotation for an example sentence from ACE05 dataset.

Zhai et al. (2017), for example, have used these for neural chunking, and Nallapati et al. (2016) and Cheng and Lapata (2016), for summarization. However, to the best of our knowledge, these networks have not been used for joint extraction of entity mentions and relations. We present first such attempt to use these attention models with recurrent neural networks for joint extraction of entity mentions and relations.

3 Model

Our model comprises of a multi-layer bi-directional recurrent network which learns a representation for each token in the sequence. We use the hidden representation from the top layer for joint entity and relation extraction. For each token in the sequence, we output an entity tag and a relation tag. The entity tag corresponds to the entity type, whereas the relation tag is a tuple of pointers to related entities and their respective relation types. Figure 1 shows the annotation for an example sentence from the dataset. We transform the relation tags from entity level to token level. For example, we separately model the relation "ORG-AFF" for each token in the entity "ITV News". Thus, we model the relations between "ITV" and "Martin Geissler", and "News" and "Martin Geissler" separately. We employ a pointer-like network on top of the sequence layer in order to find the relation tag for each token as shown in Figure 2. At each time step, the network utilizes the information available about all output tags from the previous time steps in order to output the entity tag and relation tag *jointly* for the current token.

3.1 Multi-layer Bi-directional Recurrent Network

We use multi-layer bi-directional LSTMs for sequence tagging because LSTMs are more capable of capturing long-term dependencies between tokens, making it ideal for both entity mention and relation extraction.

Using LSTMs, we can compute the hidden state $\overrightarrow{h_t}$ in the forward direction and $\overleftarrow{h_t}$ in the backward direction for every token as below:

$$\overrightarrow{h}_t = LSTM(x_t, \overrightarrow{h}_{t-1})$$
$$\overleftarrow{h}_t = LSTM(x_t, \overleftarrow{h}_{t+1})$$

For every token t in the subsequent layer l, we combine the representations $\overrightarrow{h}_t^{l-1}$ and $\overleftarrow{h}_t^{l-1}$ from previous layer l-1 and feed it as an input. In this paper, we only use the hidden state from the last layer L for output layer and compute the top hidden layer representation as below:

$$z_t^{'} = \overrightarrow{V}\overrightarrow{h}_t^{(L)} + \overleftarrow{V}\overleftarrow{h}_t^{(L)} + c$$

$\overrightarrow{V}$ and $\overleftarrow{V}$ are weight matrices for combining hidden representations from the two directions.

3.2 Entity detection

We formulate entity detection as a sequence labeling task using BILOU scheme similar to Li and Ji (2014) and Miwa and Bansal (2016). We assign each token in the entity with the tag B appended with the entity type if it is the beginning of the entity, I for inside of an entity, L for the end of the entity or U if there is only one token in the entity. Figure 1 shows an example of the entity tag sequence assigned to the sentence. For each token in the sequence, we perform a softmax over all candidate tags to output the most likely tag:

$$y_t = \text{softmax}(Uz_t^{'} + b)$$

Our network structure as shown in Figure 2 also contains connections from the output y_{t-1} of the previous time step to the current top hidden layer. Thus our outputs are not conditionally independent from each other. In order to add connections from y_{t-1}, we transform this output k into a label embedding b_{t-1}^k [1]. We represent each label type

[1] We can also add relation label embeddings using the relation tag output from the previous time step.

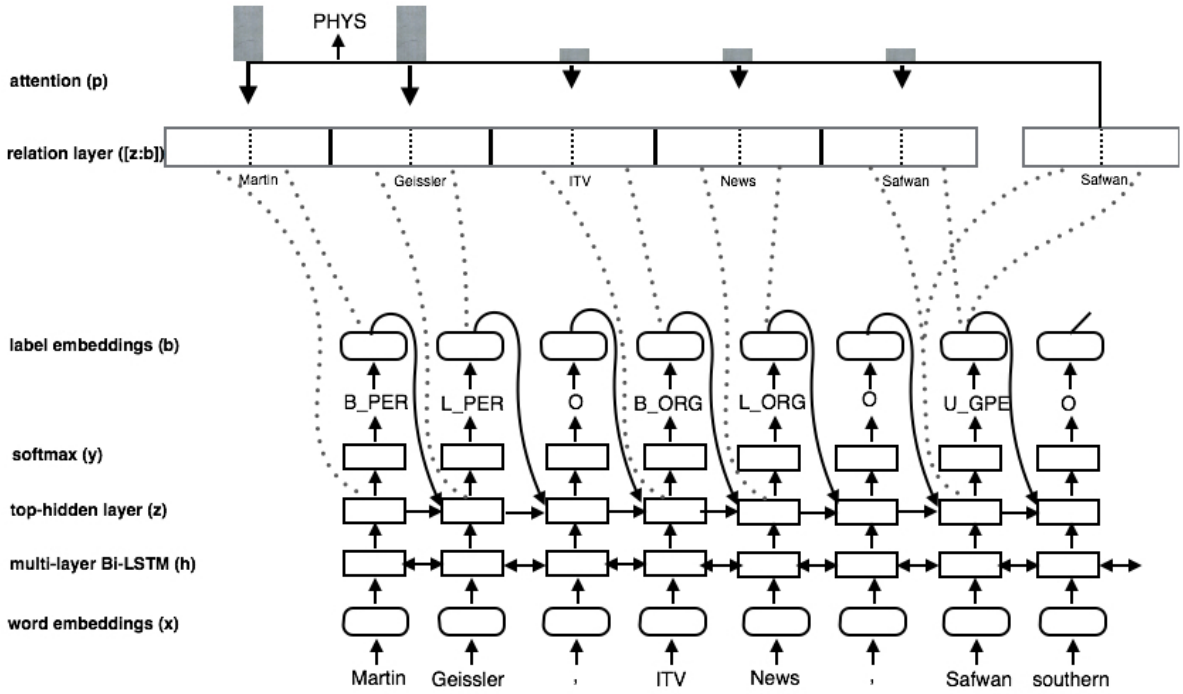


Figure 2: Our network structure based on bi-directional LSTMs for joint entity and relation extraction. This snapshot shows the network when encoding the relation tag for the word "Safwan" in the sentence. The dotted lines in the figure show that top hidden layer and label embeddings for tokens is copied into relation layer. The pointers at attention layer indicate the probability distribution over tokens, the length of the pointers is used to denote the probability value.

k with a dense representation b^k. We compute the output layer representations as:

$$z_t = LSTM([z_t^{'}; b_{t-1}^k], h_{t-1})$$
$$y_t = \text{softmax}(Uz_t + b^{'})$$

We decode the output sequence from left to right in a greedy manner.

3.3 Attention Model

We use attention model for relation extraction. Attention models, over an encoder sequence of representations z, can compute a soft probability distribution p over these learned representations, where d_i is the i^{th} token in decoder sequence. These probabilities are an indication of the importance of different tokens in the encoder sequence:

$$u_t^i = v^T \tanh(W_1 z + W_2 d_i)$$
$$p_t^i = \text{softmax}(u_t^i)$$

v is a weight matrix for attention which transforms the hidden representations into attention scores.

We use pointer networks (Vinyals et al., 2015) in our approach, which are a variation of these attention models. Pointer networks interpret these p_t^i as the probability distribution over the input encoding sequence and use u_t^i as pointers to the input elements. We can use these pointers to encode relation between the current token and the previous predicted tokens, making it fit for relation extraction as explained in Section 3.4.

3.4 Relation detection

We formulate relation extraction also as a sequence labeling task. For each token, we want to find the tokens in the past that the current token is related to along with its relation type. In Figure 1, "Safwan" is related to the tokens "Martin" as well as "Geissler" by the relation type "PHYS". For simplicity, let us assume that there is only one previous token the current token is related to when training, i.e., "Safwan" is related to "Geissler" via PHYS relation. We can extend our approach to output multiple relations as explained in Section 4.

We use pointer networks as described in Sec-

tion 3.3. At each time step, we stack the top hidden layer representations from the previous time steps $z_{\leq t}$[2] and its corresponding label embeddings $b_{\leq t}$. We only stack the top hidden layer representations for the tokens which were predicted as non-O's for previous time steps as shown in Figure 2. Our decoding representation at time t is the concatenation of z_t and b_t. The attention probabilities can now be computed as below:

$$u^t_{\leq t} = v^T \tanh(W_1[z_{\leq t}; b_{\leq t}] + W_2[z_t; b_t])$$
$$p^t_{\leq t} = \text{softmax}(u^t_{\leq t})$$

Thus, $p^t_{\leq t}$ corresponds to the probability of each token, in the sequence so far, being related to the current token at time step t. For the case of NONE relations, the token at t is related to itself.

We also want to find the type of the relations. In order to achieve this, we add an extra dimension to v corresponding to the size of relation types R space. Thus, u^i_t is no longer a score but a R dimensional vector. We then take softmax over this vector of size $O(|z_{\leq t}| \times R)$ to find the most likely tuple of pointer to the related entity and its relation type.

3.5 Bi-directional Encoding

Bi-directional LSTMs have been found to be able to capture context better than plain left-to-right LSTMs, based on their performance on various NLP tasks (Irsoy and Cardie, 2014). Also, Sutskever et al. (2014) found that their performance on machine translation task improved on reversing the input sentences during training. Inspired by these developments, we experiment with bi-directional encoding at the output layer. We add another top hidden layer on Bi-LSTM in Figure 2 which encodes the output sequence from right-to-left. The two encoding share the same multi-layer bi-directional LSTM except for the top hidden layer. Thus, we have two output layers in our network which output the entity tags and relation tags separately. At inference time, we employ heuristics to combine the output from the two directions.

[2]The notation $\leq$ is used to denote the stacking of the representations from the previous time steps. Thus, if z_t is a 2-dimensional matrix then $z_{\leq t}$ will be a 3-dimensional tensor. The size along the first dimension will now correspond to the number of 2-dimensional matrices stacked.

4 Training

We train our network by maximizing the log-probability of the correct entity E and relation R tag sequences *jointly* given the sentence S as below:

$$\begin{aligned}
&\log p(E, R|S, \theta) \\
=&\frac{1}{|S|}\sum_{i \in |S|} \log p(e_i, r_i|e_{<i}, r_{<i}, S, \theta) \\
=&\frac{1}{|S|}\sum_{i \in |S|} \log p(e_i|e_{<i}, r_{<i}) + \log p(r_i|e_{\leq i}, r_{<i})
\end{aligned}$$

Thus, we can decompose our objective into the sum of log-probabilities over entity sequence and relation sequence. We use the gold entity tags while training. As shown in Figure 2, we input the label embedding from the previous time step to the top hidden layer at the current time step along with the other recurrent inputs. During training, we pass the gold label embedding to the next time step which enables better training of our model. However, at test time when the gold label is not available we use the predicted label at previous time step as input to the current step.

At inference time, we can greedily decode the sequence to find the most likely entity $\widehat{E}$ and relation $\widehat{R}$ tag sequences:

$$(\widehat{E}, \widehat{R}) = \underset{E,R}{\text{argmax}}\, p(E, R)$$

Since, we add another top layer to encode tag sequences in the reverse order as explained in Section 3.5, there may be conflicts in the output. We select the positive and more confident label similar to Miwa and Bansal (2016).

Multiple Relations Our approach to relation extraction is different from Miwa and Bansal (2016). Miwa and Bansal (2016) present each pair of entities to their model for relation classification. In our approach, we use pointer networks to identify the related entities. Thus, for our approach described so far if we only compute the argmax on our objective then we limit our model to output only one relation label per token. However, from our analysis of the dataset, an entity may be related to more than one entity in the sentence. Hence, we modify our objective to include multiple relations. In Figure 2, token "Safwan" is related to both tokens "Martin" and "Geissler" of the entity "Martin Geissler", hence we assign probability of 0.5

to both these tokens. This can be easily expanded to include tokens from other related entities, such that we assign equal probability $\frac{1}{N}$ to all tokens[3] depending on the number N of these related tokens.

The log-probability for the entity part remain the same as in our objective discussed in Section 4, however we modify the relation log-probability as below:

$$\sum_{|j:\mathbf{r}'_{\mathbf{i,j}}>0|} \mathbf{r}'_{\mathbf{i,j}} \log p(\mathbf{r_{i,j}}|e_{\leq i}, \mathbf{r_{<i}}, S, \theta)$$

where, $\mathbf{r}'_{\mathbf{i}}$ is the true distribution over relation label space and $\mathbf{r_i}$ is the softmax output from our model. From empirical analysis, we find that $\mathbf{r}'_{\mathbf{i}}$ is generally sparse and hence using a cross entropy objective like this can be useful to find multiple relations. We can also use Sparsemax (Martins and Astudillo, 2016) instead of softmax which is more suitable for sparse distributions. However, we leave it for future work.

At inference time, we output all the labels with probability value above a certain threshold. We adapt this threshold based on the validation set.

5 Experiments

5.1 Data

We evaluate our proposed model on the two datasets from the Automatic Content Extraction (ACE) program – ACE05 and ACE04. There are 7 main entity types namely Person (PER), Organization (ORG), Geographical Entities (GPE), Location (LOC), Facility (FAC), Weapon (WEA) and Vehicle (VEH). For each entity, both entity mentions and its head phrase are annotated. For the scope of this paper, we only use the entity head phrase similar to Li and Ji (2014) and Miwa and Bansal (2016). Also, there are relation types namely Physical (PHYS), Person-Social (PER-SOC), Organization-Affiliation (ORG-AFF), Agent-Artifact (ART), GPE-Affiliation (GPE-AFF).

ACE05 has a total of 6 relation types including PART-WHOLE. We use the same data splits as Li and Ji (2014) and Miwa and Bansal (2016) such that there are 351 documents for training, 80 for development and the remaining 80 documents for the test set.

ACE04 has 7 relation types with an additional Discourse (DISC) type and split ORG-AFF relation type into ORG-AFF and OTHER-AFF. We perform 5-fold cross validation similar to Chan and Roth (2011) for fair comparison with the state-of-the-art.

5.2 Evaluation Metrics

In order to compare our system with the previous systems, we report micro F1-scores, Precision and Recall on both entities and relations similar to Li and Ji (2014) and Miwa and Bansal (2016). An entity is considered correct if we can identify its head and the entity type correctly. A relation is considered correct if we can identify the head of the argument entities and also the relation type. We also report a combined score when both argument entities and relations are correct.

5.3 Baselines and Previous Models

We compare our approach with two previous approaches. The model proposed by Li and Ji (2014) is a feature-based structured perceptron model with efficient beam-search. They employ a segment-based decoder instead of token-based decoding. Their model outperformed previous state-of-the-art pipelined models. Miwa and Sasaki (2014) (SPTree) recently proposed a LSTM-based model with a sequence layer for entity identification, and a tree-based dependency layer which identifies relations between pairs of candidate entities using the shortest dependency path between them. We also employed our previous approach (Katiyar and Cardie, 2016) for extraction of opinion entities and relations to this task. We found that the performance was not competitive with the two approaches mentioned above, performing upto 10 points lower on relations. Hence, we do not include the results in Table 1. Also, Li and Ji (2014) showed that the joint model performs better than the pipelined approaches. Thus, we do not include any pipeline baselines.

5.4 Hyperparameters and Training Details

We train our model using Adadelta (Zeiler, 2012) with gradient clipping. We regularize our network using dropout (Srivastava et al., 2014) with the drop-out rate tuned using development set. We initialized our word embeddings

[3] In this paper, we only identify mention heads and hence the span is limited to a few tokens. We can also include only the last token of the gold entity span in the gold probability distribution.

	Entity			Relation			Entity+Relation		
Method	P	R	F1	P	R	F1	P	R	F1
Li and Ji (2014)	.852	.769	.808	.689	.419	.521	.654	.398	.495
SPTree	.829	.839	.834	–	–	–	.572	.540	.556
SPTree[1]	.823	.839	.831	.605	.553	.578	.578	.529	.553
Our Model	.840	.813	.826	.579	.540	.559	.555	.518	.536

Table 1: Performance on ACE05 test dataset. The dashed ("–") performance numbers were missing in the original paper (Miwa and Bansal, 2016).

[1] We ran the system made publicly available by Miwa and Bansal (2016), on ACE05 dataset for filling in the missing values and comparing our system with theirs at fine-grained level.

	Entity			Relation			Entity+Relation		
Encoding	P	R	F1	P	R	F1	P	R	F1
Left-to-Right	.821	.812	.817	.622	.449	.522	.601	.434	.504
+Multiple Relations	.835	.811	.823	.560	.492	.524	.539	.473	.504
+Bi-directional (Our Model)	.840	.813	.826	.579	.540	.559	.555	.518	.536

Table 2: Performance of different encoding methods on ACE05 dataset.

with 300-dimensional word2vec (Mikolov et al., 2013) word embeddings trained on Google News dataset. We have 3 hidden layers in our network and the dimensionality of the hidden units is 100. All the weights in the network are initialized from small random uniform noise. We tune our hyperparameters based on ACE05 development set and use them for training on ACE04 dataset.

6 Results

Table 1 compares the performance of our system with respect to the baselines on ACE05 dataset. We find that our joint model significantly outperforms the joint structured perceptron model (Li and Ji, 2014) on both entities and relations, despite the unavailability of features such as dependency trees, POS tags, etc. However, if we compare our model to the SPTree models, then we find that their model has better recall on both entities and relations. In Section 7, we perform error analysis to understand the difference in the performance of the two models in detail.

We also compare the performance of various encoding schemes in Table 2. We compare the benefits of introducing multiple relations in our objective and bi-directional encoding compared to left-to-right encoding.

Multiple Relations We find that modifying our objective to include multiple relations improves the recall of our system on relations, leading to slight improvement on the overall performance on relations. However, careful tuning of the threshold may further improve precision.

Bi-directional Encoding By adding bi-directional encoding to our system, we find that we can significantly improve the performance of our system compared to left-to-right encoding. It also improves precision compared to left-to-right decoding combined with multiple relations objective.

We find that for some relations it is easier to detect them with respect to one of the entities in the entity pair. PHYS relation is easier identified with respect to GPE entity than PER entity. Thus, our bi-directional encoding of relations allows us to encode these relations with respect to both entities in the relation.

Table 3 shows the performance of our model on ACE04 dataset. We believe that tuning the hyperparameters of our model can further improve the results on this dataset. As also pointed out by Li and Ji (2014) that ACE05 has better annotation quality, we focused on ACE05 dataset for this work.

7 Error Analysis

In this section, we perform a fine-grained comparison of our model with respect to the SPTree (Miwa and Bansal, 2016) model. We compare the performance of the two models with respect to entities, relation types and the distance between the relation arguments and provide examples from the test set in Table 6.

	Entity			Relation			Entity+Relation		
Method	P	R	F1	P	R	F1	P	R	F1
Li and Ji (2014)	.835	.762	.797	.647	.385	.483	.608	.361	.453
SPTree	.808	.829	.818	–	–	–	.487	.481	.484
Our Model	.812	.781	.796	.502	.488	.493	.464	.453	.457

Table 3: Performance on ACE04 test dataset. The dashed ("–") performance numbers were missing in the original paper (Miwa and Bansal, 2016).

7.1 Entities

We find that our model has lower recall on entity extraction than SPTree as shown in Table 1. Miwa and Bansal (2016), in one of the ablation tests on ACE05 development set, show that their model can gain upto 2% improvement in recall by entity pretraining. Since we propose a joint-model, we cannot directly apply their pretraining trick on entities separately. We leave it for future work. Li and Ji (2014) mentioned in their analysis of the dataset that there were many "UNK" tokens in the test set which were never seen during training. We verified the same and we hypothesize that for this reason the performance on the entities depends largely on the pretrained word embeddings being used. We found considerable improvements on entity recall when using pretrained word embeddings, if available, for these "UNK" tokens. Miwa and Bansal (2016) also use additional features such as POS tags in addition to pretrained word embeddings at the input layer.

Relation Type	Method	R	P	F1
ART	SPTree	.363	.552	.438
	Our model	.431	.611	**.505**
PART-WHOLE	SPTree	.560	.538	**.548**
	Our model	.520	.538	.528
PER-SOC	SPTree	.671	.671	.671
	Our model	.657	.648	.652
PHYS	SPTree	.489	.513	**.500**
	Our model	.388	.426	.406
GEN-AFF	SPTree	.414	.640	.502
	Our model	.484	.516	.500
ORG-AFF	SPTree	.692	.704	.697
	Our model	.706	.700	.703

Table 4: Performance on different relation types in ACE05 test dataset. Numbers in the bracket denote the number of relations of each relation type in the test set.

7.2 Relation Types

We evaluate our model on different relation types and compare the performance with SPTree model in Table 4. Interestingly, we find that the performance of the two models is varied over different relation types. The dependency tree-based model significantly outperforms our joint-model on PHYS and PART-WHOLE relations, whereas our model is significantly better than tree-based model on ART relation. We show an example sentence (S1) in Table 6, where SPTree model identifies the entities in ART relation correctly but fails to identify ART relation. We compare the performance with respect to PHYS relation in Section 7.3.

		Relation		
Distance	Method	R	P	F1
≤ 7	SPTree	.589	.628	.608
	Our model	.591	.605	.598
> 7	SPTree	.275	.375	**.267**
	Our model	.153	.259	.192

Table 5: Performance based on the distance between entity arguments in relations for ACE05 test dataset.

7.3 Distance-based Analysis

We also compare the performance of the two models on relations based on the distance between the entities in a relation in Table 5. We find that the performance of both the models is very low for distance greater than 7. SPTree model can identify 36 relations out of 131 such relations correctly, while our model can only identify 20 relations in this category. We manually compare the output of the two systems on these cases on several examples to understand the gain of using dependency tree on longer distances. Interestingly, the majority of these relations belong to PHYS type, thus resulting in lower performance on PHYS as discussed in Section 7.2. We found that there were a few instances of co-reference errors as shown in S2 in Table 6. Our model identifies a PHYS relation between "here" and "baghdad", whereas the gold annotation has PHYS relation between "location" and "baghdad". We think that

S1 :	the [men]$_{\text{PER:ART-1}}$ held on the sinking [vessel]$_{\text{VEH:ART-1}}$ until the [passenger]$_{\text{PER:ART-2}}$ [ship]$_{\text{VEH:ART-2}}$ was able...
SPTree :	the [men]$_{\text{PER}}$ held on the sinking [vessel]$_{\text{VEH}}$ until the [passenger]$_{\text{PER}}$ [ship]$_{\text{VEH}}$ was able to reach them.
Our Model :	the [men]$_{\text{PER:ART-1}}$ held on the sinking [vessel]$_{\text{VEH:ART-1}}$ until the [passenger]$_{\text{PER:ART-2}}$ [ship]$_{\text{VEH:ART-2}}$ was able...
S2 :	[her]$_{\text{PER}}$ research was conducted [here]$_{\text{FAC}}$ at a [location]$_{\text{FAC:PHYS1}}$ well-known to [u.n.]$_{\text{ORG:ORG-AFF1}}$ [arms]$_{\text{WEA}}$ [inspectors]$_{\text{PER:ORG-AFF1}}$. 300 miles west of [baghdad]$_{\text{GPE:PHYS1}}$.
SPTree :	[her]$_{\text{PER}}$ research was conducted [here]$_{\text{GPE}}$ at a [location]$_{\text{LOC:PHYS1}}$ well-known to u.n. [arms]$_{\text{WEA}}$ [[inspectors]$_{\text{PER:PHYS1,PHY2}}$. 300 miles west of [baghdad]$_{\text{GPE:PHYS2}}$.
Our Model :	[her]$_{\text{PER}}$ research was conducted [here]$_{\text{FAC:PHYS1}}$ at a [location]$_{\text{GPE}}$ well-known to [u.n.]$_{\text{ORG:ORG-AFF1}}$ [arms]$_{\text{WEA}}$ [inspectors]$_{\text{PER:ORG-AFF1}}$. 300 miles west of [baghdad]$_{\text{GPE:PHYS1}}$.
S3 :	... [Abigail Fletcher]$_{\text{PER:PHYS1}}$, a [marcher]$_{\text{FAC:GEN-AFF2}}$ from [Florida]$_{\text{FAC:GEN-AFF2}}$, said outside the [president]$_{\text{PER:ART3}}$'s [[residence]$_{\text{FAC:ART3, PHYS1}}$.
SPTree :	... [Abigail Fletcher]$_{\text{PER:PHYS1}}$, a [marcher]$_{\text{FAC:GEN-AFF2}}$ from [Florida]$_{\text{FAC:GEN-AFF2}}$, said outside the [president]$_{\text{PER:ART3}}$'s [[residence]]$_{\text{FAC:ART3, PHYS1}}$.
Our Model :	... [Abigail Fletcher]$_{\text{PER}}$, a [marcher]$_{\text{FAC:GEN-AFF2}}$ from [Florida]$_{\text{FAC:GEN-AFF2}}$, said outside the [president]$_{\text{PER}}$'s residence.

Table 6: Examples from the dataset with label annotations from SPTree and our model for comparison. The first row for each example is the gold standard.

incorporating these co-reference information during both training and evaluation will further improve the performance of both systems. Another source of error that we found was the inability of our system to extract entities (lower recall) as in S3. Our model could not identify the FAC entity "residence". Hence, we think an improvement on entity performance via methods like pretraining might be helpful in identifying more relations. For distance less than 7, we find that our model has better recall but lower precision, as expected.

8 Conclusion

In this paper, we propose a novel attention-based LSTM model for joint extraction of entity mentions and relations. Experimentally, we found that our model significantly outperforms feature-rich structured perceptron joint model by Li and Ji (2014). We also compare our model to an end-to-end LSTM model by Miwa and Bansal (2016) which comprises of a sequence layer for entity extraction and a tree-based dependency layer for relation classification. We find that our model, without access to dependency trees, POS tags, etc performs within 1% on entities and 2% on relations on ACE05 dataset. We also find that our model performs significantly better than their tree-based model on the ART relation, while their tree-based model performs better on PHYS and PART-WHOLE relations; the two models perform comparably on all other relation types.

In future, we plan to explore pretraining methods for our model which were shown to improve recall on entity and relation performance by Miwa and Bansal (2016). We introduce bi-directional output encoding as well as an objective to learn multiple relations in this paper. However, this presents the challenge of combining predictions from the two directions. We use heuristics in this paper to combine the predictions. We think that using probabilistic methods to combine model predictions from both directions may further improve the performance. We also plan to use Sparsemax (Martins and Astudillo, 2016) instead of Softmax for multiple relations, as the former is more suitable for multi-label classification for sparse labels.

It would also be interesting to see the effect of reranking (Collins and Koo, 2005) on our joint model. We also plan to extend the identification of entities to full entity mention span instead of only the head phrase as in Lu and Roth (2015).

Acknowledgments

We thank Qi Li and Makoto Miwa for their help with the dataset and sharing their code for analysis. We also thank Xilun Chen, Xanda Schofield, Yiqing Hua, Vlad Niculae, Tianze Shi and the three anonymous reviewers for their helpful feedback and discussion.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '05, pages 724–731. https://doi.org/10.3115/1220575.1220666.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '11, pages 551–560. http://dl.acm.org/citation.cfm?id=2002472.2002542.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1223.pdf.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 484–494. http://www.aclweb.org/anthology/P16-1046.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sydney, Australia, pages 431–439. http://www.aclweb.org/anthology/W/W06/W06-1651.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Comput. Linguist.* 31(1):25–70. https://doi.org/10.1162/0891201053630273.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* 12:2493–2537. http://dl.acm.org/citation.cfm?id=1953048.2078186.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *CoRR* abs/1504.06580. http://arxiv.org/abs/1504.06580.

Jeffrey L. Elman. 1990. Finding structure in time. *COGNITIVE SCIENCE* 14(2):179–211.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Comput. Linguist.* 28(3):245–288. https://doi.org/10.1162/089120102760275983.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL '03, pages 172–175. https://doi.org/10.3115/1119176.1119202.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Task-oriented learning of word embeddings for semantic relation classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Beijing, China, pages 268–278. http://www.aclweb.org/anthology/K15-1027.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR* abs/1508.01991. http://arxiv.org/abs/1508.01991.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 720–728. http://aclweb.org/anthology/D/D14/D14-1080.pdf.

Rohit J. Kate and Raymond J. Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, CoNLL '10, pages 203–212. http://dl.acm.org/citation.cfm?id=1870568.1870592.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1087.pdf.

Kenton Lee, Tom Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *CoRR* abs/1611.01436. http://arxiv.org/abs/1611.01436.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*. pages 402–412. http://aclweb.org/anthology/P/P14/P14-1038.pdf.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 857–867. http://aclweb.org/anthology/D15-1102.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

André F. T. Martins and Ramón F. Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. JMLR.org, ICML'16, pages 1614–1623. http://dl.acm.org/citation.cfm?id=3045390.3045561.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1105–1116. http://www.aclweb.org/anthology/P16-1105.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1858–1869. http://aclweb.org/anthology/D/D14/D14-1200.pdf.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes* 30.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence rnns for text summarization. *CoRR* abs/1602.06023. http://arxiv.org/abs/1602.06023.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*. ACM, New York, NY, USA, AKBC '13, pages 1–6. https://doi.org/10.1145/2509558.2509559.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP-CoNLL '12, pages 1201–1211. http://dl.acm.org/citation.cfm?id=2390948.2391084.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 2692–2700. http://papers.nips.cc/paper/5866-pointer-networks.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 536–540. http://aclweb.org/anthology/D15-1062.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1785–1794. http://aclweb.org/anthology/D15-1206.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers. pages 1640–1649. http://aclweb.org/anthology/P/P13/P13-1161.pdf.

Wen-Tau Yih and D. Roth. 2007. Global inference for entity and relation identification via a linear programming formulation. In L. Getoor and B. Taskar, editors, *An Introduction to Statistical Relational Learning*, MIT Press.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '10, pages 1399–1407. http://dl.acm.org/citation.cfm?id=1944566.1944726.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.* 3:1083–1106. http://dl.acm.org/citation.cfm?id=944919.944964.

Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. 2017. Neural models for sequence chunking. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*. pages 3365–3371. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14776.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 427–434. https://doi.org/10.3115/1219840.1219893.

Naturalizing a Programming Language via Interactive Learning

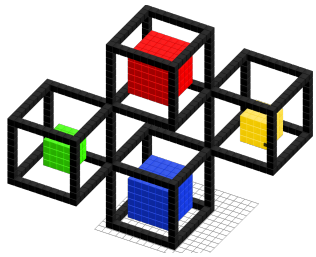
Sida I. Wang, Samuel Ginn, Percy Liang, Christopher D. Manning
Computer Science Department
Stanford University
{sidaw, samginn, pliang, manning}@cs.stanford.edu

Abstract

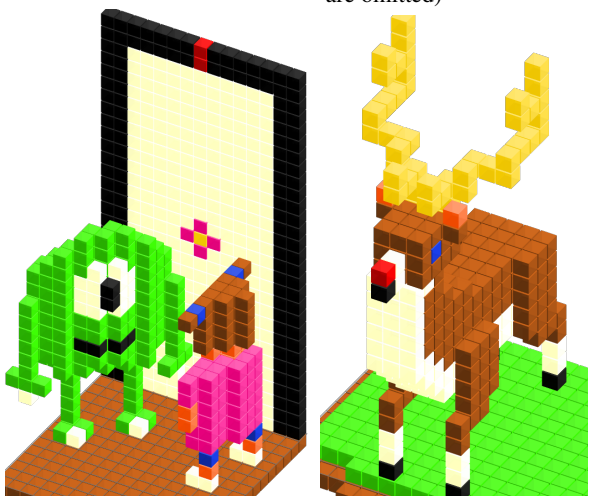
Our goal is to create a convenient natural language interface for performing well-specified but complex actions such as analyzing data, manipulating text, and querying databases. However, existing natural language interfaces for such tasks are quite primitive compared to the power one wields with a programming language. To bridge this gap, we start with a core programming language and allow users to "naturalize" the core language incrementally by defining alternative, more natural syntax and increasingly complex concepts in terms of compositions of simpler ones. In a voxel world, we show that a community of users can simultaneously teach a common system a diverse language and use it to build hundreds of complex voxel structures. Over the course of three days, these users went from using only the core language to using the naturalized language in 85.9% of the last 10K utterances.

1 Introduction

In tasks such as analyzing and plotting data (Gulwani and Marron, 2014), querying databases (Zelle and Mooney, 1996; Berant et al., 2013), manipulating text (Kushman and Barzilay, 2013), or controlling the Internet of Things (Campagna et al., 2017) and robots (Tellex et al., 2011), people need computers to perform well-specified but complex actions. To accomplish this, one route is to use a programming language, but this is inaccessible to most and can be tedious even for experts because the syntax is uncompromising and all statements have to be precise. Another route is to convert natural language into a formal language, which has been the subject of work in semantic parsing (Zettlemoyer and Collins, 2005; Artzi and Zettlemoyer, 2011, 2013; Pasupat and Liang, 2015). However, the capability of semantic parsers is still quite primitive compared to the power one wields with a programming language. This gap is increasingly limiting the potential of



Cubes: initial – select left 6 – select front 8 – black 10x10x10 frame – black 10x10x10 frame – move front 10 – red cube size 6 – move bot 2 – blue cube size 6 – green cube size 4 – (some steps are omitted)



Monsters, Inc: initial – move forward – add green monster – go down 8 – go right and front – add brown floor – add girl – go back and down – add door – add black column 30 – go up 9 – finish door – (some steps for moving are omitted)

Deer: initial – bird's eye view – deer head; up; left 2; back 2; { left antler }; right 2; {right antler} – down 4; front 2; left 3; deer body; down 6; {deer leg front}; back 7; {deer leg back}; left 4; {deer leg back}; front 7; {deer leg front} – (some steps omitted)

Figure 1: Some examples of users building structures using a naturalized language in Voxelurn: **http://www.voxelurn.com**

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 929–938
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1086

both text and voice interfaces as they become more ubiquitous and desirable.

In this paper, we propose bridging this gap with an interactive language learning process which we call *naturalization*. Before any learning, we seed a system with a core programming language that is always available to the user. As users instruct the system to perform actions, they augment the language by *defining* new utterances — e.g., the user can explicitly tell the computer that 'X' means 'Y'. Through this process, users gradually and interactively teach the system to understand the language that they *want to use*, rather than the core language that they are forced to use initially. While the first users have to learn the core language, later users can make use of everything that is already taught. This process accommodates both users' preferences and the computer action space, where the final language is both interpretable by the computer and easier to produce by human users.

Compared to interactive language learning with weak denotational supervision (Wang et al., 2016), definitions are critical for learning complex actions (Figure 1). Definitions equate a novel utterance to a sequence of utterances that the system already understands. For example, 'go left 6 and go front' might be defined as 'repeat 6 [go left]; go front', which eventually can be traced back to the expression 'repeat 6 [select left of this]; select front of this' in the core language. Unlike function definitions in programming languages, the user writes concrete values rather than explicitly declaring arguments. The system automatically extracts arguments and learns to produce the correct generalizations. For this, we propose a grammar induction algorithm tailored to the learning from definitions setting. Compared to standard machine learning, say from demonstrations, definitions provide a much more powerful learning signal: the system is told directly that 'a 3 by 4 red square' is '3 red columns of height 4', and does not have to infer how to generalize from observing many structures of different sizes.

We implemented a system called Voxelurn, which is a command language interface for a voxel world initially equipped with a programming language supporting conditionals, loops, and variable scoping etc. We recruited 70 users from Amazon Mechanical Turk to build 230 voxel structures using our system. All users teach the system at once, and what is learned from one user can be used by another user. Thus a *community* of users evolves the language to becomes more efficient over time, in a distributed way, through interaction. We show that the user community defined many new utterances—short forms, alternative syntax, and also complex concepts such as 'add green monster, add yellow plate 3 x 3'. As the system learns, users increasingly prefer to use the naturalized language over the core language: 85.9% of the last 10K accepted utterances are in the naturalized language.

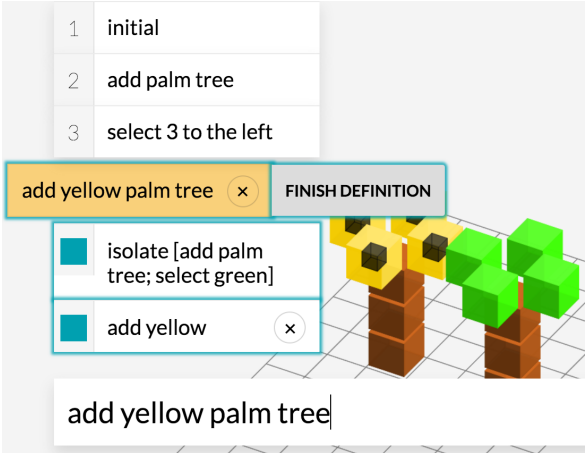


Figure 2: Interface used by users to enter utterances and create definitions.

2 Voxelurn

World. A world state in Voxelurn contains a set of voxels, where each voxel has relations 'row', 'col', 'height', and 'color'. There are two domain-specific actions, 'add' and 'move', one domain-specific relation 'direction'. In addition, the state contains a selection, which is a set of positions. While our focus is Voxelurn, we can think more generally about the world as a set of objects equiped with relations — events on a calendar, cells of a spreadsheet, or lines of text.

Core language. The system is born understanding a core language called Dependency-based Action Language (DAL), which we created (see Table 1 for an overview).

The language composes actions using the usual but expressive control primitives such as 'if', 'foreach', 'repeat', etc. Actions usually take sets as arguments, which are represented using lambda dependency-based compositional semantics (lambda DCS) expressions (Liang, 2013). Besides standard set operations like union, intersec-

Rule(s)	Example(s)	Description
$A \rightarrow A$; A	select left; add red	perform actions sequentially
$A \rightarrow$ repeat N A	repeat 3-1 add red top	repeat action N times
$A \rightarrow$ if S A	if has color red [select origin]	action if S is non-empty
$A \rightarrow$ while S A	while not has color red [select left of this]	action while S is non-empty
$A \rightarrow$ foreach S A	foreach this [remove has row row of this]	action for each item in S
$A \rightarrow [A]$	[select left or right; add red; add red top]	group actions for precedence
$A \rightarrow \{A\}$	{select left; add red}	scope only selection
$A \rightarrow$ isolate A	isolate [add red top; select has color red]	scope voxels and selection
$A \rightarrow$ select S	select all and not origin	set the selection
$A \rightarrow$ remove S	remove has color red	remove voxels
$A \rightarrow$ update R S	update color [color of left of this]	change property of selection
S	this	current selection
S	all | none | origin	all voxels, empty set, $(0, 0)$
R of S | has R S	has color red or yellow | has row [col of this]	lambda DCS joins
not S | S and S | S or S	this or left and not has color red	set operations
N | N+N | N-N	1,...,10 | 1+2 | row of this + 1	numbers and arithmetic
argmax R S | argmin R S	argmax col has color red	superlatives
R	color | row | col | height | top | left | $\cdots$	voxel relations
C	red | orange | green | blue | black | $\cdots$	color values
D	top | bot | front | back | left | right	direction values
$S \rightarrow$ very D of S	very top of very bot of has color green	syntax sugar for argmax
$A \rightarrow$ add C [D] | move D	add red | add yellow bot | move left	add voxel, move selection

Table 1: Grammar of the core language (DAL), which includes actions (A), relations (R), and sets of values (S). The grammar rules are grouped into four categories. From top to bottom: domain-general action compositions, actions using sets, lambda DCS expressions for sets, and domain-specific relations and actions.

tion and complement, lambda DCS leverages the tree dependency structure common in natural language: for the relation 'color', 'has color red' refers to the set of voxels that have color red, and its reverse 'color of has row 1' refers to the set of colors of voxels having row number 1. Tree-structured joins can be chained without using any variables, e.g., 'has color [yellow or color of has row 1]'.

We protect the core language from being redefined so it is always precise and usable.[1] In addition to expressivity, the core language *interpolates* well with natural language. We avoid explicit variables by using a *selection*, which serves as the default argument for most actions.[2] For example, 'select has color red; add yellow top; remove' adds yellow on top of red voxels and then removes the red voxels.

To enable the building of more complex structures in a more modular way, we introduce a notion of *scoping*. Suppose one is operating on one of the palm trees in Figure 2. The user might want to use 'select all' to select only the voxels in that tree rather than all of the voxels in the scene. In general, an action A can be viewed as taking a set of voxels v and a selection s, and producing an updated set of voxels v' and a modified selection s'. The default scoping is '[A]', which is the same as 'A' and returns (v', s'). There are two constructs that alter the flow: First, '{A}' takes (v, s) and returns (v', s), thus restoring the selection. This allows A to use the selection as a temporary variable without affecting the rest of the program. Second, 'isolate [A]' takes (v, s), calls A with (s, s) (restricting the set of voxels to just the selection) and returns (v'', s), where v'' consists of voxels in v' and voxels in v that occupy empty locations in v'. This allows A to focus only on the selection (e.g., one of the palm trees). Although scoping can be explicitly controlled via

[1] Not doing so resulted in ambiguities that propagated uncontrollably, e.g., once 'red' can mean many different colors.

[2] The selection is like the turtle in LOGO, but can be a set.

'[]', 'isolate', and '{}', it is an unnatural concept for non-programmers. Therefore when the choice is not explicit, the parser generates all three possible scoping interpretations, and the model learns which is intended based on the user, the rule, and potentially the context.

3 Learning interactively from definitions

The goal of the user is to build a structure in Voxelurn. In Wang et al. (2016), the user provided interactive supervision to the system by selecting from a list of candidates. This is practical when there are less than tens of candidates, but is completely infeasible for a complex action space such as Voxelurn. Roughly, 10 possible colors over the $3 \times 3 \times 4$ box containing the palm tree in Figure 2 yields 10^{36} distinct denotations, and many more programs. Obtaining the structures in Figure 1 by selecting candidates alone would be infeasible.

This work thus uses *definitions* in addition to selecting candidates as the supervision signal. Each definition consists of a *head* utterance and a *body*, which is a sequence of utterances that the system understands. One use of definitions is paraphrasing and defining alternative syntax, which helps naturalize the core language (e.g., defining 'add brown top 3 times' as 'repeat 3 add brown top'). The second use is building up complex concepts hierarchically. In Figure 2, 'add yellow palm tree' is defined as a sequence of steps for building the palm tree. Once the system understands an utterance, it can be used in the body of other definitions. For example, Figure 3 shows the full definition tree of 'add palm tree'. Unlike function definitions in a programming language, our definitions do not specify the exact arguments; the system has to learn to extract arguments to achieve the correct generalization.

The interactive definition process is described in Figure 4. When the user types an utterance x, the system parses x into a list of candidate programs. If the user selects one of them (based on its denotation), then the system executes the resulting program. If the utterance is unparsable or the user rejects all candidate programs, the user is asked to provide the definition body for x. Any utterances in the body not yet understood can be defined recursively. Alternatively, the user can first execute a sequence of commands X, and then provide a head utterance for body X.

When constructing the definition body, users can type utterances with multiple parses; e.g., 'move forward' could either modify the selection ('select front') or move the voxel ('move front'). Rather than propagating this ambiguity to the head, we force the user to commit to one interpretation by selecting a particular candidate. Note that we are using interactivity to control the exploding ambiguity.

```
def: add palm tree
    def: brown trunk height 3
        def: add brown top 3 times
            repeat 3 [add brown top]
    def: go to top of tree
        select very top of has color brown
    def: add leaves here
        def: select all sides
            select left or right or front or back
        add green
```

Figure 3: Defining 'add palm tree', tracing back to the core language (utterances without **def:**).

begin execute x:
  if x *does not parse* **then** define x;
  if *user rejects all parses* **then** define x;
  execute user choice

begin define x:
  repeat starting with $X \leftarrow [\,]$
    user enters x';
    if x' *does not parse* **then** define x';
    if *user rejects all* x' **then** define x';
    $X \leftarrow [X; x']$;
  until *user accepts* X *as the def'n of* x;

Figure 4: When the user enters an utterance, the system tries to parse and execute it, or requests that the user define it.

4 Model and learning

Let us turn to how the system learns and predicts. This section contains prerequisites before we describe definitions and grammar induction in Section 5.

Semantic parsing. Our system is based on a semantic parser that maps utterances x to programs z, which can be executed on the current state s (set of voxels and selection) to produce the next state $s' = [\![z]\!]_s$. Our system is implemented as the interactive package in SEMPRE (Berant et al., 2013);

Feature	Description
Rule.ID	ID of the rule
Rule.Type	core?, used?, used by others?
Social.Author	ID of author
Social.Friends	(ID of author, ID of user)
Social.Self	rule is authored by user?
Span	(left/right token(s), category)
Scope	type of scoping for each user

Table 2: Summary of features.

see Liang (2016) for a gentle exposition.

A *derivation* d represents the process by which an utterance x turns into a program $z = \text{prog}(d)$. More precisely, d is a tree where each node contains the corresponding span of the utterance $(\text{start}(d), \text{end}(d))$, the grammar rule $\text{rule}(d)$, the grammar category $\text{cat}(d)$, and a list of child derivations $[d_1, \ldots, d_n]$.

Following Zettlemoyer and Collins (2005), we define a log-linear model over derivations d given an utterance x produced by the user u:

$$p_\theta(d \mid x, u) \propto \exp(\theta^\mathsf{T} \phi(d, x, u)), \qquad (1)$$

where $\phi(d, x, u) \in \mathbb{R}^p$ is a feature vector and $\theta \in \mathbb{R}^p$ is a parameter vector. The user u does not appear in previous work on semantic parsing, but we use it to personalize the semantic parser trained on the community.

We use a standard chart parser to construct a chart. For each chart cell, indexed by the start and end indices of a span, we construct a list of partial derivations recursively by selecting child derivations from subspans and applying a grammar rule. The resulting derivations are sorted by model score and only the top K are kept. We use $\text{chart}(x)$ to denote the set of all partial derivations across all chart cells. The set of grammar rules starts with the set of rules for the core language (Table 1), but grows via grammar induction when users add definitions (Section 5). Rules in the grammar are stored in a trie based on the right-hand side to enable better scalability to a large number of rules.

Features. Derivations are scored using a weighted combination of features. There are three types of features, summarized in Table 2.

Rule features fire on each rule used to construct a derivation. ID features fire on specific rules (by ID). Type features track whether a rule is part of the core language or induced, whether it has been used again after it was defined, if it was used by someone other than its author, and if the user and the author are the same ($5 +$ #rules features).

Social features fire on properties of rules that capture the unique linguistic styles of different users and their interaction with each other. Author features capture the fact that some users provide better, and more generalizable definitions that tend to be accepted. Friends features are cross products of author ID and user ID, which captures whether rules from a particular author are systematically preferred or not by the current user, due to stylistic similarities or differences (#users+#users×#users features).

Span features include conjunctions of the category of the derivation and the leftmost/rightmost token on the border of the span. In addition, span features include conjunctions of the category of the derivation and the 1 or 2 adjacent tokens just outside of the left/right border of the span. These capture a weak form of context-dependence that is generally helpful ($<\approx V^4 \times$ #cats features for a vocabulary of size V).

Scoping features track how the community, as well as individual users, prefer each of the 3 scoping choices (none, selection only '{A}', and voxels+selection 'isolate {A}'), as described in Section 2. 3 global indicators, and 3 indicators for each user fire every time a particular scoping choice is made ($3 + 3 \times$ #users features).

Parameter estimation. When the user types an utterance, the system generates a list of candidate next states. When the user chooses a particular next state s' from this list, the system performs an online AdaGrad update (Duchi et al., 2010) on the parameters θ according to the gradient of the following loss function:

$$-\log \sum_{d:[\![\text{prog}(d)]\!]_s = s'} p_\theta(d \mid x, u) + \lambda ||\theta||_1,$$

which attempts to increase the model probability on derivations whose programs produce the next state s'.

5 Grammar induction

Recall that the main form of supervision is via user definitions, which allows creation of user-defined concepts. In this section, we show how to turn

these definitions into new grammar rules that can be used by the system to parse new utterances.

Previous systems of grammar induction for semantic parsing were given utterance-program pairs (x, z). Both the GENLEX (Zettlemoyer and Collins, 2005) and higher-order unification (Kwiatkowski et al., 2010) algorithms overgenerate rules that liberally associate parts of x with parts of z. Though some rules are immediately pruned, many spurious rules are undoubtedly still kept. In the interactive setting, we must keep the number of candidates small to avoid a bad user experience, which means a higher precision bar for new rules.

Fortunately, the structure of definitions makes the grammar induction task easier. Rather than being given an utterance-program (x, z) pair, we are given a definition, which consists of an utterance x (head) along with the body $X = [x_1, \ldots, x_n]$, which is a sequence of utterances. The body X is fully parsed into a derivation d, while the head x is likely only partially parsed. These partial derivations are denoted by $\text{chart}(x)$.

At a high-level, we find *matches*—partial derivations $\text{chart}(x)$ of the head x that also occur in the full derivation d of the body X. A grammar rule is produced by substituting any set of non-overlapping matches by their categories. As an example, suppose the user defines

'add red top times 3' as 'repeat 3 [add red top]'.

Then we would be able to induce the following two grammar rules:

$$A \rightarrow \text{add } C\ D \text{ times } N\ :$$
$$\lambda CDN.\text{repeat } N\ [\text{add } C\ D]$$
$$A \rightarrow A \text{ times } N\ :$$
$$\lambda AN.\text{repeat } N\ [A]$$

The first rule substitutes primitive values ('red', 'top', and '3') with their respective pre-terminal categories (C, D, N). The second rule contains compositional categories like actions (A), which require some care. One might expect that greedily substituting the largest matches or the match that covers the largest portion of the body would work, but the following example shows that this is not the case:

$$\overbrace{\underbrace{\text{add red left}}_{A_2}}^{A_1} \text{ and here} = \overbrace{\underbrace{\text{add red left;}}_{A_2}}^{A_1}\ \overbrace{\text{add red}}^{A_1}$$

Here, both the highest coverage substitution (A_1: 'add red', which covers 4 tokens of the body), and the largest substitution available (A_2: 'add red left') would generalize incorrectly. The correct grammar rule only substitutes the primitive values ('red', 'left').

5.1 Highest scoring abstractions

We now propose a grammar induction procedure that optimizes a more global objective and uses the learned semantic parsing model to choose substitutions. More formally, let M be the set of partial derivations in the head whose programs appear in the derivation d_X of the body X:

$$M \stackrel{\text{def}}{=} \{d \in \text{chart}(x) :$$
$$\exists d' \in \text{desc}(d_X) \wedge \text{prog}(d) = \text{prog}(d')\},$$

where $\text{desc}(d_X)$ are the descendant derivations of d_X. Our goal is to find a *packing* $P \subseteq M$, which is a set of derivations corresponding to non-overlapping spans of the head. We say that a packing P is maximal if no other derivations may be added to it without creating an overlap.

Let $\text{packings}(M)$ denote the set of maximal packings, we can frame our problem as finding the maximal packing that has the highest score under our current semantic parsing model:

$$P_L^* = \underset{P \in \text{packings}(M);}{\text{argmax}} \sum_{d \in P} \text{score}(d). \quad (2)$$

Finding the highest scoring packing can be done using dynamic programming on P_i^* for $i = 0, 1, \ldots, L$, where L is the length of x and $P_0^* = \emptyset$. Since $d \in M$, $\text{start}(d)$ and $\text{end}(d)$ (exclusive) refer to span in the head x. To obtain this dynamic program, let D_i be the highest scoring maximal packing containing a derivation ending *exactly* at position i (if it exists):

$$D_i = \{d_i\} \cup P^*_{\text{start}(d_i)}, \quad (3)$$
$$d_i = \underset{d \in M; \text{end}(d) = i}{\text{argmax}} \text{score}(d \cup P^*_{\text{start}(d)}). \quad (4)$$

Then the maximal packing of up to i can be defined recursively as

$$P_i^* = \underset{D \in \{D_{s(i)+1}, D_{s(i)+2}, \ldots, D_i\}}{\text{argmax}} \text{score}(D) \quad (5)$$
$$s(i) = \max_{d: \text{end}(d) \leq i} \text{start}(d), \quad (6)$$

Input : x, d_X, P^*
Output: rule
$r \leftarrow x$;
$f \leftarrow d_X$;
for $d \in P^*$ **do**
  $r \leftarrow r[\text{cat}(d)/\text{span}(d)]$
  $f \leftarrow \lambda\, \text{cat}(d).f[\text{cat}(d)/d]$
return *rule* $(\text{cat}(d_X) \rightarrow r : f)$

Algorithm 1: Extract a rule r from a derivation d_X of body X and a packing P^*. Here, $f[t/s]$ means substituting s by t in f, with the usual care about names of bound variables.

where $s(i)$ is the largest index such that $D_{s(i)}$ is no longer maximal for the span $(0, i)$ (i.e. there is a $d \in M$ on the span $\text{start}(d) \geq s(i) \wedge \text{end}(d) \leq i$.

Once we have a packing $P^* = P^*_L$, we can go through $d \in P^*$ in order of $\text{start}(d)$, as in Algorithm 1. This generates one high precision rule per packing per definition. In addition to the highest scoring packing, we also use a "simple packing", which includes only primitive values (in Voxelurn, these are colors, numbers, and directions). Unlike the simple packing, the rule induced from the highest scoring packing does not always generalize correctly. However, a rule that often generalizes incorrectly should be down-weighted, along with the score of its packings. As a result, a different rule might be induced next time, even with the same definition.

5.2 Extending the chart via alignment

Algorithm 1 yields high precision rules, but fails to generalize in some cases. Suppose that 'move up' is defined as 'move top', where 'up' does not parse, and does not match anything. We would like to infer that 'up' means 'top'. To handle this, we leverage a property of definitions that we have not used thus far: the utterances themselves. If we align the head and body, then we would intuitively expect aligned phrases to correspond to the same derivations. Under this assumption, we can then transplant these derivations from d_X to $\text{chart}(x)$ to create new matches. This is more constrained than the usual alignment problem (e.g., in machine translation) since we only need to consider spans of X which corresponds to derivations in $\text{desc}(d_X)$.

Algorithm 2 provides the algorithm for extending the chart via alignments. The aligned function is implemented using the following two heuristics:

Input : x, X, d_X
for $d \in \text{desc}(d_X)$, $x' \in \text{spans}(x)$ **do**
  if $\text{aligned}(x', d, (x, X))$ **then**
    $d' \leftarrow d$;
    $\text{start}(d') \leftarrow \text{start}(x')$;
    $\text{end}(d') \leftarrow \text{end}(x')$;
    $\text{chart}(x) \leftarrow \text{chart}(x) \cup d'$
  end
end

Algorithm 2: Extending the chart by alignment: If d is aligned with x' based on the utterance, then we pretend that x' should also parse to d, and d is transplanted to $\text{chart}(x)$ as if it parsed from x'.

- **exclusion**: if all but 1 pair of short spans (1 or 2 tokens) are matched, the unmatched pair is considered aligned.
- **projectivity**: if $d_1, d_2 \in \text{desc}(d_X) \cap \text{chart}(x)$, then $\text{ances}(d_1, d_2)$ is aligned to the corresponding span in x.

With the extended chart, we can run the algorithm from Section 5.1 to induce rules. The transplanted derivations (e.g., 'up') might now form new matches which allows the grammar induction to induce more generalizable rules. We only perform this extension when the body consists of one utterance, which tend to be a paraphrase. Bodies with multiple utterances tend to be new concepts (e.g., 'add green monster'), for which alignment is impossible. Because users have to select from candidates parses in the interactive setting, inducing low precision rules that generate many parses degrade the user experience. Therefore, we induce alignment-based rules conservatively—only when all but 1 or 2 tokens of the head aligns to the body and vice versa.

6 Experiments

Setup. Our ultimate goal is to create a community of users who can build interesting structures in Voxelurn while naturalizing the core language. We created this community using Amazon Mechanical Turk (AMT) in two stages. First, we had *qualifier* tasks, in which an AMT worker was instructed to replicate a fixed target exactly (Figure 5), ensuring that the initial users are familiar with at least some of the core language, which is the starting point of the naturalization process.

Figure 5: The target used for the qualifier.

Next, we allowed the workers who qualified to enter the second *freebuilding* task, in which they were asked to build any structure they wanted in 30 minutes. This process was designed to give users freedom while ensuring quality. The analogy of this scheme in a real system is that early users (or a small portion of expert users) have to make some learning investment, so the system can learn and become easier for other users.

Statistics. 70 workers passed the qualifier task, and 42 workers participated in the final freebuilding experiment. They built 230 structures. There were over 103,000 queries consisting of 5,388 distinct token types. Of these, 64,075 utterances were tried and 36,589 were accepted (so an action was performed). There were 2,495 definitions combining over 15,000 body utterances with 6.5 body utterances per head on average (96 max). From these definitions, 2,817 grammar rules were induced, compared to less than 100 core rules. Over all queries, there were 8.73 parses per utterance on average (starting from 1 for core).

Is naturalization happening? The answer is yes according to Figure 6, which plots the cummulative percentage of utterances that are core, induced, or unparsable. To rule out that more induced utterances are getting rejected, we consider only accepted utterances in the middle of Figure 6, which plots the percentage of induced rules among accepted utterances for the entire community, as well as for the 5 heaviest users. Since unparsable utterances cannot be accepted, accepted core (which is not shown) is the complement of accepted induced. At the conclusion of the experiment, 72.9% of all accepted utterances are induced—this becomes 85.9% if we only consider the final 10,000 accepted utterances.

Three modes of naturalization are outlined in Table 3. For very common operations, like moving the selection, people found ‘select left’ too verbose and shorterned this to `|, left, >, sel |`. One user preferred ‘go down and right’ instead of ‘select bot; select right’ in core and defined it as ‘go down; go right’. Definitions for high-level

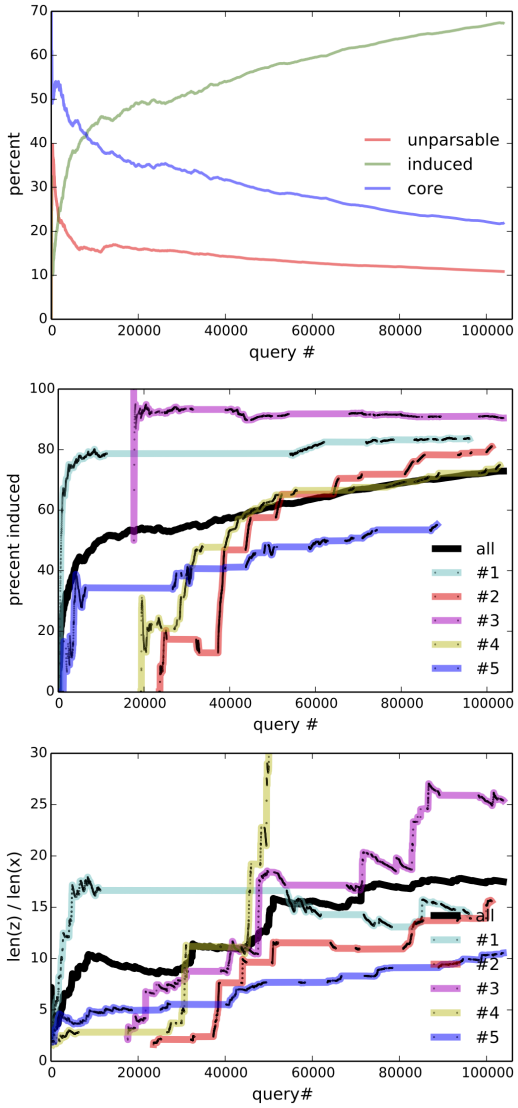


Figure 6: Learning curves. **Top:** percentage of all utterances that are part of the *core* language, the *induced* language, or *unparsable* by the system. **Middle:** percentage of accepted utterances belonging to the induced language, overall and for the 5 heaviest users. **Bottom:** expressiveness measured by the ratio of the length of the program to the length of the corresponding utterance.

concepts tend to be whole objects that are not parameterized (e.g., ‘dancer’). The bottom plot of Figure 6 suggests that users are defining and using higher level concepts, since programs become longer relative to utterances over time.

As a result of the automatic but implicit grammar induction, some concepts do not generalize correctly. In definition head ‘3 tall 9 wide white tower centered here’, arguments do not match the body; for ‘black 10x10x10 frame’, we failed to tokenize.

Short forms
left, l, mov left, go left, <, sel left
br, black, blu, brn, orangeright, left3
add row brn left 5 := add row brown left 5
Alternative syntax
go down and right := go down; go right
select orange := select has color orange
add red top 4 times := repeat 4 [add red top]
l white := go left and add white
mov up 2 := repeat 2 [select up]
go up 3 := go up 2; go up
Higher level
add red plate 6 x 7, green cube size 4,
add green monster, black 10x10x10 frame,
flower petals, deer leg back, music box, dancer

Table 3: Example definitions. See CodaLab worksheet for the full leaderboard.

Learned parameters. Training using L1 regularization, we obtained 1713 features with nonzero parameters. One user defined many concepts consisting of a single short token, and the Social.Author feature for that user has the most negative weight overall. With user compatibility (Social.Friends), some pairs have large positive weights and others large negative weights. The 'isolate' scoping choice (which allows easier hierarchical building) received the most positive weights, both overall and for many users. The 2 highest scoring induced rules correspond to 'add row red right 5' and 'select left 2'.

Incentives. Having complex structures show that the actions in Voxelurn are expressive and that hierarchical definitions are useful. To incentivize this behavior, we created a leaderboard which ranked structures based on recency and upvotes (like Hacker News). Over the course of 3 days, we picked three prize categories to be released daily. The prize categories for each day were bridge, house, animal; tower, monster, flower; ship, dancer, and castle.

To incentivize more definitions, we also track *citations*. When a rule is used in an accepted utterance by another user, the rule (and its author) receives a citation. We pay bonuses to top users according to their h-index. Most cited definitions are also displayed on the leaderboard. Our qualitative results should be robust to the incentives scheme, because the users do not overfit to the incentives—e.g., around 20% of the structures are not in the prize categories and users define complex concepts that are rarely cited.

7 Related work and discussion

This work is an evolution of Wang et al. (2016), but differs crucially in several ways: While Wang et al. (2016) starts from scratch and relies on selecting candidates, this work starts with a programming language (PL) and additionally relies on definitions, allowing us to scale. Instead of having a private language for each user, the user community in this work shares one language.

Azaria et al. (2016) presents Learning by Instruction Agent (LIA), which also advocates learning from users. They argue that developers cannot anticipate all the actions that users want, and that the system cannot understand the corresponding natural language even if the desired action is built-in. Like Jia et al. (2017), Azaria et al. (2016) starts with an ad-hoc set of initial slot-filling commands in natural language as the basis of further instructions—our approach starts with a more expressive core PL designed to interpolate with natural language. Compared to previous work, this work studied interactive learning in a shared community setting and hierarchical definitions resulting in more complex concepts.

Allowing ambiguity and a flexible syntax is a key reason why natural language is easier to produce—this cannot be achieved by PLs such as Inform and COBOL which look like natural language. In this work, we use semantic parsing techniques that can handle ambiguity (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010; Liang et al., 2011; Pasupat and Liang, 2015). In semantic parsing, the semantic representation and action space is usually designed to accommodate the natural language that is considered constant. In contrast, the action space is considered constant in the naturalizing PL approach, and the language adapts to be more natural while accommodating the action space.

Our work demonstrates that interactive definitions is a strong and usable form of supervision. In the future, we wish to test these ideas in more domains, naturalize a real PL, and handle paraphrasing and implicit arguments. In the process of naturalization, both data and the semantic grammar have important roles in the evolution of a language that is easier for humans to produce while still parsable by computers.

Acknowledgments. We thank our reviewers, Panupong (Ice) Pasupat for helpful suggestions and discussions on lambda DCS, DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF-15-1-0462, and NSF CAREER Award no. IIS-1552635.

Reproducibility. All code, data, and experiments for this paper are available on the CodaLab platform:
`https://worksheets.codalab.org/worksheets/0xbf8f4f5b42e54eba9921f7654b3c5c5d`
and a demo: `http://www.voxelurn.com`

References

Y. Artzi and L. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 421–432.

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

A. Azaria, J. Krishnamurthy, and T. M. Mitchell. 2016. Instructable intelligent personal agent. In *Association for the Advancement of Artificial Intelligence (AAAI)*. pages 2681–2689.

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

G. Campagna, R. Ramesh, S. Xu, M. Fischer, and M. S. Lam. 2017. Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In *World Wide Web (WWW)*. pages 341–350.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

S. Gulwani and M. Marron. 2014. NLyze: interactive programming by natural language for spreadsheet data analysis and manipulation. In *International Conference on Management of Data, SIGMOD*. pages 803–814.

R. Jia, L. Heck, D. Hakkani-Tür, and G. Nikolov. 2017. Learning concepts through conversations in spoken dialogue systems. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

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

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

P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408* .

P. Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM* 59.

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

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

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

S. I. Wang, P. Liang, and C. Manning. 2016. Learning language games through interaction. In *Association for Computational Linguistics (ACL)*.

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

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

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

Semantic Word Clusters Using Signed Spectral Clustering

João Sedoc, Jean Gallier, Lyle Ungar
Computer & Information Science
University of Pennsylvania
joao, jean, ungar@cis.upenn.edu

Dean Foster
Amazon LLC
dean@foster.net

Abstract

Vector space representations of words capture many aspects of word similarity, but such methods tend to produce vector spaces in which antonyms (as well as synonyms) are close to each other. For spectral clustering using such word embeddings, words are points in a vector space where synonyms are linked with positive weights, while antonyms are linked with negative weights. We present a new signed spectral normalized graph cut algorithm, *signed clustering*, that overlays existing thesauri upon distributionally derived vector representations of words, so that antonym relationships between word pairs are represented by negative weights. Our signed clustering algorithm produces clusters of words that simultaneously capture distributional and synonym relations. By using randomized spectral decomposition (Halko et al., 2011) and sparse matrices, our method is both fast and scalable. We validate our clusters using datasets containing human judgments of word pair similarities and show the benefit of using our word clusters for sentiment prediction.

1 Introduction

In distributional vector representations, opposite relations are not fully captured. Take, for example, words such as "great" and "awful" that can appear with similar frequency in the same sentence structure: "John had a great meeting" and "John had an awful day." Word embeddings, which are successful in a wide array of NLP tasks (Turney et al., 2010; Dhillon et al., 2015), fail to capture this antonymy because they follow the *distributional hypothesis* that similar words are used in similar contexts (Harris, 1954), thus assigning small cosine or euclidean distances between the vector representations of "great" and "awful".

While vector space models (Turney et al., 2010) such as word2vec (Mikolov et al., 2013), Global vectors (GloVe) (Pennington et al., 2014), or Eigenwords (Dhillon et al., 2015) capture relatedness, they do not adequately encode synonymy and semantic similarity (Mohammad et al., 2013; Scheible et al., 2013). Our goal is to create clusters of synonyms or semantically equivalent words and linguistically motivated unified constructs. Signed graphs, which are graphs with negative edge weights, were first introduced by Cartwright and Harary (1956). However, signed graph clustering for multiclass normalized cuts (K-clusters) has been largely unexplored until recently. We present a novel theory and method that extends multiclass normalized cuts (K-cluster) of Yu and Shi (2003) to signed graphs (Gallier, 2016)[1] and the work of Kunegis et al. (2010) to K-clustering. This extension allows the incorporation of knowledge base information, positive and negatively weighted links (see figure 2.1). Negative edges serve as repellent or opposite relationships between nodes.

Our signed spectral normalized graph cut algorithm (henceforth, signed clustering) builds negative edge relations into graph embeddings using similarity structure in vector spaces. It takes as input an initial set of vectors and edge relations, and hence is easy to combine with any word embedding method. This paper formally improves on the discrete optimization problem of Yu and Shi (2003).

Signed clustering gives better clusters than spectral clustering (Shi and Malik, 2000) of word embeddings, and it has better coverage and is more robust than thesaurus look-up. This is because the-

[1] Gallier (2016) is a full theoretical exposition of our methods with proofs on arXiv.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 939–949
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1087

sauri erroneously give equal weight to rare senses of a word – for example, "rich" as a rarely used synonym of "absurd". Also, the overlap between thesauri is small, due to their manual creation. Lin (1998) found 17.8397% overlap between synonym sets from Roget's Thesaurus and WordNet 1.5. We find similarly small overlap between all three thesauri tested.

We evaluate our clusters using SimLex-999 (Hill et al., 2014) and SimVerb-3500 (Gerz et al., 2016) as a ground truth for our cluster evaluation. Finally, we test our method on the sentiment analysis task. Overall, signed spectral clustering can augment methods using signed information and has broad application for many fields.

Our main contributions are: the novel extension of signed clustering to the multiclass (K-cluster), and the application of this method to create semantic word clusters that are agnostic to vector space representations and thesauri.

1.1 Related Work

Semantic word cluster and distributional thesauri have been well studied in the NLP literature (Lin, 1998; Curran, 2004). Recently there has been a line of research on incorporating synonyms and antonyms into word embeddings. Our approach is very much in the line of Vlachos et al. (2009). However, they explicitly made verb clusters using Dirichlet Process Mixture Models and must-link / cannot-link clustering. Furthermore, they note that cannot-link clustering does not improve performance whereas our signed clustering antonyms are key.

Most recent models either attempt to make richer contexts, in order to find semantic similarity, or overlay thesaurus information in a supervised or semi-supervised manner. One line of active research is post processing the word vector embedding by transforming the space using a single or multi-relational objective (Yih et al., 2012; Tang et al., 2014; Chang et al., 2013; Tang et al., 2014; Zhang et al., 2014; Faruqui et al., 2015; Mrkšić et al., 2016).

Alternatively, there are methods to modify the objective function for generating the word embeddings (Ono et al., 2015; Pham et al., 2015; Schwartz et al., 2015).

Our approach differs from the aforementioned methods in that we created word clusters using the antonym relationships as negative links. Unlike the previous approaches using semi-supervised methods, we incorporated the thesauri as a knowledge base. Similar to word vector retrofitting and counter-fitting methods described in Faruqui et al. (2015) and Mrkšić et al. (2016), our signed clustering method uses existing vector representations to create word clusters.

To our knowledge, this work is the first theoretical foundation of multiclass signed normalized cuts.[2] Zass and Shashua (2005) solved multiclass cluster from another approach, by relaxing the orthogonality assumption and focusing instead on the non-negativity constraint. This led to a doubly stochastic optimization problem. Negative edges are handled by a constrained hyperparameter. Hou (2005) used positive degrees of nodes in the degree matrix of a signed graph with weights (-1, 0, 1), which was advanced by Kolluri et al. (2004) and Kunegis et al. (2010) using absolute values of weights in the degree matrix. Interestingly, Chiang et al. (2014) presented a theoretical foundation for edge sign prediction and a recursive clustering approach. Mercado et al. (2016) found that using the geometric mean of the graph Laplacian improves performance.

Wang et al. (2016) used semi-supervised polarity induction (Rao and Ravichandran, 2009) to create clusters of words with similar valence and arousal. Must-link and cannot-link soft spectral clustering (Rangapuram and Hein, 2012) share similarities with our method, particularly in the limit where there are no must-link edges present. Both must-link and cannot-link clustering as well as polarity induction differ in optimization method. Our method is significantly faster due to the use of randomized SVD (Halko et al., 2011) and can thus be applied to large scale NLP problems.

We developed a novel theory and algorithm that extends the clustering of Shi and Malik (2000) and Yu and Shi (2003) to the multiclass signed graph case.

2 Signed Graph Cluster Estimation

2.1 Signed Normalized Cut

Weighted graphs for which the weight matrix is a symmetric matrix in which negative and positive entries are allowed are called *signed graphs*.

[2]The full exposition by Gallier (2016) is available on arXiv.

Such graphs (with weights $(-1, 0, +1)$) were introduced as early as 1953 by (Harary, 1953), to model social relations involving disliking, indifference, and liking. The problem of clustering the nodes of a signed graph arises naturally as a generalization of the clustering problem for weighted graphs. Figure 1 shows a signed graph of word similarities with a thesaurus overlay. Gallier

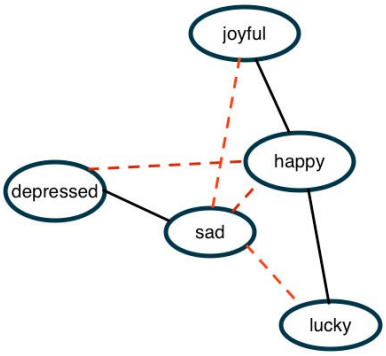


Figure 1: Signed graph of words using a distance metric from the word embedding. The red dashed edges represent the antonym relation while solid edges represent synonymy relations.

(2016) extends normalized cuts to signed graphs in order to incorporate antonym information into word clusters.

Definition 2.1. A *weighted graph* is a pair $G = (V, W)$, where $V = \{v_1, \ldots, v_m\}$ is a set of *nodes* or *vertices*, and W is a symmetric matrix called the *weight matrix*, such that $w_{ij} \geq 0$ for all $i, j \in \{1, \ldots, m\}$, and $w_{ii} = 0$ for $i = 1, \ldots, m$. We say that a set $\{v_i, v_j\}$ is an edge iff $w_{ij} > 0$. The corresponding (undirected) graph (V, E) with $E = \{\{v_i, v_j\} \mid w_{ij} > 0\}$, is called the *underlying graph* of G.

Given a signed graph $G = (V, W)$ (where W is a symmetric matrix with zero diagonal entries), the *underlying graph* of G is the graph with node set V and set of (undirected) edges $E = \{\{v_i, v_j\} \mid w_{ij} \neq 0\}$.

If (V, W) is a signed graph, where W is an $m \times m$ symmetric matrix with zero diagonal entries and with the other entries $w_{ij} \in \mathbb{R}$ arbitrary, for any node $v_i \in V$, the *signed degree* of v_i is defined as

$$\overline{d}_i = \overline{d}(v_i) = \sum_{j=1}^{m} |w_{ij}|,$$

and the *signed degree matrix* $\overline{D}$ as

$$\overline{D} = \mathrm{diag}(\overline{d}(v_1), \ldots, \overline{d}(v_m)).$$

For any subset A of the set of nodes V, let

$$\mathrm{vol}(A) = \sum_{v_i \in A} \overline{d}_i = \sum_{v_i \in A} \sum_{j=1}^{m} |w_{ij}|.$$

For any two subsets A and B of V and A^C which is the complement of A, define $\mathrm{links}^+(A, B)$, $\mathrm{links}^-(A, B)$, and $\mathrm{cut}(A, A^C)$ by

$$\mathrm{links}^+(A, B) = \sum_{\substack{v_i \in A, v_j \in B \\ w_{ij} > 0}} w_{ij}$$

$$\mathrm{links}^-(A, B) = \sum_{\substack{v_i \in A, v_j \in B \\ w_{ij} < 0}} -w_{ij}$$

$$\mathrm{cut}(A, A^C) = \sum_{\substack{v_i \in A, v_j \in A^C \\ w_{ij} \neq 0}} |w_{ij}|.$$

Then, the *signed Laplacian* $\overline{L}$ is defined by

$$\overline{L} = \overline{D} - W,$$

and its normalized version $\overline{L}_{\mathrm{sym}}$ by

$$\overline{L}_{\mathrm{sym}} = \overline{D}^{-1/2} \overline{L} \overline{D}^{-1/2} = I - \overline{D}^{-1/2} W \overline{D}^{-1/2}.$$

Kunegis et al. (2010) showed that $\overline{L}$ is positive semidefinite. For a graph without isolated vertices, we have $\overline{d}(v_i) > 0$ for $i = 1, \ldots, m$, so $\overline{D}^{-1/2}$ is well defined.

Given a partition of V into K clusters $(A_1, \ldots, A_K)$, if we represent the jth block of this partition by a vector X^j such that

$$X_i^j = \begin{cases} a_j & \text{if } v_i \in A_j \\ 0 & \text{if } v_i \notin A_j, \end{cases}$$

for some $a_j \neq 0$. For illustration, suppose $m = 5$ and $A_1 = \{v_1, v_3\}$ then $(X^1)^\top = [a_1, 0, a_1, 0, 0]$.

Definition 2.2. The *signed normalized cut* $\mathrm{sNcut}(A_1, \ldots, A_K)$ of the partition $(A_1, ..., A_K)$ is defined as

$$\mathrm{sNcut}(A_1, \ldots, A_K) = \sum_{j=1}^{K} \frac{\mathrm{cut}(A_j, A_j^C) + 2\mathrm{links}^-(A_j, A_j)}{\mathrm{vol}(A_j)}.$$

It should be noted that this formulation differs significantly from Kunegis et al. (2010) and even more so from must-link / cannot-link clustering.

Observe that minimizing $\mathrm{sNcut}(A_1, \ldots, A_K)$ minimizes the number of positive and negative edges between clusters and also the number of negative edges within clusters. Removing the term $\mathrm{links}^-(A_j, A_j)$ reduces sNcut to normalized cuts.

A linear algebraic formulation is

$$\mathrm{sNcut}(A_1, \ldots, A_K) = \sum_{j=1}^{K} \frac{(X^j)^\top \overline{L} X^j}{(X^j)^\top \overline{D} X^j}.$$

where X is the $N \times K$ matrix whose jth column is X^j.

2.2 Optimization Problem

We now formulate K-way clustering of a graph using normalized cuts.

If we let

$$\mathcal{X} = \Big\{ [X^1 \ \ldots \ X^K] \mid X^j = a_j(x_1^j, \ldots, x_N^j), \\ x_i^j \in \{1, 0\}, a_j \in \mathbb{R},\ X^j \neq 0 \Big\}$$

our solution set is

$$\mathcal{K} = \Big\{ X \in \mathcal{X} \mid (X^i)^\top \overline{D} X^j = 0, \\ 1 \leq i, j \leq K, \quad i \neq j \Big\}.$$

The resulting optimization problem is

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{K} \frac{(X^j)^\top \overline{L} X^j}{(X^j)^\top \overline{D} X^j} \\ \text{subject to} \quad & (X^i)^\top \overline{D} X^j = 0, \\ & 1 \leq i, j \leq K,\ i \neq j, \quad X \in \mathcal{X}. \end{aligned}$$

The problem can be reformulated to an equivalent optimization problem:

$$\begin{aligned} \text{minimize} \quad & \mathrm{tr}(X^\top \overline{L} X) \\ \text{subject to} \quad & X^\top \overline{D} X = I, \quad X \in \mathcal{X}. \end{aligned}$$

We then form a relaxation of the above problem, dropping the condition that $X \in \mathcal{X}$, giving

Relaxed Problem

$$\begin{aligned} \text{minimize} \quad & \mathrm{tr}(Y^\top \overline{D}^{-1/2} \overline{L} \overline{D}^{-1/2} Y) \\ \text{subject to} \quad & Y^\top Y = I. \end{aligned}$$

The minimum of the relaxed problem is achieved by the K unit eigenvectors associated with the smallest eigenvalues of L_{sym}.

2.3 Finding an Approximate Discrete Solution

Given a solution Z of the relaxed problem, we look for pairs (X, Q) with $X \in \mathcal{X}$ and where Q is a $K \times K$ matrix with nonzero and pairwise orthogonal columns, with $\|X\|_F = \|Z\|_F$, that minimize

$$\varphi(X, Q) = \|X - ZQ\|_F .$$

Here, $\|A\|_F$ is the Frobenius norm of A.

This nonlinear optimization problem involves two unknown matrices X and Q. To solve the relaxed problem, we proceed by alternating between minimizing $\varphi(X, Q) = \|X - ZQ\|_F$ with respect to X holding Q fixed (step 5 in algorithm 1), and minimizing $\varphi(X, Q)$ with respect to Q holding X fixed (steps 6 and 7 in algorithm 1).

This second stage in which X is held fixed has been studied, but it is still a hard problem for which no closed-form solution is known. Hence we divide the problem into steps 6 and 7 for which the solution is known. Since Q is of the form $Q = R\Lambda$ where $R \in \mathbf{O}(K)$ and Λ is a diagonal invertible matrix, we minimize $\|X - ZR\Lambda\|_F$. The matrix $R\Lambda$ is not a minimizer of $\|X - ZR\Lambda\|_F$ in general, but it is an improvement on R alone, and both stages can be solved quite easily. In step 6 the problem reduces to minimizing $-2\mathrm{tr}(Q^\top Z^\top X)$; that is, maximizing $\mathrm{tr}(Q^\top Z^\top X)$.

Algorithm 1 Signed Clustering

1: **Input:** W the weight matrix (without isolated nodes), K the number of clusters, and termination threshold ϵ.
2: Using the $\overline{D}$ the degree matrix, and the signed Laplacian $\overline{L}$, compute $\overline{L}_{sym}$ the signed normalized Laplacian.
3: Initialize $\Lambda = I$, $X = \overline{D}^{-\frac{1}{2}} U$ where U is the matrix of the eigenvectors corresponding to the K smallest eigenvalues of $\overline{L}_{sym}$. [3]
4: **while** $\|X - ZR\Lambda\|_F > \epsilon$ **do**
5: Minimize $\|X - ZR\Lambda\|_F$ with respect to X holding Q fixed.
6: Fix X, Z, and Λ, find $R \in \mathbf{O}(K)$ that minimizes $\|X - ZR\Lambda\|_F$.
7: Fix X, Z, and R, find a diagonal invertible matrix Λ that minimizes $\|X - ZR\Lambda\|_F$.
8: **end while**
9: Find the discrete solution X^* by choosing the largest entry x_{ij} on row i set $x_{ij} = 1$ and all other $x_{ij} = 0$ for row i.
10: **Output:** X^*.

Steps 3 through 10 may be replaced by standard K-means clustering. It should also be noted that by

removing the solution requirement that $X^j \neq 0$, the algorithm can find $k \leq K$ clusters.

3 Similarity Calculation

The main input to the spectral signed clustering algorithm is the similarity matrix W, which overlays both the distributional properties and thesaurus information. Following Belkin and Niyogi (2003), we chose the heat kernel based on the Euclidean distance between word vector representations as our similarity metric, such that

$$W_{ij} = \begin{cases} 0 & \text{if } e^{-\frac{\|w_i - w_j\|^2}{\sigma}} < \epsilon \\ e^{-\frac{\|w_i - w_j\|^2}{\sigma}} & \text{otherwise} \end{cases}.$$

where σ and ϵ are hyperparameters found using grid search (see Supplemental material for more detail).

We represented the thesaurus as two matrices where

$$T^{syn}_{ij} = \begin{cases} 1 & \text{if words } i \text{ and } j \text{ are synonyms} \\ 0 & \text{otherwise} \end{cases}.$$

and

$$T^{ant}_{ij} = \begin{cases} -1 & \text{if words } i \text{ and } j \text{ are antonyms} \\ 0 & \text{otherwise} \end{cases}.$$

T^{syn} is the synonym graph and T^{ant} is the antonym graph. The signed graph can then be written in matrix form as $\hat{W} = \gamma W + \beta^{ant} T^{ant} \odot W + \beta^{syn} T^{syn} \odot W$, where $\odot$ computes Hadamard product (element-wise multiplication).

The parameters γ, β^{syn}, and β^{ant} are tuned to the data target dataset using cross validation. The reader should note that σ and ϵ are not found using a target dataset, but instead using cross validation and grid search to minimize the number of negative edges within clusters and the number of disconnected components in the cluster.

4 Evaluation Metrics

We evaluated the clusters using both intrinsic and extrinsic methods. For intrinsic evaluation, we used thesaurus information for two novel metrics: 1) the number of negative edges (NNE) within the clusters, which in our semantic clusters is the number of antonyms in the same cluster, and 2) the number of disconnected components (NDC) in the synonym graph, so the number of groups of words that are not connected by a synonym relation in the thesaurus. The NDC thus has the disadvantage that it is a function of the thesaurus coverage. Our third intrinsic measure uses a gold standard designed to measure how well we capture word similarity: Semantically similar words should be in the same cluster and semantically dissimilar words should not. For extrinsic evaluation, as descibed below, we measure how much our clusters help to identify text polarity. We also compare multiple word embeddings and thesauri to demonstrate the stability of our method.

5 Experiments with Synthetic Data

In order to evaluate our signed graph clustering method, we first focused on intrinsic measures of cluster quality in synthetic data. To do so, we created random signed graphs with the same proportion of positive and negative edges as in our real dataset. Figure 2 demonstrates that the number of negative edges within a cluster is minimized using our clustering algorithm on simulated data. As the number of clusters becomes large, the number of disconnected components, which includes clusters of size one, consistently increases. Determining the optimal cluster size and similarity parameters requires making a trade off between NDC and NNE. For example, in figure 2 the optimal cluster size is 20. One can see that as the number of clusters increases NNE goes to zero, but the number of disconnected components becomes the number of vertices. In the extreme case all clusters contain one vertex. K-means, also shown in figure 2, does not optimize NNE.

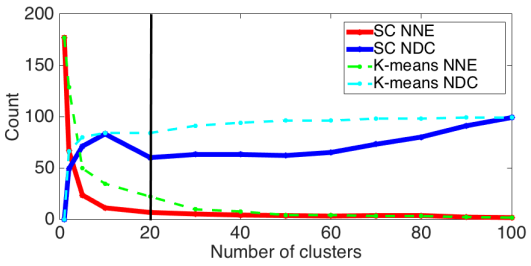


Figure 2: The relation between disconnected component (NDC) and negative edge (NNE) using simulated signed graphs with 100 vertices.

6 Experimental Setup

6.1 Word Embeddings

We used four different word embedding methods for evaluation: Skip-gram vectors (word2vec) (Mikolov et al., 2013), Global vectors (GloVe) (Pennington et al., 2014), Eigenwords (Dhillon et al., 2015), and Global Context (GloCon) (Huang et al., 2012); however, we only report the results for word2vec, which is the most popular word embedding (see the supplemental material for other embeddings). We used word2vec 300 dimensional embeddings which were trained on several billion words of English: the Gigaword and the English discussion forum data gathered as part of BOLT. Tokenization was performed using CMU's Twokenize.[4]

6.2 Thesauri

Several thesauri were used in order to test the robustness including Roget's Thesaurus (Roget, 1852), the Microsoft Word English (MS Word) thesaurus from Samsonovic et al. (2010) and WordNet 3.0 (Miller, 1995).

We chose a subset of 5108 words for the training dataset, which had high overlap between various sources. Changes to the training dataset had minimal effects on the optimal parameters. Within the training dataset, each of the thesauri had roughly 3700 antonym pairs; combined they had 6680. However, the number of distinct connected components varied, with Roget's Thesaurus having the fewest (629), and MS Word Thesaurus (1162) and WordNet (2449) having the most. These ratios were consistent across the full dataset.

6.3 Gold Standard SimLex-999 And SimVerb-3500

Following the analysis of Vlachos et al. (2009), we threshold the semantically similar datasets to find word pairs which should or should not belong to the same cluster. As ground truth, we extracted 120 semantically similar words from SimLex-999 with a similarity score greater than 8 out of 10. SimLex-999 is a gold standard resource for semantic similarity, not relatedness, based on ratings by human annotators.

Our 120 pair subset of SimLex-999 has multiple parts-of-speech including Noun-Noun pairs, Verb-Verb pairs and Adjective-Adjective pairs. Within SimVerb-3500, we used a subset of 318 semantically similar verb pairs.

The community is attempting to define better gold standards; however, currently these are the best datasets that we are aware of. We tried to use WordNet, Roget, and the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) as a gold standard, but manual inspection as well as empirical results showed that none of the automatically generated datasets were a sufficient gold standard. Possibly the symmetric pattern of (Schwartz et al., 2015) would have been sufficient; we did not have time to validate this.

6.4 Stanford Sentiment Treebank

We also evaluated our clusters by using them as features for predicting sentiment, using sentiment treebank [5] (Socher et al., 2013) with coarse-grained labels on phrases and sentences from movie review excerpts. This dataset is widely used for the evaluation of sentiment analysis. We used the standard partition of the treebank into training (6920), development (872), and test (1821) sets.

7 Cluster Evaluation

Table 1 shows the four most-associated words with "accept" using different methods.

We now turn to quantitative measures of word similarity and synonym cluster quality.

7.1 Comparison with K-means and Normalized Cuts

In order to assess the model we tested (1) K-means, (2) normalized cuts without thesaurus, and (3) signed normalized cuts. As a baseline, we created clusters using K-means on the original word2vec vector representations where the number of K clusters was set to 750.

Table 2 shows the relative ratios of the different clustering methods of with respect to antonym pair inclusion and the number of disconnected components within the clusters. For both methods, over twenty percent of the clusters contain antonym pairs even though the median cluster size is six. Signed clustering radically reduced the number of antonyms within clusters compared to the other methods.

[4] https://github.com/brendano/ark-tweet-nlp

[5] http://nlp.stanford.edu/sentiment/treebank.html

Ref word	Roget	WordNet	MS Word	W2V	SC W2V
accept	adopt	agree	take	accepts	grant
	accept your fate	get	swallow	reject	permit
	be fooled by	fancy	consent	agree	let
	acquiesce	hold	assume	accepting	okay

Table 1: Qualitative comparison of clusters.

Method	Antonym Ratio	DC Ratio
K-Means	0.24	0.95
NC	0.21	0.97
SC	0.06	0.49

Table 2: Clustering evaluation of K-means, normalized cuts, and signed normalized cuts with 750 clusters. Ratio of clusters with containing one or more antonym pair and ratio of clusters with disconnected components.

Method	Acc SimLex	Err
MSW Lookup	0.70	0
Roget Lookup	0.63	0
WordNet Lookup	0.43	0
Combined Lookup	0.90	0
NC(W2V)	0.36	0.05
SC (W2V)	**0.67**	0
Lookup + NC(W2V)	0.91	0.05
Lookup + **SC**(W2V)	**0.96**	0
MSW + **SC**(W2V)	0.95	0

Table 3: Clustering evaluation using SimLex-999 with 120 word pairs having similarity score over 8. SC stands for our signed clustering and NC is standard normalized cuts. **SC**(W2V) are the word clusters from signed clustering using word2vec and the combined thesauri. **Err** is the proportion of dissimilar words (with score < 2) present in the same cluster.

8 Empirical Results

Tables 3 and 5 present our main result. When using our signed clustering method with similar words, as labeled by SimLex-999 and SimVerb-3500, our clustering accuracy increased by 5% on both SimLex-999 and SimVerb-3000. Furthermore, by combining the thesauri lookup with our clustering, we achieved almost perfect accuracy (96%). Table 5 shows the sentiment analysis task performance. Our method outperforms all methods with similar complexity; however, we did not reach state-of-the-art results when compared to much more complex models which also use a richer dataset.

8.1 Evaluation Using Word Similarity Datasets

In a perfect setting, all word pairs rated highly similar by human annotators would be in the same cluster, and all words which were rated dissimilar would be in different clusters. Since our clustering algorithm produced sets of words, we used this evaluation instead of the more commonly reported correlations.

In table 3 we show the results of the evaluation with SimLex-999. Combining thesaurus lookup and word2vec+CombThes clusters, labeled as Lookup + **SC**(W2V), yielded an accuracy of 0.96 (5 errors). Note that clusters using word2vec with normalized cuts does not improve accuracy. The MSW thesaurus has much lower coverage, but 100 % accuracy, which is why when combined with the signed clustering the performance is 0.95. In table 3 we state the proportion of clusters containing dissimilar words as a sanity check for cluster size. (See supplemental material for full cluster size optimization information.) Another important result is that the verb accuracy yielded the largest accuracy gains, consistent with the results of Schwartz et al. (2015).

Table 4 clearly shows that the overall performance of all methods is lower for verb similarity. However, the improvement using both signed clustering as well as thesaurus look is also larger.

8.2 Sentiment Analysis

We trained an l_2-norm regularized logistic regression (Friedman et al., 2001) and simultaneously γ, β^{syn}, and β^{ant} using our word clusters in order to predict the coarse-grained sentiment at the sentence level. The γ and β parameters were found using a portion of the data where we iteratively switch between the logistic regression and the parameters, holding each fixed. However, hyperparameters σ and ϵ, and the number of clusters

Method	Acc SimVerb
MSW Lookup	0.45
Roget Lookup	0.59
WordNet Lookup	0.43
Combined Lookup	0.83
NC(W2V)	0.24
SC (W2V)	**0.56**
Lookup + NC(W2V)	0.83
Lookup + **SC**(W2V)	**0.88**

Table 4: Clustering evaluation using SimVerb-3500 with 317 word pairs having similarity score over 8. SC stands for our signed clustering and NC is standard normalized cuts. **SC**(W2V) are the word clusters from signed clustering using word2vec and the combined thesauri.

Model	Accuracy
NB (Socher et al., 2013)	0.818
VecAvg (W2V) (Faruqui et al., 2015)	0.812
RVecAvg (W2V) (Faruqui et al., 2015)	0.821
RNN(Socher et al., 2013)	0.824
NC(W2V)	0.79
SC(Thes)	0.752
SC(W2V)	**0.836**

Table 5: Sentiment analysis accuracy for binary predictions of signed clustering algorithm (SC) versus other models. **SC**(W2V) are the signed clusters using word2vec word representations.

K were optimized minimizing error using grid search. We compared our model against existing models: Naive Bayes with bag of words (NB) (Socher et al., 2013), sentence word embedding averages (VecAvg), retrofitted sentence word embeddings (RVecAvg) (Faruqui et al., 2015) that incorporate thesaurus information, simple recurrent neural networks (RNN), and two baselines of normalized cuts and signed normalized cuts using only thesaurus information.

While the state-of-the art Convolutional Neural Network (CNN) (Kim, 2014) is at 0.881, our model performs quite well with much less information and complexity. Table 5 shows that signed clustering outperforms the baselines of Naive Bayes, normalized cuts, and signed cuts using just thesaurus information. Furthermore, we outperform comparable models, including retrofitting, which has thesaurus information, and the recurrent neural network, which has access to domain specific context information.

Signed clustering using only thesaurus information (**SC(Thes)**) performed significantly worse than all other methods. This was largely due to low coverage; rare words such as "WOW" and "???" are not covered. As expected, because normalized cut clusters include antonyms, the method performs worse than others. Nonetheless the improvement from 0.79 to 0.836 is quite drastic.

9 Conclusion

We developed a novel theory for signed normalized cuts and an algorithm for finding their discrete solution. We showed that we can find superior semantically similar clusters which do not require new word embeddings but simply overlay thesaurus information on preexisting ones. The clusters are general and can be used with many out-of-the-box word embeddings. By accounting for antonym relationships, our algorithm greatly outperforms simple normalized cuts. Finally, we examined our clustering method on the sentiment analysis task from Socher et al. (2013) sentiment treebank dataset and showed that it improved performance versus comparable models.

Our automatically generated clusters give better coverage than manually constructed thesauri. Our signed spectral clustering method allows us to incorporate the knowledge contained in these thesauri without modifying the word embeddings themselves. We further showed that use of the thesauri can be tuned to the task at hand.

Our signed spectral clustering method could be applied to a broad range of NLP tasks, such as prediction of social group clustering, identification of personal versus non-personal verbs, and analyses of clusters which capture positive, negative, and objective emotional content. It could also be used to explore multi-view relationships, such as aligning synonym clusters across multiple languages. Another possibility is to use thesauri and word vector representations together with word sense disambiguation to generate semantically similar clusters for multiple senses of words. Furthermore, signed spectral clustering has broader applications such as cellular biology, social networking, and electricity networks. Finally, we plan to extend the hard signed clustering presented here to probabilistic soft clustering.

References

Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation* 15(6):1373–1396.

Dorwin Cartwright and Frank Harary. 1956. Structural balance: a generalization of heider's theory. *Psychological review* 63(5):277.

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1602–1612. http://aclweb.org/anthology/D13-1167.

Kai-Yang Chiang, Cho-Jui Hsieh, Nagarajan Natarajan, Inderjit S. Dhillon, and Ambuj Tewari. 2014. Prediction and clustering in signed networks: A local to global perspective. *Journal of Machine Learning Research* 15:1177–1213. http://jmlr.org/papers/v15/chiang14a.html.

James Richard Curran. 2004. From distributional to semantic similarity .

Paramveer S. Dhillon, Dean P. Foster, and Lyle H. Ungar. 2015. Eigenwords: Spectral word embeddings. *Journal of Machine Learning Research* 16:3035–3078. http://jmlr.org/papers/v16/dhillon15a.html.

Manaal Faruqui, Jesse Dodge, Kumar Sujay Jauhar, Chris Dyer, Eduard Hovy, and A. Noah Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1606–1615. https://doi.org/10.3115/v1/N15-1184.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin.

Jean Gallier. 2016. Spectral theory of unsigned and signed graphs applications to graph clustering: a survey. *arXiv preprint arXiv:1601.04692* .

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 758–764. http://aclweb.org/anthology/N13-1092.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869* .

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53(2):217–288.

Frank Harary. 1953. On the notion of balance of a signed graph. *The Michigan Mathematical Journal* 2(2):143–146.

Zellig S Harris. 1954. Distributional structure. *Word* .

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *arXiv preprint arXiv:1408.3456* .

Yao Ping Hou. 2005. Bounds for the least laplacian eigenvalue of a signed graph. *Acta Mathematica Sinica* 21(4):955–960.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 873–882. http://aclweb.org/anthology/P12-1092.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1746–1751. https://doi.org/10.3115/v1/D14-1181.

Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F O'Brien. 2004. Spectral surface reconstruction from noisy point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM, pages 11–21.

Jérôme Kunegis, Stephan Schmidt, Andreas Lommatzsch, Jürgen Lerner, Ernesto William De Luca, and Sahin Albayrak. 2010. Spectral analysis of signed graphs for clustering, prediction and visualization. In *SDM*. SIAM, volume 10, pages 559–559.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*. http://aclweb.org/anthology/P98-2127.

Pedro Mercado, Francesco Tudisco, and Matthias Hein. 2016. Clustering signed networks with the geometric mean of laplacians. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., pages 4421–4429. http://papers.nips.cc/paper/6164-clustering-signed-networks-with-the-geometric-mean-of-laplacians.pdf.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

M. Saif Mohammad, J. Bonnie Dorr, Graeme Hirst, and D. Peter Turney. 2013. Computing lexical contrast. *Computational Linguistics* 39(3). https://doi.org/10.1162/COLI_a_00143.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, M. Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 142–148. https://doi.org/10.18653/v1/N16-1018.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 984–989. https://doi.org/10.3115/v1/N15-1100.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1532–1543. https://doi.org/10.3115/v1/D14-1162.

The Nghia Pham, Angeliki Lazaridou, and Marco Baroni. 2015. A multitask objective to inject lexical contrast into distributional semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 21–26. https://doi.org/10.3115/v1/P15-2004.

Syama Sundar Rangapuram and Matthias Hein. 2012. Constrained 1-spectral clustering. *International conference on Artificial Intelligence and Statistics (AISTATS)* 22:1143—1151.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. Association for Computational Linguistics, pages 675–682. http://aclweb.org/anthology/E09-1077.

Peter Mark Roget. 1852. *Roget's Thesaurus of English Words and Phrases....* Longman Group Ltd.

Alexei V Samsonovic, Giorgio A Ascoli, and Jeffrey Krichmar. 2010. Principal semantic components of language and the measurement of meaning. *PloS one* 5(6):e10921.

Silke Scheible, Sabine Schulte im Walde, and Sylvia Springorum. 2013. Uncovering distributional differences between synonyms and antonyms in a word space model. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, pages 489–497. http://aclweb.org/anthology/I13-1056.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 258–267. https://doi.org/10.18653/v1/K15-1026.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22(8):888–905.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, D. Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1631–1642. http://aclweb.org/anthology/D13-1170.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1555–1565. https://doi.org/10.3115/v1/P14-1146.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37(1):141–188.

Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. 2009. *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, Association for Computational Linguistics, chapter Unsupervised and Constrained Dirichlet Process Mixture Models for Verb Clustering, pages 74–82. http://aclweb.org/anthology/W09-0210.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016. Community-based weighted graph model for valence-arousal prediction of affective words. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24(11):1957–1968.

Wen-tau Yih, Geoffrey Zweig, and John Platt. 2012. Polarity inducing latent semantic analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 1212–1222. http://aclweb.org/anthology/D12-1111.

Stella X Yu and Jianbo Shi. 2003. Multiclass spectral clustering. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, pages 313–319.

Ron Zass and Amnon Shashua. 2005. A unifying approach to hard and probabilistic clustering. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. IEEE, volume 1, pages 294–301.

Jingwei Zhang, Jeremy Salwen, Michael Glass, and Alfio Gliozzo. 2014. Word semantic representations using bayesian probabilistic tensor factorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1522–1531. https://doi.org/10.3115/v1/D14-1161.

An Interpretable Knowledge Transfer Model for Knowledge Base Completion

Qizhe Xie, Xuezhe Ma, Zihang Dai, Eduard Hovy
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{qzxie, xuezhem, dzihang, hovy}@cs.cmu.edu

Abstract

Knowledge bases are important resources for a variety of natural language processing tasks but suffer from incompleteness. We propose a novel embedding model, *ITransF*, to perform knowledge base completion. Equipped with a sparse attention mechanism, ITransF discovers hidden concepts of relations and transfer statistical strength through the sharing of concepts. Moreover, the learned associations between relations and concepts, which are represented by sparse attention vectors, can be interpreted easily. We evaluate ITransF on two benchmark datasets—WN18 and FB15k for knowledge base completion and obtains improvements on both the mean rank and Hits@10 metrics, over all baselines that do not use additional information.

1 Introduction

Knowledge bases (KB), such as WordNet (Fellbaum, 1998), Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and DBpedia (Lehmann et al., 2015), are useful resources for many applications such as question answering (Berant et al., 2013; Yih et al., 2015; Dai et al., 2016) and information extraction (Mintz et al., 2009). However, knowledge bases suffer from incompleteness despite their formidable sizes (Socher et al., 2013; West et al., 2014), leading to a number of studies on automatic knowledge base completion (KBC) (Nickel et al., 2015) or link prediction.

The fundamental motivation behind these studies is that there exist some statistical regularities under the intertwined facts stored in the multi-relational knowledge base. By discovering generalizable regularities in known facts, missing ones may be recovered in a faithful way. Due to its excellent generalization capability, distributed representations, a.k.a. embeddings, have been popularized to address the KBC task (Nickel et al., 2011; Bordes et al., 2011, 2014, 2013; Socher et al., 2013; Wang et al., 2014; Guu et al., 2015; Nguyen et al., 2016b).

As a seminal work, Bordes et al. (2013) proposes the TransE, which models the statistical regularities with linear translations between entity embeddings operated by a relation embedding. Implicitly, TransE assumes both entity embeddings and relation embeddings dwell in the same vector space, posing an unnecessarily strong prior. To relax this requirement, a variety of models first project the entity embeddings to a relation-dependent space (Bordes et al., 2014; Ji et al., 2015; Lin et al., 2015b; Nguyen et al., 2016b), and then model the translation property in the projected space. Typically, these relation-dependent spaces are characterized by the projection matrices unique to each relation. As a benefit, different aspects of the same entity can be temporarily emphasized or depressed as an effect of the projection. For instance, STransE (Nguyen et al., 2016b) utilizes two projection matrices per relation, one for the head entity and the other for the tail entity.

Despite the superior performance of STransE compared to TransE, it is more prone to the data sparsity problem. Concretely, since the projection spaces are unique to each relation, projection matrices associated with rare relations can only be exposed to very few facts during training, resulting in poor generalization. For common relations, a similar issue exists. Without any restrictions on the number of projection matrices, logically related or conceptually similar relations may have distinct projection spaces, hindering the discovery, sharing, and generalization of statistical regularities.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 950–962
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1088

Previously, a line of research makes use of external information such as textual relations from web-scale corpus or node features (Toutanova et al., 2015; Toutanova and Chen, 2015; Nguyen et al., 2016a), alleviating the sparsity problem. In parallel, recent work has proposed to model regularities beyond local facts by considering multi-relation paths (García-Durán et al., 2015; Lin et al., 2015a; Shen et al., 2016). Since the number of paths grows exponentially with its length, as a side effect, path-based models enjoy much more training cases, suffering less from the problem.

In this paper, we propose an interpretable knowledge transfer model (ITransF), which encourages the sharing of statistic regularities between the projection matrices of relations and alleviates the data sparsity problem. At the core of ITransF is a sparse attention mechanism, which learns to compose shared concept matrices into relation-specific projection matrices, leading to a better generalization property. Without any external resources, ITransF improves mean rank and Hits@10 on two benchmark datasets, over all previous approaches of the same kind. In addition, the parameter sharing is clearly indicated by the learned sparse attention vectors, enabling us to interpret how knowledge transfer is carried out. To induce the desired sparsity during optimization, we further introduce a block iterative optimization algorithm.

In summary, the contributions of this work are: (i) proposing a novel knowledge embedding model which enables knowledge transfer by learning to discover shared regularities; (ii) introducing a learning algorithm to directly optimize a sparse representation from which the knowledge transferring procedure is interpretable; (iii) showing the effectiveness of our model by outperforming baselines on two benchmark datasets for knowledge base completion task.

2 Notation and Previous Models

Let E denote the set of entities and R denote the set of relations. In knowledge base completion, given a training set P of triples (h, r, t) where $h, t \in E$ are the head and tail entities having a relation $r \in R$, e.g., (*Steve Jobs*, `FounderOf`, *Apple*), we want to predict missing facts such as (*Steve Jobs*, `Profession`, *Businessperson*).

Most of the embedding models for knowledge base completion define an energy function $f_r(h, t)$ according to the fact's plausibility (Bordes et al., 2011, 2014, 2013; Socher et al., 2013; Wang et al., 2014; Yang et al., 2015; Guu et al., 2015; Nguyen et al., 2016b). The models are learned to minimize energy $f_r(h, t)$ of a plausible triple (h, r, t) and to maximize energy $f_r(h', t')$ of an implausible triple (h', r, t').

Motivated by the linear translation phenomenon observed in well trained word embeddings (Mikolov et al., 2013), TransE (Bordes et al., 2013) represents the head entity h, the relation r and the tail entity t with vectors $\mathbf{h}, \mathbf{r}$ and $\mathbf{t} \in \mathbb{R}^n$ respectively, which were trained so that $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. They define the energy function as

$$f_r(h, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_\ell$$

where $\ell = 1$ or 2, which means either the ℓ_1 or the ℓ_2 norm of the vector $\mathbf{h} + \mathbf{r} - \mathbf{t}$ will be used depending on the performance on the validation set.

To better model relation-specific aspects of the same entity, TransR (Lin et al., 2015b) uses projection matrices and projects the head entity and the tail entity to a relation-dependent space. STransE (Nguyen et al., 2016b) extends TransR by employing different matrices for mapping the head and the tail entity. The energy function is

$$f_r(h, t) = \|\mathbf{W}_{r,1}\mathbf{h} + \mathbf{r} - \mathbf{W}_{r,2}\mathbf{t}\|_\ell$$

However, not all relations have abundant data to estimate the relation specific matrices as most of the training samples are associated with only a few relations, leading to the data sparsity problem for rare relations.

3 Interpretable Knowledge Transfer

3.1 Model

As discussed above, a fundamental weakness in TransR and STransE is that they equip each relation with a set of unique projection matrices, which not only introduces more parameters but also hinders knowledge sharing. Intuitively, many relations share some concepts with each other, although they are stored as independent symbols in KB. For example, the relation "(somebody) won award for (some work)" and "(somebody) was nominated for (some work)" both describe a person's high-quality work which wins an award or a nomination respectively. This phenomenon suggests that one relation actually represents a collection of real-world concepts, and one concept

can be shared by several relations. Inspired by the existence of such lower-level concepts, instead of defining a unique set of projection matrices for every relation, we can alternatively define a small set of concept projection matrices and then compose them into customized projection matrices. Effectively, the relation-dependent translation space is then reduced to the smaller concept spaces.

However, in general, we do not have prior knowledge about what concepts exist out there and how they are composed to form relations. Therefore, in ITransF, we propose to learn this information simultaneously from data, together with all knowledge embeddings. Following this idea, we first present the model details, then discuss the optimization techniques for training.

Energy function Specifically, we stack all the concept projection matrices to a 3-dimensional tensor $\mathbf{D} \in \mathbb{R}^{m \times n \times n}$, where m is the pre-specified number of concept projection matrices and n is the dimensionality of entity embeddings and relation embeddings. We let each relation select the most useful projection matrices from the tensor, where the selection is represented by an attention vector. The energy function of ITransF is defined as:

$$f_r(h,t) = \|\boldsymbol{\alpha}_r^H \cdot \mathbf{D} \cdot \mathbf{h} + \mathbf{r} - \boldsymbol{\alpha}_r^T \cdot \mathbf{D} \cdot \mathbf{t}\|_\ell \quad (1)$$

where $\boldsymbol{\alpha}_r^H, \boldsymbol{\alpha}_r^T \in [0,1]^m$, satisfying $\sum_i \boldsymbol{\alpha}_{r,i}^H = \sum_i \boldsymbol{\alpha}_{r,i}^T = 1$, are normalized attention vectors used to compose all concept projection matrices in $\mathbf{D}$ by a convex combination. It is obvious that STransE can be expressed as a special case of our model when we use $m = 2|R|$ concept matrices and set attention vectors to disjoint one-hot vectors. Hence our model space is a generalization of STransE. Note that we can safely use fewer concept matrices in ITransF and obtain better performance (see section 4.3), though STransE always requires $2|R|$ projection matrices.

We follow previous work to minimize the following hinge loss function:

$$\mathcal{L} = \sum_{\substack{(h,r,t)\sim P,\\ (h',r,t')\sim N}} \left[\gamma + f_r(h,t) - f_r(h',t')\right]_+ \quad (2)$$

where P is the training set consisting of correct triples, N is the distribution of corrupted triples defined in section 3.3, and $[\cdot]_+ = \max(\cdot, 0)$. Note that we have omitted the dependence of N on (h,r,t) to avoid clutter. We normalize the entity vectors $\mathbf{h}, \mathbf{t}$, and the projected entity vectors $\boldsymbol{\alpha}_r^H \cdot \mathbf{D} \cdot \mathbf{h}$ and $\boldsymbol{\alpha}_r^T \cdot \mathbf{D} \cdot \mathbf{t}$ to have unit length after each update, which is an effective regularization method that benefits all models.

Sparse attention vectors In Eq. (1), we have defined $\boldsymbol{\alpha}_r^H, \boldsymbol{\alpha}_r^T$ to be some normalized vectors used for composition. With a dense attention vector, it is computationally expensive to perform the convex combination of m matrices in each iteration. Moreover, a relation usually does not consist of all existing concepts in practice. Furthermore, when the attention vectors are sparse, it is often easier to interpret their behaviors and understand how concepts are shared by different relations.

Motivated by these potential benefits, we further hope to learn sparse attention vectors in ITransF. However, directly posing ℓ_1 regularization (Tibshirani, 1996) on the attention vectors fails to produce sparse representations in our preliminary experiment, which motivates us to enforce ℓ_0 constraints on $\boldsymbol{\alpha}_r^T, \boldsymbol{\alpha}_r^H$.

In order to satisfy both the normalization condition and the ℓ_0 constraints, we reparameterize the attention vectors in the following way:

$$\boldsymbol{\alpha}_r^H = \mathrm{SparseSoftmax}(\mathbf{v}_r^H, \mathbf{I}_r^H)$$
$$\boldsymbol{\alpha}_r^T = \mathrm{SparseSoftmax}(\mathbf{v}_r^T, \mathbf{I}_r^T)$$

where $\mathbf{v}_r^H, \mathbf{v}_r^T \in \mathbb{R}^m$ are the pre-softmax scores, $\mathbf{I}_r^H, \mathbf{I}_r^T \in \{0,1\}^m$ are the sparse assignment vectors, indicating the non-zero entries of attention vectors, and the $\mathrm{SparseSoftmax}$ is defined as

$$\mathrm{SparseSoftmax}(\mathbf{v}, \mathbf{I})_i = \frac{\exp(\mathbf{v}_i/\tau)\mathbf{I}_i}{\sum_j \exp(\mathbf{v}_j/\tau)\mathbf{I}_j}$$

with τ being the temperature of Softmax.

With this reparameterization, $\mathbf{v}_r^H, \mathbf{v}_r^T$ and $\mathbf{I}_r^H, \mathbf{I}_r^T$ replace $\boldsymbol{\alpha}_r^T, \boldsymbol{\alpha}_r^H$ to become the real parameters of the model. Also, note that it is equivalent to pose the ℓ_0 constraints on $\mathbf{I}_r^H, \mathbf{I}_r^T$ instead of $\boldsymbol{\alpha}_r^T, \boldsymbol{\alpha}_r^H$. Putting these modifications together, we can rewrite the optimization problem as

$$\begin{aligned} &\text{minimize} && \mathcal{L} \\ &\text{subject to} && \|\mathbf{I}_r^H\|_0 \le k, \|\mathbf{I}_r^T\|_0 \le k \end{aligned} \quad (3)$$

where $\mathcal{L}$ is the loss function defined in Eq. (2).

3.2 Block Iterative Optimization

Though sparseness is favorable in practice, it is generally NP-hard to find the optimal solution under ℓ_0 constraints. Thus, we resort to an approximated algorithm in this work.

For convenience, we refer to the parameters with and without the sparse constraints as the *sparse* partition and the *dense* partition, respectively. Based on this notion, the high-level idea of the approximated algorithm is to iteratively optimize one of the two partitions while holding the other one fixed. Since all parameters in the dense partition, including the embeddings, the projection matrices, and the pre-softmax scores, are fully differentiable with the sparse partition fixed, we can simply utilize SGD to optimize the dense partition. Then, the core difficulty lies in the step of optimizing the sparse partition (i.e. the sparse assignment vectors), during which we want the following two properties to hold

1. the sparsity required by the ℓ_0 constaint is maintained, and
2. the cost define by Eq. (2) is decreased.

Satisfying the two criterion seems to highly resemble the original problem defined in Eq. (3). However, the dramatic difference here is that with parameters in the dense partition regarded as constant, the cost function is decoupled w.r.t. each relation r. In other words, the optimal choice of $\mathbf{I}_r^H, \mathbf{I}_r^T$ is independent of $\mathbf{I}_{r'}^H, \mathbf{I}_{r'}^T$ for any $r' \neq r$. Therefore, we only need to consider the optimization for a single relation r, which is essentially an assignment problem. Note that, however, $\mathbf{I}_r^H$ and $\mathbf{I}_r^T$ are still coupled, without which we basically reach the situation in a backpack problem. In principle, one can explore combinatorial optimization techniques to optimize $\mathbf{I}_{r'}^H, \mathbf{I}_{r'}^T$ jointly, which usually involve some iterative procedure. To avoid adding another inner loop to our algorithm, we turn to a simple but fast approximation method based on the following single-matrix cost.

Specifically, for each relation r, we consider the induced cost $\mathcal{L}_{r,i}^H$ where only a single projection matrix i is used for the head entity:

$$\mathcal{L}_{r,i}^H = \sum_{\substack{(h,r,t)\sim P_r,\\ (h',r,t')\sim N_r}} \left[\gamma + f_{r,i}^H(h,t) - f_{r,i}^H(h',t')\right]_+$$

where $f_{r,i}^H(h,t) = \|\mathbf{D}_i \cdot \mathbf{h} + \mathbf{r} - \boldsymbol{\alpha}_r^T \cdot \mathbf{D} \cdot \mathbf{t}\|$ is the corresponding energy function, and the subscript in P_r and N_r denotes the subsets with relation r. Intuitively, $\mathcal{L}_{r,i}^H$ measures, given the current tail attention vector $\boldsymbol{\alpha}_r^T$, if only one project matrix could be chosen for the head entity, how implausible D_i would be. Hence, $i^* = \arg\min_i \mathcal{L}_{r,i}^H$ gives us the best single projection matrix on the head side given $\boldsymbol{\alpha}_r^T$.

Now, in order to choose the best k matrices, we basically ignore the interaction among projection matrices, and update $\mathbf{I}_r^H$ in the following way:

$$\mathbf{I}_{r,i}^H \leftarrow \begin{cases} 1, & i \in \text{argpartition}_i(\mathcal{L}_{r,i}^H, k) \\ 0, & \text{otherwise} \end{cases}$$

where the function $\text{argpartition}_i(x_i, k)$ produces the index set of the lowest-k values of x_i.

Analogously, we can define the single-matrix cost $\mathcal{L}_{r,i}^T$ and the energy function $f_{r,i}^T(h,t)$ on the tail side in a symmetric way. Then, the update rule for $\mathbf{I}_r^H$ follows the same derivation. Admittedly, the approximation described here is relatively crude. But as we will show in section 4, the proposed algorithm yields good performance empirically. We leave the further improvement of the optimization method as future work.

3.3 Corrupted Sample Generating Method

Recall that we need to sample a negative triple (h', r, t') to compute hinge loss shown in Eq. 2, given a positive triple $(h, r, t) \in P$. The distribution of negative triple is denoted by $N(h, r, t)$. Previous work (Bordes et al., 2013; Lin et al., 2015b; Yang et al., 2015; Nguyen et al., 2016b) generally constructs a set of corrupted triples by replacing the head entity or tail entity with a random entity uniformly sampled from the KB.

However, uniformly sampling corrupted entities may not be optimal. Often, the head and tail entities associated a relation can only belong to a specific domain. When the corrupted entity comes from other domains, it is very easy for the model to induce a large energy gap between true triple and corrupted one. As the energy gap exceeds γ, there will be no training signal from this corrupted triple. In comparison, if the corrupted entity comes from the same domain, the task becomes harder for the model, leading to more consistent training signal.

Motivated by this observation, we propose to sample corrupted head or tail from entities in the same domain with a probability p_r and from the whole entity set with probability $1 - p_r$. The choice of relation-dependent probability p_r is specified in Appendix A.1. In the rest of the paper, we refer to the new proposed sampling method as "domain sampling".

4 Experiments

4.1 Setup

To evaluate link prediction, we conduct experiments on the WN18 (WordNet) and FB15k (Freebase) introduced by Bordes et al. (2013) and use the same training/validation/test split as in (Bordes et al., 2013). The information of the two datasets is given in Table 1.

Dataset	#E	#R	#Train	#Valid	#Test
WN18	40,943	18	141,442	5,000	5,000
FB15k	14,951	1,345	483,142	50,000	59,071

Table 1: Statistics of FB15k and WN18 used in experiments. #E, #R denote the number of entities and relation types respectively. #Train, #Valid and #Test are the numbers of triples in the training, validation and test sets respectively.

In knowledge base completion task, we evaluate model's performance of predicting the head entity or the tail entity given the relation and the other entity. For example, to predict head given relation r and tail t in triple (h, r, t), we compute the energy function $f_r(h', t)$ for each entity h' in the knowledge base and rank all the entities according to the energy. We follow Bordes et al. (2013) to report the *filter* results, i.e., removing all other correct candidates h' in ranking. The rank of the correct entity is then obtained and we report the mean rank (mean of the predicted ranks) and Hits@10 (top 10 accuracy). Lower mean rank or higher Hits@10 mean better performance.

4.2 Implementation Details

We initialize the projection matrices with identity matrices added with a small noise sampled from normal distribution $\mathcal{N}(0,\, 0.005^2)$. The entity and relation vectors of ITransF are initialized by TransE (Bordes et al., 2013), following Lin et al. (2015b); Ji et al. (2015); García-Durán et al. (2016, 2015); Lin et al. (2015a). We ran mini-batch SGD until convergence. We employ the "*Bernoulli*" sampling method to generate incorrect triples as used in Wang et al. (2014), Lin et al. (2015b), He et al. (2015), Ji et al. (2015) and Lin et al. (2015a).

STransE (Nguyen et al., 2016b) is the most similar knowledge embedding model to ours except that they use distinct projection matrices for each relation. We use the same hyperparameters as used in STransE and no significant improvement is observed when we alter hyperparameters. We set the margin γ to 5 and dimension of embedding n to 50 for WN18, and $\gamma = 1, n = 100$ for FB15k. We set the batch size to 20 for WN18 and 1000 for FB15k. The learning rate is 0.01 on WN18 and 0.1 on FB15k. We use 30 matrices on WN18 and 300 matrices on FB15k. All the models are implemented with Theano (Bergstra et al., 2010). The Softmax temperature is set to $1/4$.

4.3 Results & Analysis

The overall link prediction results[1] are reported in Table 2. Our model consistently outperforms previous models without external information on both the metrics of WN18 and FB15k. On WN18, we even achieve a much better mean rank with comparable Hits@10 than current state-of-the-art model IRN employing external information.

We can see that path information is very helpful on FB15k and models taking advantage of path information outperform intrinsic models by a significant margin. Indeed, a lot of facts are easier to recover with the help of multi-step inference. For example, if we know Barack Obama is born in Honolulu, a city in the United States, then we easily know the nationality of Obama is the United States. An straightforward way of extending our proposed model to k-step path $P = \{r_i\}_{i=1}^{k}$ is to define a path energy function $\|\boldsymbol{\alpha}_P^H \cdot \mathbf{D} \cdot \mathbf{h} + \sum_{r_i \in P} \mathbf{r}_i - \boldsymbol{\alpha}_P^T \cdot \mathbf{D} \cdot \mathbf{t}\|_\ell$, $\boldsymbol{\alpha}_P^H$ is a concept association related to the path. We plan to extend our model to multi-step path in the future.

To provide a detailed understanding why the proposed model achieves better performance, we present some further analysis in the sequel.

Performance on Rare Relations In the proposed ITransF, we design an attention mechanism to encourage knowledge sharing across different relations. Naturally, facts associated with rare relations should benefit most from such sharing, boosting the overall performance. To verify this hypothesis, we investigate our model's performance on relations with different frequency.

The overall distribution of relation frequencies resembles that of word frequencies, subject to the zipf's law. Since the frequencies of relations approximately follow a power distribution, their log

[1]Note that although IRN (Shen et al., 2016) does not explicitly exploit path information, it performs multi-step inference through the multiple usages of external memory. When IRN is allowed to access memory once for each prediction, its Hits@10 is 80.7, similar to models without path information.

Model	Additional Information	WN18		FB15k	
		Mean Rank	Hits@10	Mean Rank	Hits@10
SE (Bordes et al., 2011)	No	985	80.5	162	39.8
Unstructured (Bordes et al., 2014)	No	304	38.2	979	6.3
TransE (Bordes et al., 2013)	No	251	89.2	125	47.1
TransH (Wang et al., 2014)	No	303	86.7	87	64.4
TransR (Lin et al., 2015b)	No	225	92.0	77	68.7
CTransR (Lin et al., 2015b)	No	218	92.3	75	70.2
KG2E (He et al., 2015)	No	348	93.2	59	74.0
TransD (Ji et al., 2015)	No	212	92.2	91	77.3
TATEC (García-Durán et al., 2016)	No	-	-	**58**	76.7
NTN (Socher et al., 2013)	No	-	66.1	-	41.4
DISTMULT (Yang et al., 2015)	No	-	94.2	-	57.7
STransE (Nguyen et al., 2016b)	No	206 (244)	93.4 (94.7)	69	79.7
ITransF	No	**205**	94.2	65	81.0
ITransF (domain sampling)	No	223	**95.2**	77	**81.4**
RTransE (García-Durán et al., 2015)	Path	-	-	50	76.2
PTransE (Lin et al., 2015a)	Path	-	-	58	84.6
NLFeat (Toutanova and Chen, 2015)	Node + Link Features	-	94.3	-	87.0
Random Walk (Wei et al., 2016)	Path	-	94.8	-	74.7
IRN (Shen et al., 2016)	External Memory	249	*95.3*	*38*	*92.7*

Table 2: Link prediction results on two datasets. Higher Hits@10 or lower Mean Rank indicates better performance. Following Nguyen et al. (2016b) and Shen et al. (2016), we divide the models into two groups. The first group contains intrinsic models without using extra information. The second group make use of additional information. Results in the brackets are another set of results STransE reported.

frequencies are linear. The statistics of relations on FB15k and WN18 are shown in Figure 1. We can clearly see that the distributions exhibit long tails, just like the Zipf's law for word frequency.

In order to study the performance of relations with different frequencies, we sort all relations by their frequency in the training set, and split them into 3 buckets evenly so that each bucket has a similar interval length of log frequency.

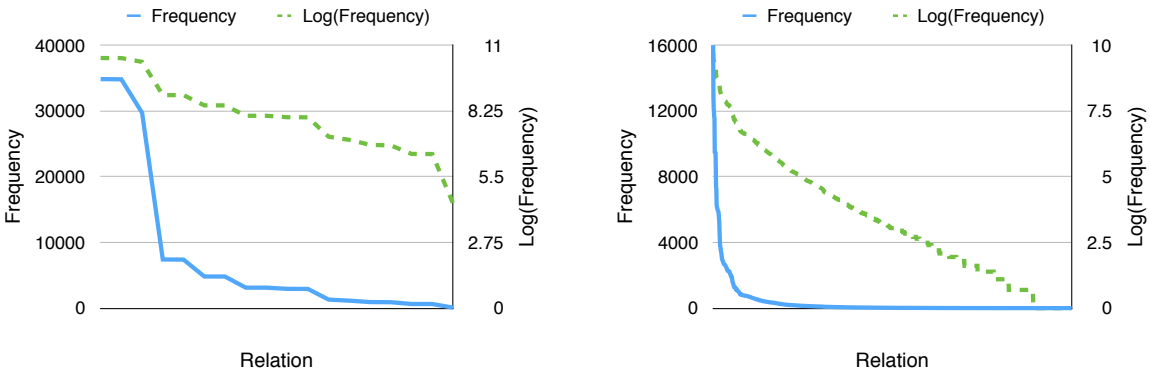
Within each bucket, we compare our model with STransE, as shown in Figure 2.[2] As we can see, on WN18, ITransF outperforms STransE by a significant margin on rare relations. In particular, in the last bin (rarest relations), the average Hits@10 increases from 55.2 to 93.8, showing the great benefits of transferring statistical strength from common relations to rare ones. The comparison on each relation is shown in Appendix A.2. On FB15k, we can also observe a similar pattern, although the degree of improvement is less significant. We conjecture the difference roots in the fact that many rare relations on FB15k have disjoint domains, knowledge transfer through common concepts is harder.

Interpretability In addition to the quantitative evidence supporting the effectiveness of knowledge sharing, we provide some intuitive examples to show how knowledge is shared in our model. As we mentioned earlier, the sparse attention vectors fully capture the association between relations and concepts and hence the knowledge transfer among relations. Thus, we visualize the attention vectors for several relations on both WN18 and FB15K in Figure 3.

For WN18, the words "hyponym" and "hypernym" refer to words with more specific or general meaning respectively. For example, PhD is a hyponym of student and student is a hypernym of PhD. As we can see, concepts associated with the head entities in one relation are also associated with the tail entities in its reverse relation. Further, "instance_hypernym" is a special hypernym with the head entity being an instance, and the tail entity being an abstract notion. A typical example is (*New York*, `instance_hypernym`, *city*). This connection has also been discovered by our model, indicated by the fact that "instance_hypernym(T)" and "hypernym(T)" share a common concept matrix. Finally, for symmetric relations like "similar_to", we see the head attention is identical to the tail attention, which well matches our intuition.

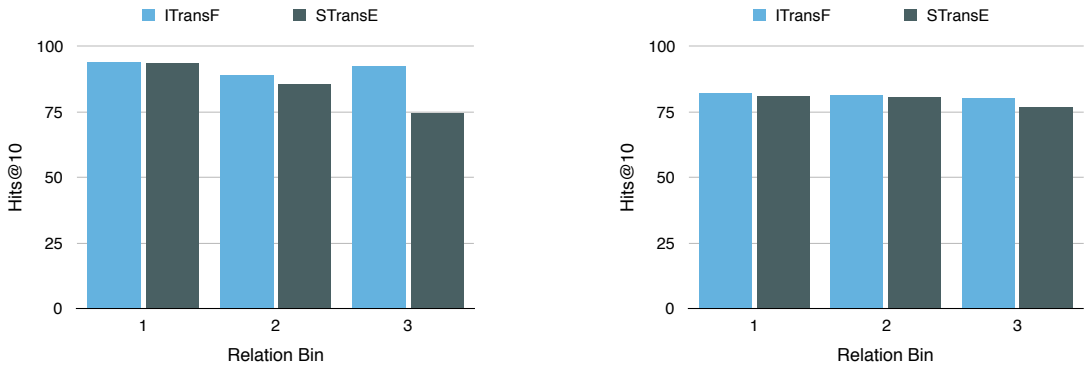
On FB15k, we also see the sharing between reverse relations, as in "(somebody) won_award_for (some work)" and "(some work) award_winning_work (somebody)". What's more, although relation "won_award_for" and "was_nominated_for" share the same concepts,

[2]Domain sampling is not employed.



(a) WN18 (b) FB15k

Figure 1: Frequencies and log frequencies of relations on two datasets. The X-axis are relations sorted by frequency.



(a) WN18 (b) FB15k

Figure 2: Hits@10 on relations with different amount of data. We give each relation the equal weight and report the average Hits@10 of each relation in a bin instead of reporting the average Hits@10 of each sample in a bin. Bins with smaller index corresponding to high-frequency relations.

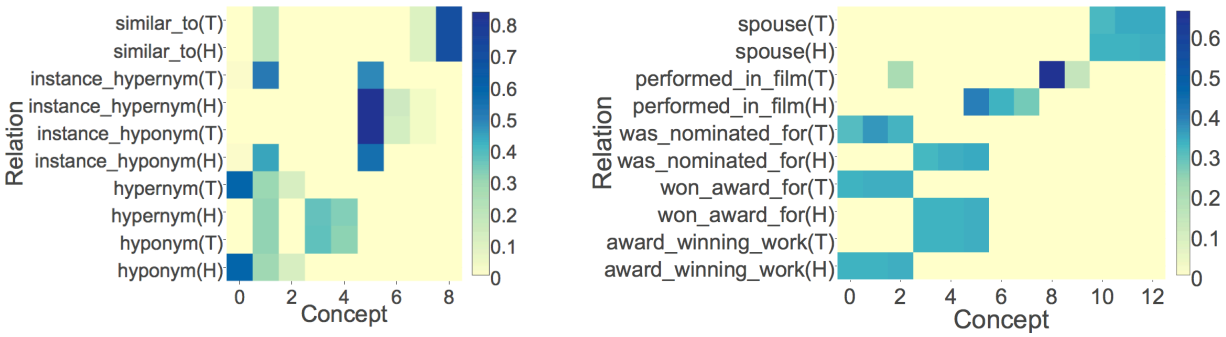
their attention distributions are different, suggesting distinct emphasis. Finally, symmetric relations like spouse behave similarly as mentioned before.

Model Compression A byproduct of parameter sharing mechanism employed by ITransF is a much more compact model with equal performance. Figure 5 plots the average performance of ITransF against the number of projection matrices m, together with two baseline models. On FB15k, when we reduce the number of matrices from 2200 to 30 ($\sim$ 90$\times$ compression), our model performance decreases by only 0.09% on Hits@10, still outperforming STransE. Similarly, on WN18, ITransF continues to achieve the best performance when we reduce the number of concept project matrices to 18.

5 Analysis on Sparseness

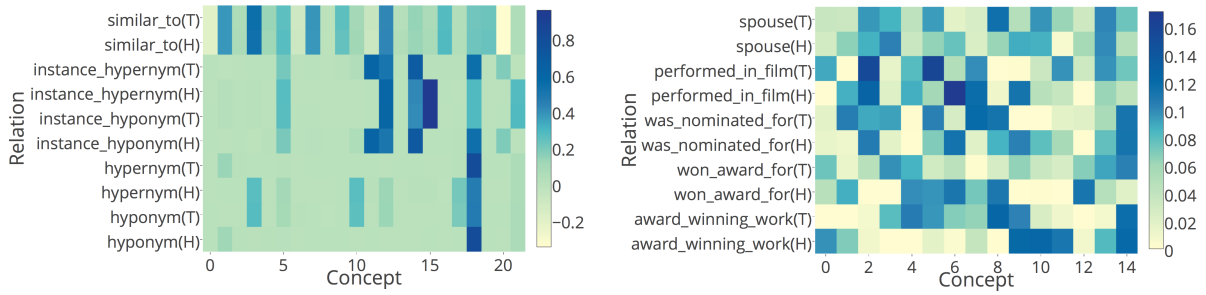
Sparseness is desirable since it contribute to interpretability and computational efficiency of our model. We investigate whether enforcing sparseness would deteriorate the model performance and compare our method with another sparse encoding methods in this section.

Dense Attention w/o ℓ_1 regularization Although ℓ_0 constrained model usually enjoys many practical advantages, it may deteriorate the model performance when applied improperly. Here, we show that our model employing sparse attention can achieve similar results with dense attention with a significantly less computational burden. We also compare dense attention with ℓ_1 regularization. We set the ℓ_1 coefficient to 0.001 in our experiments and does not apply Softmax since the ℓ_1 of a vector after Softmax is always 1. We compare models in a setting where the computation time of



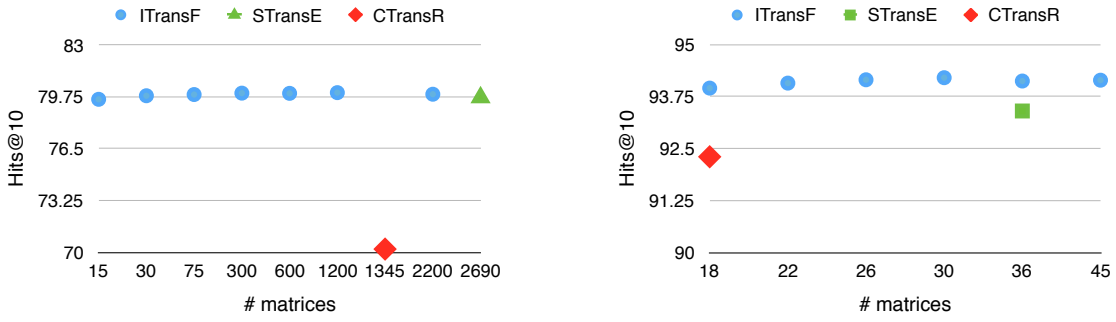
(a) WN18 (b) FB15k

Figure 3: Heatmap visualization of attention vectors for ITransF on WN18 and FB15k. Each row is an attention vector $\boldsymbol{\alpha}_r^H$ or $\boldsymbol{\alpha}_r^T$ for a relation's head or tail concepts.



(a) WN18 (b) FB15k

Figure 4: Heatmap visualization of ℓ_1 regularized dense attention vectors, which are not sparse. Note that the colorscale is not from 0 to 1 since Softmax is not applied.



(a) FB15k (b) WN18

Figure 5: Performance with different number of projection matrices. Note that the X-axis denoting the number of matrices is not linearly scaled.

dense attention model is acceptable[3]. We use 22 weight matrices on WN18 and 15 weight matrices on FB15k and train both the models for 2000 epochs.

The results are reported in Table 3. Generally, ITransF with sparse attention has slightly better or comparable performance comparing to dense attention. Further, we show the attention vectors of model with ℓ_1 regularized dense attention in Figure 4. We see that ℓ_1 regularization does not produce a sparse attention, especially on FB15k.

Nonnegative Sparse Encoding In the proposed model, we induce the sparsity by a carefully designed iterative optimization procedure. Apart from this approach, one may utilize sparse encoding techniques to obtain sparseness based on the pretrained projection matrices from STransE. Concretely, stacking $|2R|$ pretrained projection

[3]With 300 projection matrices, it takes $1h1m$ to run one epoch for a model with dense attention.

Method	WN18			FB15k		
	MR	H10	Time	MR	H10	Time
Dense	**199**	94.0	4m34s	69	79.4	4m30s
Dense + ℓ_1	228	**94.2**	4m25s	131	78.9	5m47s
Sparse	207	94.1	**2m32s**	**67**	**79.6**	**1m52s**

Table 3: Performance of model with dense attention vectors or sparse attention vectors. MR, H10 and Time denotes mean rank, Hits@10 and training time per epoch respectively

matrices into a 3-dimensional tensor $X \in \mathbb{R}^{2|R|\times n\times n}$, similar sparsity can be induced by solving an ℓ_1-regularized tensor completion problem $\min_{\mathbf{A},\mathbf{D}} ||\mathbf{X} - \mathbf{DA}||_2^2 + \lambda\|\mathbf{A}\|_{\ell_1}$. Basically, $\mathbf{A}$ plays the same role as the attention vectors in our model. For more details, we refer readers to (Faruqui et al., 2015).

For completeness, we compare our model with the aforementioned approach[4]. The comparison is summarized in table 4. On both benchmarks, ITransF achieves significant improvement against sparse encoding on pretrained model. This performance gap should be expected since the objective function of sparse encoding methods is to minimize the reconstruction loss rather than optimize the criterion for link prediction.

Method	WN18		FB15k	
	MR	H10	MR	H10
Sparse Encoding	211	86.6	66	79.1
ITransF	**205**	**94.2**	**65**	**81.0**

Table 4: Different methods to obtain sparse representations

6 Related Work

In KBC, CTransR (Lin et al., 2015b) enables relation embedding sharing across similar relations, but they cluster relations before training rather than learning it in a principled way. Further, they do not solve the data sparsity problem because there is no sharing of projection matrices which have a lot more parameters. Learning the association between semantic relations has been used in related problems such as relational similarity measurement (Turney, 2012) and relation adaptation (Bollegala et al., 2015).

Data sparsity is a common problem in many fields. Transfer learning (Pan and Yang, 2010) has been shown to be promising to transfer knowledge and statistical strengths across similar models or languages. For example, Bharadwaj et al. (2016) transfers models on resource-rich languages to low resource languages by parameter sharing through common phonological features in name entity recognition. Zoph et al. (2016) initialize from models trained by resource-rich languages to translate low-resource languages.

Several works on obtaining a sparse attention (Martins and Astudillo, 2016; Makhzani and Frey, 2014; Shazeer et al., 2017) share a similar idea of sorting the values before softmax and only keeping the K largest values. However, the sorting operation in these works is not GPU-friendly.

The block iterative optimization algorithm in our work is inspired by LightRNN (Li et al., 2016). They allocate every word in the vocabulary in a table. A word is represented by a row vector and a column vector depending on its position in the table. They iteratively optimize embeddings and allocation of words in tables.

7 Conclusion and Future Work

In summary, we propose a knowledge embedding model which can discover shared hidden concepts, and design a learning algorithm to induce the interpretable sparse representation. Empirically, we show our model can improve the performance on two benchmark datasets without external resources, over all previous models of the same kind.

In the future, we plan to enable ITransF to perform multi-step inference, and extend the sharing mechanism to entity and relation embeddings, further enhancing the statistical binding across parameters. In addition, our framework can also be applied to multi-task learning, promoting a finer sharing among different tasks.

Acknowledgments

We thank anonymous reviewers and Graham Neubig for valuable comments. We thank Yulun Du, Paul Mitchell, Abhilasha Ravichander, Pengcheng Yin and Chunting Zhou for suggestions on the draft. We are also appreciative for the great working environment provided by staff in LTI.

This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program.

[4]We use the toolkit provided by (Faruqui et al., 2015).

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1533–1544.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*. Austin, TX, volume 4, page 3.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1462–1472.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. pages 1247–1250.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Embedding semantic relations into word representations. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A Semantic Matching Energy Function for Learning with Multi-relational Data. *Machine Learning* 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. pages 301–306.

Zihang Dai, Lei Li, and Wei Xu. 2016. Cfo: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 800–810.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1491–1500.

Christiane D. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. 2015. Composing Relationships with Translations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 286–290.

Alberto García-Durán, Antoine Bordes, Nicolas Usunier, and Yves Grandvalet. 2016. Combining Two and Three-Way Embedding Models for Link Prediction in Knowledge Bases. *Journal of Artificial Intelligence Research* 55:715–742.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing Knowledge Graphs in Vector Space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 318–327.

Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to Represent Knowledge Graphs with Gaussian Embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. pages 623–632.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge Graph Embedding via Dynamic Mapping Matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. pages 687–696.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web* 6(2):167–195.

Xiang Li, Tao Qin, Jian Yang, and Tieyan Liu. 2016. LightRNN: Memory and Computation-Efficient Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 29*.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling Relation Paths for Representation Learning of Knowledge Bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence Learning*, pages 2181–2187.

Alireza Makhzani and Brendan Frey. 2014. K-sparse autoencoders. In *Proceedings of the International Conference on Learning Representations*.

André FT Martins and Ramón Fernandez Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33th International Conference on Machine Learning*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, pages 1003–1011.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016a. Neighborhood mixture model for knowledge base completion. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*. Association for Computational Linguistics, page 4050.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016b. STransE: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 460–466.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE, to appear* .

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on Machine Learning*. pages 809–816.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10):1345–1359.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *Proceedings of the International Conference on Learning Representations*.

Yelong Shen, Po-Sen Huang, Ming-Wei Chang, and Jianfeng Gao. 2016. Implicit reasonet: Modeling large-scale structured relationships with shared memory. *arXiv preprint arXiv:1611.04642* .

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*, pages 926–934.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web*. pages 697–706.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* pages 267–288.

Kristina Toutanova and Danqi Chen. 2015. Observed Versus Latent Features for Knowledge Base and Text Inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. pages 57–66.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1499–1509.

Peter D Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research* 44:533–585.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Zhuoyu Wei, Jun Zhao, and Kang Liu. 2016. Mining inference formulas by goal-directed random walks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1379–1388.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge Base Completion via Search-based Question Answering. In *Proceedings of the 23rd International Conference on World Wide Web*. pages 515–526.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1321–1331.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1568–1575.

A Appendix

A.1 Domain Sampling Probability

In this section, we define the probability p_r to generate a negative sample from the same domain mentioned in Section 3.3. The probability cannot be too high to avoid generating negative samples that are actually correct, since there are generally a lot of facts missing in KBs.

Specifically, let $\mathrm{M}_r^H = \{h \mid \exists t(h,r,t) \in P\}$ and $\mathrm{M}_r^T = \{t \mid \exists h(h,r,t) \in P\}$ denote the head or tail domain of relation r. Suppose $N_r = \{(h,r,t) \in P\}$ is the induced set of edges with relation r. We define the probability p_r as

$$p_r = min(\frac{\lambda|\mathrm{M}_r^T||\mathrm{M}_r^H|}{|N_r|}, 0.5) \tag{4}$$

Our motivation of such a formulation is as follows: Suppose O_r is the set that contains all truthful fact triples on relation r, i.e., all triples in training set and all other missing correct triples. If we assume all fact triples within the domain has uniform probability of being true, the probability of a random triple being correct is $Pr((h,r,t) \in O_r \mid h \in \mathrm{M}_r^H, t \in \mathrm{M}_r^T) = \frac{|O_r|}{|\mathrm{M}_r^H||\mathrm{M}_r^T|}$

Assume that all facts are missing with a probability λ, then $|N_r| = \lambda|O_r|$ and the above probability can be approximated by $\frac{|N_r|}{\lambda|\mathrm{M}_r^H||\mathrm{M}_r^T|}$. We want the probability of generating a negative sample from the domain to be inversely proportional to the probability of the sample being true, so we define the probability as Eq. 4. The results in section 4 are obtained with λ set to 0.001.

We compare how different value of λ would influence our model's performance in Table. 5. With large λ and higher domain sampling probability, our model's Hits@10 increases while mean rank also increases. The rise of mean rank is due to higher probability of generating a valid triple as a negative sample causing the energy of a valid triple to increase, which leads to a higher overall rank of a correct entity. However, the reasoning capability is boosted with higher Hits@10 as shown in the table.

A.2 Performance on individual relations of WN18

We plot the performance of ITransF and STransE on each relation. We see that the improvement is greater on rare relations.

Method	WN18		FB15k	
	MR	H10	MR	H10
$\lambda = 0.0003$	**217**	95.0	**68**	80.4
$\lambda = 0.001$	223	**95.2**	73	80.6
$\lambda = 0.003$	239	**95.2**	82	**80.9**

Table 5: Different λ's effect on our model performance. The compared models are trained for 2000 epochs

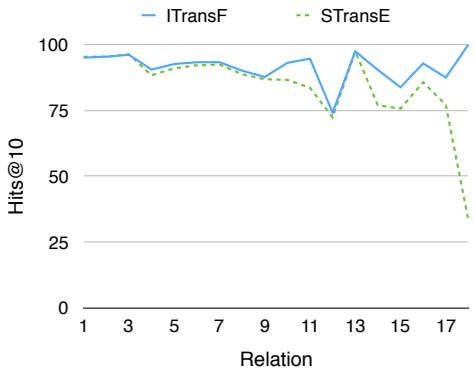


Figure 6: Hits@10 on each relation in WN18. The relations are sorted according to their frequency.

Learning a Neural Semantic Parser from User Feedback

Srinivasan Iyer[†◇], **Ioannis Konstas**[†], **Alvin Cheung**[†]
Jayant Krishnamurthy[‡] and **Luke Zettlemoyer**[†‡]

[†]Paul G. Allen School of Computer Science & Engineering, Univ. of Washington, Seattle, WA
{sviyer,ikonstas,akcheung,lsz}@cs.washington.edu

[‡]Allen Institute for Artificial Intelligence, Seattle, WA
{jayantk,lukez}@allenai.org

Abstract

We present an approach to rapidly and easily build natural language interfaces to databases for new domains, whose performance improves over time based on user feedback, and requires minimal intervention. To achieve this, we adapt neural sequence models to map utterances directly to SQL with its full expressivity, bypassing any intermediate meaning representations. These models are immediately deployed online to solicit feedback from real users to flag incorrect queries. Finally, the popularity of SQL facilitates gathering annotations for incorrect predictions using the crowd, which is directly used to improve our models. This complete feedback loop, without intermediate representations or database specific engineering, opens up new ways of building high quality semantic parsers. Experiments suggest that this approach can be deployed quickly for any new target domain, as we show by learning a semantic parser for an online academic database from scratch.

1 Introduction

Existing semantic parsing approaches for building natural language interfaces to databases (NLIDBs) either use special-purpose intermediate meaning representations that lack the full expressivity of database query languages or require extensive feature engineering, making it difficult to deploy them in new domains. We present a robust approach to quickly and easily learn and deploy semantic parsers from scratch, whose performance improves over time based on user feedback, and requires very little expert intervention.

Most recent papers of Michael I. Jordan

```
SELECT paper.paperId, paper.year
FROM paper, writes, author
WHERE paper.paperId = writes.paperId
  AND writes.authorId = author.authorId
  AND author.authorName = "michael i. jordan"
  AND paper.year =
    (SELECT max(paper.year)
     FROM paper, writes, author
     WHERE paper.paperId = writes.paperId
      AND writes.authorId = author.authorId
      AND author.authorName = "michael i. jordan");
```

I'd like to book a flight from San Diego to Toronto

```
SELECT DISTINCT f1.flight_id
FROM flight f1, airport_service a1, city c1,
  airport_service a2, city c2
WHERE f1.from_airport = a1.airport_code
  AND a1.city_code = c1.city_code
  AND c1.city_name = 'san diego'
  AND f1.to_airport = a2.airport_code
  AND a2.city_code = c2.city_code
  AND c2.city_name = 'toronto';
```

Figure 1: Utterances with corresponding SQL queries to answer them for two domains, an academic database and a flight reservation database.

To learn these semantic parsers, we (1) adapt neural sequence models to map utterances directly to SQL thereby bypassing intermediate representations and taking full advantage of SQL's querying capabilities, (2) immediately deploy the model online to solicit questions and user feedback on results to reduce SQL annotation efforts, and (3) use crowd workers from skilled markets to provide SQL annotations that can directly be used for model improvement, in addition to being easier and cheaper to obtain than logical meaning representations. We demonstrate the effectiveness of the complete approach by successfully learning a semantic parser for an academic domain by simply deploying it online for three days.

This type of interactive learning is related to a number of recent ideas in semantic parsing, in-

[◇]Work done partly during an internship at the Allen Institute for Artificial Intelligence.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 963–973
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1089

cluding batch learning of models that directly produce programs (e.g., regular expressions (Locascio et al., 2016)), learning from paraphrases (often gathered through crowdsourcing (Wang et al., 2015)), data augmentation (e.g. based on manually engineered semantic grammars (Jia and Liang, 2016)) and learning through direct interaction with users (e.g., where a single user teaches the model new concepts (Wang et al., 2016)). However, there are unique advantages to our approach, including showing (1) that non-linguists can write SQL to encode complex, compositional computations (see Fig 1 for an example), (2) that external paraphrase resources and the structure of facts from the target database itself can be used for effective data augmentation, and (3) that actual database users can effectively drive the overall learning by simply providing feedback about what the model is currently getting correct.

Our experiments measure the performance of these learning advances, both in batch on existing datasets and through a simple online experiment for the full interactive setting. For the batch evaluation, we use sentences from the benchmark GeoQuery and ATIS domains, converted to contain SQL meaning representations. Our neural learning with data augmentation achieves reasonably high accuracies, despite the extra complexities of mapping directly to SQL. We also perform simulated interactive learning on this data, showing that with perfect user feedback our full approach could learn high quality parsers with only 55% of the data. Finally, we do a small scale online experiment for a new domain, academic paper metadata search, demonstrating that actual users can provide useful feedback and our full approach is an effective method for learning a high quality parser that continues to improve over time as it is used.

2 Related Work

Although diverse meaning representation languages have been used with semantic parsers – such as regular expressions (Kushman and Barzilay, 2013; Locascio et al., 2016), Abstract Meaning Representations (AMR) (Artzi et al., 2015; Misra and Artzi, 2016), and systems of equations (Kushman et al., 2014; Roy et al., 2016) – parsers for querying databases have typically used either logic programs (Zelle and Mooney, 1996), lambda calculus (Zettlemoyer and Collins, 2005), or λ-DCS (Liang et al., 2013) as the meaning representation language. All three of these languages are modeled after natural language to simplify parsing. However, none of them is used to query databases outside of the semantic parsing literature; therefore, they are understood by few people and not supported by standard database implementations. In contrast, we parse directly to SQL, which is a popular database query language with wide usage and support. Learning parsers directly from SQL queries has the added benefit that we can potentially hire programmers on skilled-labor crowd markets to provide labeled examples, such as UpWork[1], which we demonstrate in this work.

A few systems have been developed to directly generate SQL queries from natural language (Popescu et al., 2003; Giordani and Moschitti, 2012; Poon, 2013). However, all of these systems make strong assumptions on the structure of queries: they use manually engineered rules that can only generate a subset of SQL, require lexical matches between question tokens and table/column names, or require questions to have a certain syntactic structure. In contrast, our approach can generate arbitrary SQL queries, only uses lexical matching for entity names, and does not depend on syntactic parsing.

We use a neural sequence-to-sequence model to directly generate SQL queries from natural language questions. This approach builds on recent work demonstrating that such models are effective for tasks such as machine translation (Bahdanau et al., 2015) and natural language generation (Kiddon et al., 2016). Recently, neural models have been successfully applied to semantic parsing with simpler meaning representation languages (Dong and Lapata, 2016; Jia and Liang, 2016) and short regular expressions (Locascio et al., 2016). Our work extends these results to the task of SQL generation. Finally, Ling et al. (2016) generate Java/Python code for trading cards given a natural language description; however, this system suffers from low overall accuracy.

A final direction of related work studies methods for reducing the annotation effort required to train a semantic parser. Semantic parsers have been trained from various kinds of annotations, including labeled queries (Zelle and Mooney, 1996; Wong and Mooney, 2007; Zettlemoyer and Collins, 2005), question/answer pairs (Liang et al., 2013; Kwiatkowski et al., 2013; Berant et al.,

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

2013), distant supervision (Krishnamurthy and Mitchell, 2012; Choi et al., 2015), and binary correct/incorrect feedback signals (Clarke et al., 2010; Artzi and Zettlemoyer, 2013). Each of these schemes presents a particular trade-off between annotation effort and parser accuracy; however, recent work has suggested that labeled queries are the most effective (Yih et al., 2016). Our approach trains on fully labeled SQL queries to maximize accuracy, but uses binary feedback from users to reduce the number of queries that need to be labeled. Annotation effort can also be reduced by using crowd workers to paraphrase automatically generated questions (Wang et al., 2015); however, this approach may not generate the questions that users actually want to ask the database – an experiment in this paper demonstrated that 48% of users' questions in a calendar domain could not be generated.

3 Feedback-based Learning

Our feedback-based learning approach can be used to quickly deploy semantic parsers to create NLIDBs for any new domain. It is a simple interactive learning algorithm that deploys a preliminary semantic parser, then iteratively improves this parser using user feedback and selective query annotation. A key requirement of this algorithm is the ability to cheaply and efficiently annotate queries for chosen user utterances. We address this requirement by developing a model that directly outputs SQL queries (Section 4), which can also be produced by crowd workers.

Our algorithm alternates between stages of training the model and making predictions to gather user feedback, with the goal of improving performance in each successive stage. The procedure is described in Algorithm 1. Our neural model $\mathscr{N}$ is initially trained on synthetic data T generated by domain-independent schema templates (see Section 4), and is then ready to answer new user questions, n. The results $\mathscr{R}$ of executing the predicted SQL query q are presented to the user who provides a binary correct/incorrect feedback signal. If the user marks the result correct, the pair (n, q) is added to the training set. If the user marks the result incorrect, the algorithm asks a crowd worker to annotate the utterance with the correct query, $\hat{q}$, and adds $(n, \hat{q})$ to the training set. This procedure can be repeated indefinitely, ideally increasing parser accuracy and requesting fewer annotations in each successive stage.

```
Procedure LEARN(schema)
    T ← initial_data(schema)
    while true do
        𝒯 ← T ∪ paraphrase(T)
        𝒩 ← train_model(𝒯)
        for n ∈ new utterances do
            q ← predict(𝒩, n)
            ℛ ← execute(q)
            f ← feedback(ℛ)
            if f = correct then
                T ← T ∪ (n, q)
            else if f = wrong then
                q̂ ← annotate(n)
                T ← T ∪ (n, q̂)
            end
        end
    end
end
```

Algorithm 1: Feedback-based learning.

4 Semantic Parsing to SQL

We use a neural sequence-to-sequence model for mapping natural language questions directly to SQL queries and this allows us to scale our feedback-based learning approach, by easily crowdsourcing labels when necessary. We further present two data augmentation techniques which use content from the database schema and external paraphrase resources.

4.1 Model

We use an encoder-decoder model with global attention, similar to Luong et al. (2015), where the anonymized utterance (see Section 4.2) is encoded using a bidirectional LSTM network, then decoded to directly predict SQL query tokens. Fixed pre-trained word embeddings from word2vec (Mikolov et al., 2013) are concatenated to the embeddings that are learned for source tokens from the training data. The decoder predicts a conditional probability distribution over possible values for the next SQL token given the previous tokens using a combination of the previous SQL token embedding, attention over the hidden states of the encoder network, and an attention signal from the previous time step.

Formally, if $\mathbf{q_i}$ represents an embedding for the

i^{th} SQL token q_i, the decoder distribution is

$$p(q_i|q_1, \ldots, q_{i-1}) \propto \exp\left(\mathbf{W} \tanh(\hat{\mathbf{W}}[\mathbf{h_i} : \mathbf{c_i}])\right)$$

where $\mathbf{h_i}$ represents the hidden state output of the decoder LSTM at the i^{th} timestep, $\mathbf{c_i}$ represents the context vector generated using an attention weighted sum of encoder hidden states based on $\mathbf{h_i}$, and, $\mathbf{W}$ and $\hat{\mathbf{W}}$ are linear transformations. If $\mathbf{s_j}$ is the hidden representation generated by the encoder for the j^{th} word in the utterance (k words long), then the context vectors are defined to be:

$$\mathbf{c_i} = \sum_{j=1}^{k} \alpha_{i,j} \cdot \mathbf{s_j}$$

The attention weights $\alpha_{i,j}$ are computed using an inner product between the decoder hidden state for the current timestep $\mathbf{h_i}$, and the hidden representation of the j^{th} source token $\mathbf{s_j}$:

$$\alpha_{i,j} = \frac{\exp(\mathbf{h_i}^{\mathrm{T}} \mathbf{F} \mathbf{s_j})}{\sum_{j=1}^{k} \exp(\mathbf{h_i}^{\mathrm{T}} \mathbf{F} \mathbf{s_j})}$$

where $\mathbf{F}$ is a linear transformation. The decoder LSTM cell f computes the next hidden state $\mathbf{h_i}$, and cell state, $\mathbf{m_i}$, based on the previous hidden and cell states, $\mathbf{h_{i-1}}, \mathbf{m_{i-1}}$, the embeddings of the previous SQL token $\mathbf{q_{i-1}}$ and the context vector of the previous timestep, $\mathbf{c_{i-1}}$

$$\mathbf{h_i}, \mathbf{m_i} = f(\mathbf{h_{i-1}}, \mathbf{m_{i-1}}, \mathbf{q_{i-1}}, \mathbf{c_{i-1}})$$

We apply dropout on non-recurrent connections for regularization, as suggested by Pham et al. (2014). Beam search is used for decoding the SQL queries after learning.

4.2 Entity Anonymization

We handle entities in the utterances and SQL by replacing them with their types, using incremental numbering to model multiple entities of the same type (e.g., `CITY_NAME_1`). During training, when the SQL is available, we infer the type from the associated column name; for example, Boston is a city in `city.city_name = 'Boston'`. To recognize entities in the utterances at test time, we build a search engine on all entities from the target database. For every span of words (starting with a high span size and progressively reducing it), we query the search engine using a TF-IDF scheme to retrieve the entity that most closely matches the span, then replace the span with the entity's type. We store these mappings and apply them to the generated SQL to fill in the entity names. TF-IDF matching allows some flexibility in matching entity names in utterances, for example, a user could say *Donald Knuth* instead of *Donald E. Knuth*.

4.3 Data Augmentation

We present two data augmentation strategies that either (1) provide the initial training data to start the interactive learning, before more labeled examples become available, or (2) use external paraphrase resources to improve generalization.

Schema Templates To bootstrap the model to answer simple questions initially, we defined 22 language/SQL templates that are schema-agnostic, so they can be applied to any database. These templates contain slots whose values are populated given a database schema. An example template is shown in Figure 2a. The `<ENT>` types represent tables in the database schema, `<ENT>.<COL>` represents a column in the particular table and `<ENT>.<COL>.<TYPE>` represents the type associated with the particular column. A template is instantiated by first choosing the entities and attributes. Next, join conditions, i.e., `JOIN_FROM` and `JOIN_WHERE` clauses, are generated from the tables on the shortest path between the chosen tables in the database schema graph, which connects tables (graph nodes) using foreign key constraints. Figure 2b shows an instantiation of a template using the path author - writes - paper - paperdataset - dataset. SQL queries generated in this manner are guaranteed to be executable on the target database. On the language side, an English name of each entity is plugged into the template to generate an utterance for the query.

Paraphrasing The second data augmentation strategy uses the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) to automatically generate paraphrases of training utterances. Such methods have been recently used to improve performance for parsing to logical forms (Chen et al., 2016). PPDB contains over 220 million paraphrase pairs divided into 6 sets (small to XXXL) based on precision of the paraphrases. We use the one-one and one-many paraphrases from the large version of PPDB. To paraphrase a training utterance, we pick a random word in the utterance that is not a stop word or entity and replace it with a random paraphrase. We perform paraphrase expansion on all examples labeled during learning, as well as the initial seed examples from schema templates.

```
Get all <ENT1>.<NAME> having
<ENT2>.<COL1>.<NAME> as <ENT2>.<COL1>.<TYPE>

SELECT <ENT1>.<DEF> FROM JOIN_FROM(<ENT1>, <ENT2>)
WHERE JOIN_WHERE(<ENT1>, <ENT2>) AND
 <ENT2>.<COL1> = <ENT2>.<COL1>.<TYPE>
```

(a) Schema template

```
Get all author having dataset  as DATASET_TYPE

SELECT author.authorId
FROM author , writes , paper , paperDataset , dataset
WHERE author.authorId = writes.authorId
 AND writes.paperId = paper.paperId
 AND paper.paperId = paperDataset.paperId
 AND paperDataset.datasetId = dataset.datasetId
 AND dataset.datasetName = DATASET_TYPE
```

(b) Generated utterance-SQL pair

Figure 2: (a) Example schema template consisting of a question and SQL query with slots to be filled with database entities, columns, and values; (b) Entity-anonymized training example generated by applying the template to an academic database.

5 Benchmark Experiments

Our first set of experiments demonstrates that our semantic parsing model has comparable accuracy to previous work, despite the increased difficulty of directly producing SQL. We demonstrate this result by running our model on two benchmark datasets for semantic parsing, GEO880 and ATIS.

5.1 Data sets

GEO880 is a collection of 880 utterances issued to a database of US geographical facts (Geobase), originally in Prolog format. Popescu et al. (2003) created a relational database schema for Geobase together with SQL queries for a subset of 700 utterances. To compare against prior work on the full corpus, we annotated the remaining utterances and used the standard 600/280 training/test split (Zettlemoyer and Collins, 2005).

ATIS is a collection of 5,418 utterances to a flight booking system, accompanied by a relational database and SQL queries to answer the questions. We use 4,473 utterances for training, 497 for development and 448 for test, following Kwiatkowski et al. (2011). The original SQL queries were very inefficient to execute due to the use of `IN` clauses, so we converted them to joins (Ramakrishnan and Gehrke, 2003) while verifying that the output of the queries was unchanged.

Table 1 shows characteristics of both data sets. GEO880 has shorter queries but is more compositional: almost 40% of the SQL queries have at least one nested subquery. ATIS has the longest utterances and queries, with an average utterance length of 11 words and an average SQL query length of 67 tokens. They also operate on approximately 6 tables per query on average. We will release our processed versions of both datasets.

	Geo880	ATIS	SCHOLAR
Avg. NL length	7.56	10.97	6.69
NL vocab size	151	808	303
Avg. SQL length	16.06	67.01	28.85
SQL vocab size	89	605	163
% Subqueries > 1	39.8	12.42	2.58
# Tables	1.19	5.88	3.33

Table 1: Utterance and SQL query statistics for each dataset. Vocabulary sizes are counted after entity anonymization.

5.2 Experimental Methodology

We follow a standard train/dev/test methodology for our experiments. The training set is augmented using schema templates and 3 paraphrases per training example, as described in Section 4. Utterances were anonymized by replacing them with their corresponding types and all words that occur only once were replaced by UNK symbols. The development set is used for hyperparameter tuning and early stopping. For GEO880, we use cross validation on the training set to tune hyperparameters. We used a minibatch size of 100 and used Adam (Kingma and Ba, 2015) with a learning rate of 0.001 for 70 epochs for all our experiments. We used a beam size of 5 for decoding. We report test set accuracy of our SQL query predictions by executing them on the target database and comparing the result with the true result.

5.3 Results

Tables 2 and 3 show test accuracies based on denotations for our model on GEO880 and ATIS respectively, compared with previous work.[2] To our knowledge, this is the first result on directly parsing to SQL to achieve comparable performance to prior work without using any database-specific feature engineering. Popescu et al. (2003) and Giordani and Moschitti (2012) also directly produce SQL queries but on a subset of 700 examples from GEO880. The former only works on semantically tractable utterances where words can be un-

[2]Note that 2.8% of GEO880 and 5% ATIS gold test set SQL queries (before any processing) produced empty results.

System	Acc.
Ours (SQL)	82.5
Popescu et al. (2003) (SQL)	77.5*
Giordani and Moschitti (2012) (SQL)	87.2*
Dong and Lapata (2016)	84.6◇†
Jia and Liang (2016)	89.3◇
Liang et al. (2013)	91.1◇

Table 2: Accuracy of SQL query results on the Geo880 corpus; * use Geo700; ◇ convert to logical forms instead of SQL; † measure accuracy in terms of obtaining the correct logical form, other systems, including ours, use denotations.

System	Acc.
Ours (SQL)	79.24
GUSP (Poon, 2013) (SQL)	74.8
GUSP++ (Poon, 2013) (SQL)	83.5
Zettlemoyer and Collins (2007)	84.6◇†
Dong and Lapata (2016)	84.2◇†
Jia and Liang (2016)	83.3◇
Wang et al. (2014)	91.3◇†

Table 3: Accuracy of SQL query results on ATIS; ◇ convert to logical forms instead of SQL; † measure accuracy in terms of obtaining the correct logical form, other systems, including ours, use denotations.

ambiguously mapped to schema elements, while the latter uses a reranking approach that also limits the complexity of SQL queries that can be handled. GUSP (Poon, 2013) creates an intermediate representation that is then deterministically converted to SQL to obtain an accuracy of 74.8% on ATIS, which is boosted to 83.5% using manually introduced disambiguation rules. However, it requires a lot of SQL specific engineering (for example, special nodes for argmax) and is hard to extend to more complex SQL queries.

On both datasets, our SQL model achieves reasonably high accuracies approaching that of the best non-SQL results. Most relevant to this work are the neural sequence based approaches of Dong and Lapata (2016) and Jia and Liang (2016). We note that Jia and Liang (2016) use a data recombination technique that boosts accuracy from 85.0 on GEO880 and 76.3 on ATIS; this technique is also compatible with our model and we hope to experiment with this in future work. Our results demonstrate that these models are powerful enough to directly produce SQL queries. Thus, our methods enable us to utilize the full expressivity of the SQL language without any extensions that certain logical representations require to answer more complex queries. More importantly, it can be immediately deployed for users in new domains, with a large programming community available for annotation, and thus, fits effectively into a framework for interactive learning.

System	GEO880	ATIS
Ours	84.8	86.2
- paraphrases	81.8	84.3
- templates	84.7	85.7

Table 4: Addition of paraphrases to the training set helps performance, but template based data augmentation does not significantly help in the fully supervised setting. Accuracies are reported on the standard dev set for ATIS and on the training set, using cross-validation, for Geo880.

We perform ablation studies on the development sets (see Table 4) and find that paraphrasing using PPDB consistently helps boost performance. However, unlike in the interactive experiments (Section 6), data augmentation using schema templates does not improve performance in the fully supervised setting.

6 Interactive Learning Experiments

In this section, we learn a semantic parser for an academic domain from scratch by deploying an online system using our interactive learning algorithm (Section 3). After three train-deploy cycles, the system correctly answered 63.51% of user's questions. To our knowledge, this is the first effort to learn a semantic parser using a live system, and is enabled by our models that can directly parse language to SQL without manual intervention.

6.1 User Interface

We developed a web interface for accepting natural language questions to an academic database from users, using our model to generate a SQL query, and displaying the results after execution. Several example utterances are also displayed to help users understand the domain. Together with the results of the generated SQL query, users are prompted to provide feedback which is used for

interactive learning. Screenshots of our interface are included in our Supplementary Materials.

Collecting accurate user feedback on predicted queries is a key challenge in the interactive learning setting for two reasons. First, the system's results can be incorrect due to poor entity identification or incompleteness in the database, neither of which are under the semantic parser's control. Second, it can be difficult for users to determine if the presented results are in fact correct. This determination is especially challenging if the system responds with the correct type of result, for example, if the user requests "papers at ACL 2016" and the system responds with all ACL papers.

We address this challenge by providing users with two assists for understanding the system's behavior, and allowing users to provide more granular feedback than simply correct/incorrect. The first assist is **type highlighting**, which highlights entities identified in the utterance, for example, "paper by *Michael I. Jordan (AUTHOR)* in *ICRA (VENUE)* in *2016 (YEAR)*." This assist is especially helpful because the academic database contains noisy keyword and dataset tables that were automatically extracted from the papers. The second assist is **utterance paraphrasing**, which shows the user another utterance that maps to the same SQL query. For example, for the above query, the system may show "what papers does *Michael I. Jordan (AUTHOR)* have in *ICRA (VENUE)* in *2016 (YEAR)*." This assist only appears if a matching query (after entity anonymization) exists in the model's training set.

Using these assists and the predicted results, users are asked to select from five feedback options: *Correct*, *Wrong Types*, *Incomplete Result*, *Wrong Result* and *Can't Tell*. The *Correct* and *Wrong Result* options represent scenarios when the user is satisfied with the result, or the result is identifiably wrong, respectively. *Wrong Types* indicates incorrect entity identification, which can be determined from type highlighting. *Incomplete Result* indicates that the query is correct but the result is not; this outcome can occur because the database is incomplete. *Can't Tell* indicates that the user is unsure about the feedback to provide.

6.2 Three-Stage Online Experiment

In this experiment, using our developed user interface, we use Algorithm 1 to learn a semantic parser from scratch. The experiment had three stages; in each stage, we recruited 10 new users (computer science graduate students) and asked them to issue at least 10 utterances each to the system and to provide feedback on the results. We considered results marked as either *Correct* or *Incomplete Result* as correct queries for learning. The remaining incorrect utterances were sent to a crowd worker for annotation and were used to retrain the system for the next stage. The crowd worker had prior experience in writing SQL queries and was hired from Upwork after completing a short SQL test. The worker was also given access to the database to be able to execute the queries and ensure that they are correct. For the first stage, the system was trained using 640 examples generated using templates, that were augmented to 1746 examples using paraphrasing (see Section 4.3). The complexity of the utterances issued in each of the three phases were comparable, in that, the average length of the correct SQL query for the utterances, and the number of tables required to be queried, were similar.

Table 5 shows the percent of utterances judged by users as either *Correct* or *Incomplete Result* in each stage. In the first stage, we do not have any labeled examples, and the model is trained using only synthetically generated data from schema templates and paraphrases (see Section 4.3). Despite the lack of real examples, the system correctly answers 25% of questions. The system's accuracy increases and annotation effort decreases in each successive stage as additional utterances are contributed and incorrect utterances are labeled. This result demonstrates that we can successfully build semantic parsers for new domains by using neural models to generate SQL with crowdsourced annotations driven by user feedback.

We analyzed the feedback signals provided by the users in the final stage of the experiment to measure the quality of feedback. We found that 22.3% of the generated queries did not execute (and hence were incorrect). 6.1% of correctly generated queries were marked wrong by users (see Table 6). This erroneous feedback results in redundant annotation of already correct examples. The main cause of this erroneous feedback was incomplete data for aggregation queries, where users chose *Wrong* instead of *Incomplete*. 6.3% of incorrect queries were erroneously deemed correct by users. It is important that this fraction be low, as these queries become incorrectly-labeled exam-

	Stage 1	Stage 2	Stage 3
Accuracy (%)	25	53.7	63.5

Table 5: Percentage of utterances marked as *Correct* or *Incomplete* by users, in each stage of our online experiment.

	Feedback Error Rate (%)
Correct SQL	6.1
Incorrect SQL	6.3

Table 6: Error rates of user feedback when the SQL is correct and incorrect. The *Correct* and *Incomplete results* options are erroneous if the SQL query is correct, and vice versa for incorrect queries.

ples in the training set that may contribute to the deterioration of model accuracy over time. This quality of feedback is already sufficient for our neural models to improve with usage, and creating better interfaces to make feedback more accurate is an important task for future work.

6.3 SCHOLAR dataset

We release a new semantic parsing dataset for academic database search using the utterances gathered in the user study. We augment these labeled utterances with additional utterances labeled by crowd workers. (Note that these additional utterances were not used in the online experiment). The final dataset comprises 816 natural language utterances labeled with SQL, divided into a 600/216 train/test split. We also provide a database on which to execute these queries containing academic papers with their authors, citations, journals, keywords and datasets used. Table 1 shows statistics of this dataset. Our parser achieves an accuracy of 67% on this train/test split in the fully supervised setting. In comparison, a nearest neighbor strategy that uses the cosine similarity metric using a TF-IDF representation for the utterances yields an accuracy of 52.75%.

We found that 15% of the predicted queries did not execute, predominantly owing to (1) accessing table columns without joining with those tables, and (2) generating incorrect types that could not be deanonymized using the utterance. The main types of errors in the remaining well-formed queries that produced incorrect results were (1) portions of the utterance (such as 'top' and 'cited by both') were ignored, and (2) some types from the utterance were not transferred to the SQL query.

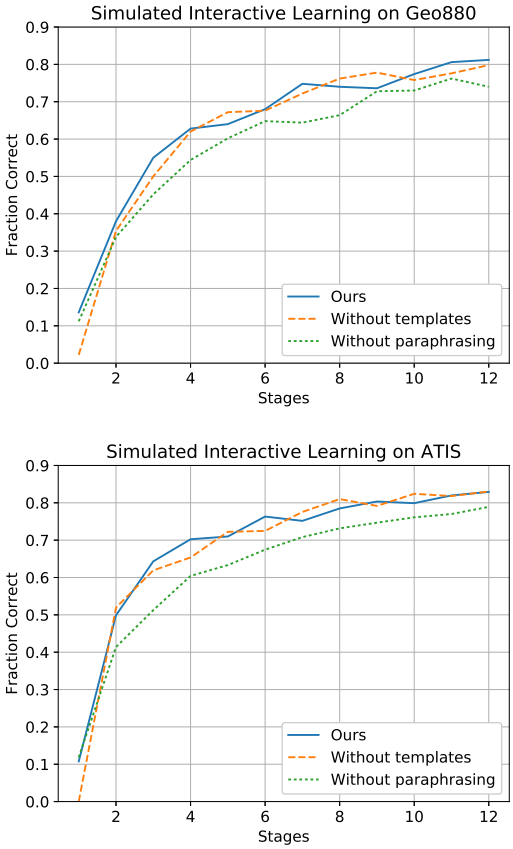


Figure 3: Accuracy as a function of batch number in simulated interactive learning experiments on Geo880 (top) and ATIS (bottom).

6.4 Simulated Interactive Experiments

We conducted additional simulated interactive learning experiments using GEO880 and ATIS to better understand the behavior of our train-deploy feedback loop, the effects of our data augmentation approaches, and the annotation effort required. We randomly divide each training set into K batches and present these batches sequentially to our interactive learning algorithm. Correctness feedback is provided by comparing the result of the predicted query to the gold query, i.e., we assume that users are able to perfectly distinguish correct results from incorrect ones.

Figure 3 shows accuracies on GEO880 and ATIS respectively of each batch when the model is trained on all previous batches. As in the live experiment, accuracy improves with successive batches. Data augmentation using templates helps in the initial stages of GEO880, but its advantage

Batch Size	150	100	50
% Wrong	70.2	60.4	54.3

Table 7: Percentage of examples that required annotation (i.e., where the model initially made an incorrect prediction) on GEO880 vs. batch size.

is reduced as more labeled data is obtained. Templates did not improve accuracy on ATIS, possibly because most ATIS queries involve two entities, i.e., a source city and a destination city, whereas our templates only generate questions with a single entity type. Nevertheless, templates are important in a live system to motivate users to interact with it in early stages. As observed before, paraphrasing improves performance at all stages.

Table 7 shows the percent of examples that require annotation using various batch sizes for GEO880. Smaller batch sizes reduce annotation effort, with a batch size of 50 requiring only 54.3% of the examples to be annotated. This result demonstrates that more frequent deployments of improved models leads to fewer mistakes.

7 Conclusion

We describe an approach to rapidly train a semantic parser as a NLIDB that iteratively improves parser accuracy over time while requiring minimal intervention. Our approach uses an attention-based neural sequence-to-sequence model, with data augmentation from the target database and paraphrasing, to parse utterances to SQL. This model is deployed in an online system, where user feedback on its predictions is used to select utterances to send for crowd worker annotation.

We find that the semantic parsing model is comparable in performance to previous systems that either map from utterances to logical forms, or generate SQL, on two benchmark datasets, GEO880 and ATIS. We further demonstrate the effectiveness of our online system by learning a semantic parser from scratch for an academic domain. A key advantage of our approach is that it is not language-specific, and can easily be ported to other commonly used query languages, such as SPARQL or ElasticSearch. Finally, we also release a new dataset of utterances and SQL queries for an academic domain.

Acknowledgments

The research was supported in part by DARPA, under the DEFT program through the AFRL (FA8750-13-2-0019), the ARO (W911NF-16-1-0121), the NSF (IIS-1252835, IIS-1562364, IIS-1546083, IIS-1651489, CNS-1563788), the DOE (DE-SC0016260), an Allen Distinguished Investigator Award, and gifts from NVIDIA, Adobe, and Google. The authors thank Rik Koncel-Kedziorski and the anonymous reviewers for their helpful comments.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1699–1710. https://doi.org/10.18653/v1/D15-1198.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* 1(1):49–62. http://aclweb.org/anthology/Q13-1005.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*. CBLS, San Diego, California. http://arxiv.org/abs/1409.0473.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1533–1544. http://aclweb.org/anthology/D13-1160.

Bo Chen, Le Sun, Xianpei Han, and Bo An. 2016. Sentence rewriting for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 766–777. http://www.aclweb.org/anthology/P16-1073.

Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. 2015. Scalable semantic parsing with partial ontologies. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1311–1320. https://doi.org/10.3115/v1/P15-1127.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 18–27. http://aclweb.org/anthology/W10-2903.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 33–43. https://doi.org/10.18653/v1/P16-1004.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 758–764. http://aclweb.org/anthology/N13-1092.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to SQL queries with generative parsers discriminatively reranked. In *Proceedings of COLING 2012: Posters*. The COLING 2012 Organizing Committee, pages 401–410. http://aclweb.org/anthology/C12-2040.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 12–22. https://doi.org/10.18653/v1/P16-1002.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 329–339. http://aclweb.org/anthology/D16-1032.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 754–765. http://aclweb.org/anthology/D12-1069.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 271–281. http://www.aclweb.org/anthology/P14-1026.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1545–1556. http://www.aclweb.org/anthology/D13-1161.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1512–1523. http://aclweb.org/anthology/D11-1140.

Percy Liang, I. Michael Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics* 39(2). https://doi.org/10.1162/COLI_a_00127.

Wang Ling, Phil Blunsom, Edward Grefenstette, Moritz Karl Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 599–609. https://doi.org/10.18653/v1/P16-1057.

Nicholas Locascio, Karthik Narasimhan, Eduardo De Leon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1918–1923. https://aclweb.org/anthology/D16-1197.

Thang Luong, Hieu Pham, and D. Christopher Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1412–1421. https://doi.org/10.18653/v1/D15-1166.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Kumar Dipendra Misra and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association

for Computational Linguistics, pages 1775–1786. http://aclweb.org/anthology/D16-1183.

V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*. pages 285–290. https://doi.org/10.1109/ICFHR.2014.55.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 933–943. http://aclweb.org/anthology/P13-1092.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, pages 149–157.

Raghu Ramakrishnan and Johannes Gehrke. 2003. *Database Management Systems*. McGraw-Hill, Inc., New York, NY, USA, 3 edition.

Subhro Roy, Shyam Upadhyay, and Dan Roth. 2016. Equation parsing : Mapping sentences to grounded equations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1088–1097. http://aclweb.org/anthology/D16-1117.

Adrienne Wang, Tom Kwiatkowski, and Luke Zettlemoyer. 2014. Morpho-syntactic lexical generalization for CCG semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1284–1295. https://doi.org/10.3115/v1/D14-1135.

I. Sida Wang, Percy Liang, and D. Christopher Manning. 2016. Learning language games through interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2368–2378. https://doi.org/10.18653/v1/P16-1224.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1332–1342. https://doi.org/10.3115/v1/P15-1129.

Wah Yuk Wong and Raymond Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Association for Computational Linguistics, pages 172–179. http://aclweb.org/anthology/N07-1022.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 201–206. https://doi.org/10.18653/v1/P16-2033.

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

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. http://aclweb.org/anthology/D07-1071.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.

Joint Modeling of Content and Discourse Relations in Dialogues

Kechen Qin[1] **Lu Wang**[1] **Joseph Kim**[2]
[1]College of Computer and Information Science, Northeastern University
[2]Computer Science and Artificial Intelligence Laboratory,
Massachusetts Institute of Technology
[1]qin.ke@husky.neu.edu, luwang@ccs.neu.edu
[2]joseph_kim@csail.mit.edu

Abstract

We present a joint modeling approach to identify salient discussion points in spoken meetings as well as to label the discourse relations between speaker turns. A variation of our model is also discussed when discourse relations are treated as latent variables. Experimental results on two popular meeting corpora show that our joint model can outperform state-of-the-art approaches for both phrase-based content selection and discourse relation prediction tasks. We also evaluate our model on predicting the consistency among team members' understanding of their group decisions. Classifiers trained with features constructed from our model achieve significant better predictive performance than the state-of-the-art.

1 Introduction

Goal-oriented dialogues, such as meetings, negotiations, or customer service transcripts, play an important role in our daily life. Automatically extracting the critical points and important outcomes from dialogues would facilitate generating summaries for complicated conversations, understanding the decision-making process of meetings, or analyzing the effectiveness of collaborations.

We are interested in a specific type of dialogues — spoken meetings, which is a common way for collaboration and idea sharing. Previous work (Kirschner et al., 2012) has shown that discourse structure can be used to capture the main discussion points and arguments put forward during problem-solving and decision-making processes in meetings. Indeed, content of different speaker turns do not occur in isolation, and should be interpreted within the context of discourse. Meanwhile, content can also reflect the purpose of speaker turns, thus facilitate with discourse relation understanding. Take the meeting snippet from

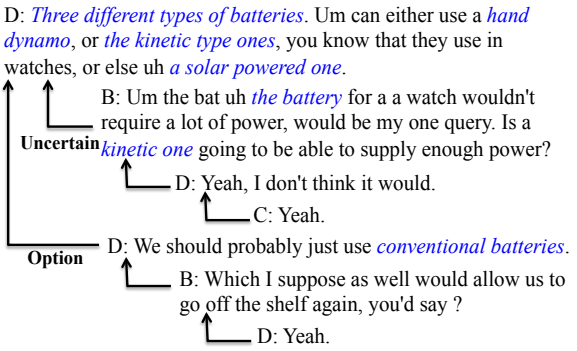


Figure 1: A sample clip from AMI meeting corpus. B, C, and D denotes different speakers. Here we highlight salient phrases (in *italics*) that are relevant to the major topic discussed, i.e., "which type of battery to use for the remote control". Arrows indicate discourse structure between speaker turns. We also show some of the discourse relations for illustration.

AMI corpus (Carletta et al., 2006) in Figure 1 as an example. This discussion is annotated with discourse structure based on the Twente Argumentation Schema (TAS) by Rienks et al. (2005), which focuses on argumentative discourse information. As can be seen, meeting participants evaluate different options by showing doubt (UNCERTAIN), bringing up alternative solution (OPTION), or giving feedback. The discourse information helps with the identification of the key discussion point, i.e., "which type of battery to use", by revealing the discussion flow.

To date, most efforts to leverage discourse information to detect salient content from dialogues have focused on encoding gold-standard discourse relations as features for use in classifier training (Murray et al., 2006; Galley, 2006; McKeown et al., 2007; Bui et al., 2009). However, automatic discourse parsing in dialogues is still a challenging problem (Perret et al., 2016). Moreover, acquiring human annotation on discourse relations is a time-consuming and expensive process, and does not

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 974–984
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1090

scale for large datasets.

In this paper, we propose *a joint modeling approach to select salient phrases reflecting key discussion points as well as label the discourse relations between speaker turns in spoken meetings*. We hypothesize that leveraging the interaction between content and discourse has the potential to yield better prediction performance on both *phrase-based content selection* and *discourse relation prediction*. Specifically, we utilize argumentative discourse relations as defined in Twente Argument Schema (TAS) (Rienks et al., 2005), where discussions are organized into tree structures with discourse relations labeled between nodes (as shown in Figure 1). Algorithms for joint learning and joint inference are proposed for our model. We also present a variation of our model to treat discourse relations as latent variables when true labels are not available for learning. We envision that the extracted salient phrases by our model can be used as input to abstractive meeting summarization systems (Wang and Cardie, 2013; Mehdad et al., 2014). Combined with the predicted discourse structure, a visualization tool can be exploited to display conversation flow to support intelligent meeting assistant systems.

To the best of our knowledge, our work is the first to jointly model content and discourse relations in meetings. We test our model with two meeting corpora — the AMI corpus (Carletta et al., 2006) and the ICSI corpus (Janin et al., 2003). Experimental results show that our model yields an accuracy of 63.2 on phrase selection, which is significantly better than a classifier based on Support Vector Machines (SVM). Our discourse prediction component also obtains better accuracy than a state-of-the-art neural network-based approach (59.2 vs. 54.2). Moreover, our model trained with latent discourse outperforms SVMs on both AMI and ICSI corpora for phrase selection. We further evaluate the usage of selected phrases as extractive meeting summaries. Results evaluated by ROUGE (Lin and Hovy, 2003) demonstrate that our system summaries obtain a ROUGE-SU4 F1 score of 21.3 on AMI corpus, which outperforms non-trivial extractive summarization baselines and a keyword selection algorithm proposed in Liu et al. (2009).

Moreover, since both content and discourse structure are critical for building shared understanding among participants (Mulder et al., 2002; Mercer, 2004), we further investigate whether our learned model can be utilized to predict the consistency among team members' understanding of their group decisions. This task is first defined as *consistency of understanding* (COU) prediction by Kim and Shah (2016), who have labeled a portion of AMI discussions with consistency or inconsistency labels. We construct features from our model predictions to capture different discourse patterns and word entrainment scores for discussion with different COU level. Results on AMI discussions show that SVM classifiers trained with our features significantly outperform the state-of-the-art results (Kim and Shah, 2016) (F1: 63.1 vs. 50.5) and non-trivial baselines.

The rest of the paper is structured as follows: we first summarize related work in Section 2. The joint model is presented in Section 3. Datasets and experimental setup are described in Section 4, which is followed by experimental results (Section 5). We then study the usage of our model for predicting consistency of understanding in groups in Section 6. We finally conclude in Section 7.

2 Related Work

Our model is inspired by research work that leverages discourse structure for identifying salient content in conversations, which is still largely reliant on features derived from gold-standard discourse labels (McKeown et al., 2007; Murray et al., 2010; Bokaei et al., 2016). For instance, adjacency pairs, which are paired utterances with question-answer or offer-accept relations, are found to frequently appear in meeting summaries together and thus are utilized to extract summary-worthy utterances by Galley (2006). There is much less work that jointly predicts the importance of content along with the discourse structure in dialogus. Oya and Carenini (2014) employs Dynamic Conditional Random Field to recognize sentences in email threads for use in summary as well as their dialogue acts. Only local discourse structures from adjacent utterances are considered. Our model is built on tree structures, which captures more global information.

Our work is also in line with keyphrase identification or phrase-based summarization for conversations. Due to the noisy nature of dialogues, recent work focuses on identifying summary-worthy phrases from meetings (Fernández et al., 2008; Riedhammer et al., 2010) or email threads (Loza

et al., 2014). For instance, Wang and Cardie (2012) treat the problem as an information extraction task, where summary-worthy content represented as indicator and argument pairs is identified by an unsupervised latent variable model. Our work also targets at detecting salient phrases from meetings, but focuses on the joint modeling of critical discussion points and discourse relations held between them.

For the area of discourse analysis in dialogues, a significant amount of work has been done in predicting local discourse structures, such as recognizing dialogue acts or social acts of adjacent utterances from phone conversations (Stolcke et al., 2000; Kalchbrenner and Blunsom, 2013; Ji et al., 2016), spoken meetings (Dielmann and Renals, 2008), or emails (Cohen et al., 2004). Although discourse information from non-adjacent turns has been studied in the context of online discussion forums (Ghosh et al., 2014) and meetings (Hakkani-Tur, 2009), none of them models the effect of discourse structure on content selection, which is a gap that this work fills in.

3 The Joint Model of Content and Discourse Relations

In this section, we first present our joint model in Section 3.1. The algorithms for learning and inference are described in Sections 3.2 and 3.3, followed by feature description (Section 3.4).

3.1 Model Description

Our proposed model learns to jointly perform phrase-based content selection and discourse relation prediction by making use of the interaction between the two sources of information. Assume that a meeting discussion is denoted as $\mathbf{x}$, where $\mathbf{x}$ consists of a sequence of discourse units $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$. Each discourse unit can be a complete speaker turn or a part of it. As demonstrated in Figure 1, a tree-structured discourse diagram is constructed for each discussion with each discourse unit x_i as a node of the tree. In this work, we consider the argumentative discourse structure by Twente Argument Schema (TAS) (Rienks et al., 2005). For each node x_i, it is attached to another node $x_{i'}$ ($i' < i$) in the discussion, and a discourse relation d_i is hold on the link $\langle x_i, x_{i'} \rangle$ (d_i is empty if x_i is the root). Let $\mathbf{t}$ denote the set of links $\langle x_i, x_{i'} \rangle$ in $\mathbf{x}$. Following previous work on discourse analysis in meetings (Rienks et al., 2005; Hakkani-Tur, 2009), we assume that the attachment structure between discourse units are given during both training and testing.

A set of candidate phrases are extracted from each discourse unit x_i, from which salient phrases that contain gist information will be identified. We obtain constituent and dependency parses for utterances using Stanford parser (Klein and Manning, 2003). We restrict eligible candidate to be a noun phrase (NP), verb phrase (VP), prepositional phrase (PP), or adjective phrase (ADJP) with at most 5 words, and its head word cannot be a stop word.[1] If a candidate is a parent of another candidate in the constituent parse tree, we will only keep the parent. We further merge a verb and a candidate noun phrase into one candidate if the later is the direct object or subject of the verb. For example, from utterance "let's use a rubber case as well as rubber buttons", we can identify candidates "use a rubber case" and "rubber buttons". For x_i, the set of candidate phrases are denoted as $c_i = \{c_{i,1}, c_{i,2}, \cdots, c_{i,m_i}\}$, where m_i is the number of candidates. $c_{i,j}$ takes a value of 1 if the corresponding candidate is selected as salient phrase; otherwise, $c_{i,j}$ is equal to 0. All candidate phrases in discussion $\mathbf{x}$ are represented as $\mathbf{c}$.

We then define a log-linear model with feature parameters $\mathbf{w}$ for the candidate phrases $\mathbf{c}$ and discourse relations $\mathbf{d}$ in $\mathbf{x}$ as:

$$
\begin{aligned}
p(\mathbf{c}, \mathbf{d} | \mathbf{x}, \mathbf{w}) &\propto \exp[\mathbf{w} \cdot \Phi(\mathbf{c}, \mathbf{d}, \mathbf{x})] \\
&\propto \exp[\mathbf{w} \cdot \sum_{i=1, <x_i, x_{i'}> \in \mathbf{t}}^{n} \phi(c_i, d_i, d_{i'}, \mathbf{x})] \\
&\propto \exp[\sum_{i=1, <x_i, x_{i'}> \in \mathbf{t}}^{n} (\mathbf{w_c} \cdot \sum_{j=1}^{m_i} \phi_c(c_{i,j}, \mathbf{x}) \\
&+ \mathbf{w_d} \cdot \phi_d(d_i, d_{i'}, \mathbf{x}) + \mathbf{w_{cd}} \cdot \sum_{j=1}^{m_i} \phi_{cd}(c_{i,j}, d_i, \mathbf{x}))]
\end{aligned}
\tag{1}
$$

Here $\Phi(\cdot)$ and $\phi(\cdot)$ denote feature vectors. We utilize three types of feature functions: (1) content-only features $\phi_c(\cdot)$, which capture the importance of phrases, (2) discourse-only features $\phi_d(\cdot)$, which characterize the (potentially higher-order) discourse relations, and (3) joint features of content and discourse $\phi_{cd}(\cdot)$, which model the interaction between the two. $\mathbf{w_c}$, $\mathbf{w_d}$, and $\mathbf{w_{cd}}$ are

[1] Other methods for mining candidate phrases, such as frequency-based method (Liu et al., 2015), will be studied for future work.

corresponding feature parameters. Detailed feature descriptions can be found in Section 3.4.

Discourse Relations as Latent Variables. As we mentioned in the introduction, acquiring labeled training data for discourse relations is a time-consuming process since it would require human annotators to inspect the full discussions. Therefore, we further propose a variation of our model where it treats the discourse relations as latent variables, so that $p(\mathbf{c}|\mathbf{x},\mathbf{w}) = \sum_{\mathbf{d}} p(\mathbf{c},\mathbf{d}|\mathbf{x},\mathbf{w})$. Its learning algorithm is slightly different as described in the next section.

3.2 Joint Learning for Parameter Estimation

For learning the model parameters $\mathbf{w}$, we employ an algorithm based on SampleRank (Rohanimanesh et al., 2011), which is a stochastic structure learning method. In general, the learning algorithm constructs a sequence of configurations for sample labels as a Markov chain Monte Carlo (MCMC) chain based on a task-specific loss function, where stochastic gradients are distributed across the chain.

The full learning procedure is described in Algorithm 1. To start with, the feature weights $\mathbf{w}$ is initialized with each value randomly drawn from $[-1, 1]$. Multiple epochs are run through all samples. For each sample, we randomly initialize the assignment of candidate phrases labels $\mathbf{c}$ and discourse relations $\mathbf{d}$. Then an MCMC chain is constructed with a series of configurations $\sigma = (\mathbf{c}, \mathbf{d})$: at each step, it first samples a discourse structure $\mathbf{d}$ based on the proposal distribution $q(\mathbf{d}'|\mathbf{d},\mathbf{x})$, and then samples phrase labels conditional on the new discourse relations and previous phrase labels based on $q(\mathbf{c}'|\mathbf{c},\mathbf{d}',\mathbf{x})$. Local search is used for both proposal distributions.[2] The new configuration is accepted if it improves on the score by $\omega(\sigma')$. The parameters $\mathbf{w}$ are updated accordingly.

For the scorer ω, we use a weighted combination of F1 scores of phrase selection ($F1_c$) and discourse relation prediction ($F1_d$): $\omega(\sigma) = \alpha \cdot F1_c + (1-\alpha) \cdot F1_d$. We fix α to 0.1.

When discourse relations are treated as latent, we initialize discourse relations for each sample with a label in $\{1, 2, \ldots, K\}$ if there are K relations indicated, and we only use $F1_c$ as the scorer.

[2]For future work, we can explore other proposal distributions that utilize the conditional distribution of salient phrases given sampled discourse relations.

Input : $\mathbf{X} = \{\mathbf{x}\}$: discussions in the training set,
η: learning rate, ϵ: number of epochs,
δ: number of sampling rounds,
$\omega(\cdot)$: scoring function, $\Phi(\cdot)$: feature functions
Output: feature weights $\frac{1}{|\mathcal{W}|}\sum_{\mathbf{w}\in\mathcal{W}} \mathbf{w}$

Initialize $\mathbf{w}$;
$\mathcal{W} \leftarrow \{\mathbf{w}\}$;
for $e = 1$ *to* ϵ **do**
  for $\mathbf{x}$ *in* $\mathbf{X}$ **do**
    // Initialize configuration for $\mathbf{x}$
    Initialize $\mathbf{c}$ and $\mathbf{d}$;
    $\sigma = (\mathbf{c}, \mathbf{d})$;
    for $s = 1$ *to* δ **do**
      // New configuration via local search
      $\mathbf{d}' \sim q_d(\cdot|\mathbf{x},\mathbf{d})$;
      $\mathbf{c}' \sim q_d(\cdot|\mathbf{x},\mathbf{c},\mathbf{d}')$;
      $\sigma' = (\mathbf{c}', \mathbf{d}')$;
      $\sigma^+ = \arg\max_{\tilde{\sigma}\in\{\sigma,\sigma'\}} \omega(\tilde{\sigma})$;
      $\sigma^- = \arg\min_{\tilde{\sigma}\in\{\sigma,\sigma'\}} \omega(\tilde{\sigma})$;
      $\hat{\nabla} = \Phi(\sigma^+) - \Phi(\sigma^-)$;
      $\Delta\omega = \omega(\sigma^+) - \omega(\sigma^-)$;
      // Update parameters
      if $\mathbf{w}\cdot\hat{\nabla} < \Delta\omega$ & $\Delta\omega \neq 0$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot \hat{\nabla}$;
        Add $\mathbf{w}$ in $\mathcal{W}$;
      end
      // Accept or reject new configuration
      if $\sigma^+ == \sigma'$ **then**
        $\sigma = \sigma'$
      end
    end
  end
end

Algorithm 1: SampleRank-based joint learning.

3.3 Joint Inference for Prediction

Given a new sample $\mathbf{x}$ and learned parameters $\mathbf{w}$, we predict phrase labels and discourse relations as $\arg\max_{\mathbf{c},\mathbf{d}} p(\mathbf{c},\mathbf{d}|\mathbf{x},\mathbf{w})$.

Dynamic programming can be employed to carry out joint inference, however, it would be time-consuming since our objective function has a large search space for both content and discourse labels. Hence we propose an alternating optimizing algorithm to search for $\mathbf{c}$ and $\mathbf{d}$ iteratively. Concretely, for each iteration, we first optimize on $\mathbf{d}$ by maximizing $\sum_{i=1,<x_i,x_i'>\in\mathbf{t}}^{n}(\mathbf{w_d}\cdot\phi_d(d_i,d_{i'},\mathbf{x}) + \mathbf{w_{cd}}\cdot\sum_{j=1}^{m_i}\phi_{cd}(c_{i,j},d_i,\mathbf{x}))$. Message-passing (Smith and Eisner, 2008) is used to find the best $\mathbf{d}$.

In the second step, we search for $\mathbf{c}$ that maximizes $\sum_{i=1,<x_i,x_i'>\in\mathbf{t}}^{n}(\mathbf{w_c}\cdot\sum_{j=1}^{m_i}\phi_c(c_{i,j},\mathbf{x}) + \mathbf{w_{cd}}\cdot\sum_{j=1}^{m_i}\phi_{cd}(c_{i,j},d_i,\mathbf{x}))$. We believe that candidate phrases based on the same concepts should have the same predicted label. Therefore, candidates of the same phrase type and sharing the same head word are grouped into one cluster. We then cast our task as an integer linear programming

problem.[3] We optimize our objective function under constraints: (1) $c_{i,j} = c_{i',j'}$ if $c_{i,j}$ and $c_{i',j'}$ are in the same cluster, and (2) $c_{i,j} \in \{0, 1\}, \forall i, j$.

The inference process is the same for models trained with latent discourse relations.

3.4 Features

We use features that characterize content, discourse relations, and the combination of both.

Content Features. For modeling the salience of content, we calculate the minimum, maximum, and average of `TF-IDF` scores of words and `number of content words` in each phrase based on the intuition that important phrases tend to have more content words with high TF-IDF scores (Fernández et al., 2008). We also consider whether the head word of the phrase has been `mentioned in preceding turn`, which implies the focus of a discussion. The `size of the cluster` each phrase belongs to is also included. `Number of POS tags` and `phrase types` are counted to characterize the syntactic structure. Previous work (Wang and Cardie, 2012) has found that a discussion usually ends with decision-relevant information. We thus identify the `absolute and relative positions` of the turn containing the candidate phrase in the discussion. Finally, we record whether the candidate phrase is `uttered by the main speaker`, who speakers the most words in the discussion.

Discourse Features. For each discourse unit, we collect the `dialogue act types` of the current unit and its parent node in discourse tree, whether there is any `adjacency pair` held between the two nodes (Hakkani-Tur, 2009), and the `Jaccard similarity` between them. We record whether two turns are `uttered by the same speaker`, for example, ELABORATION is commonly observed between the turns from the same participant. We also calculate the `number of candidate phrases` based on the observation that OPTION and SPECIALIZATION tend to contain more informative words than POSITIVE feedback. Length of the discourse unit is also relevant. Therefore, we compute the `time span` and `number of words`. To incorporate global structure features, we encode the `depth of the node` in the discourse tree and the `number of its siblings`. Finally, we include an `order-2 discourse relation` feature that encodes the relation between current discourse unit and its parent, and the relation between the parent and its grandparent if it exists.

Joint Features. For modeling the interaction between content and discourse, the discourse relation is added to each content feature to compose a joint feature. For example, if candidate c in discussion x has a content feature $\phi_{[avg-TFIDF]}(c, \mathbf{x})$ with a value of 0.5, and its discourse relation d is POSITIVE, then the joint feature takes the form of $\phi_{[avg-TFIDF,Positive]}(c, d, \mathbf{x}) = 0.5$.

4 Datasets and Experimental Setup

Meeting Corpora. We evaluate our joint model on two meeting corpora with rich annotations: the AMI meeting corpus (Carletta et al., 2006) and the ICSI meeting corpus (Janin et al., 2003). AMI corpus consists of 139 scenario-driven meetings, and ICSI corpus contains 75 naturally occurring meetings. Both of the corpora are annotated with dialogue acts, adjacency pairs, and topic segmentation. We treat each topic segment as one discussion, and remove discussions with less than 10 turns or labeled as "opening" and "chitchat". 694 discussions from AMI and 1139 discussions from ICSI are extracted, and these two datasets are henceforth referred as AMI-FULL and ICSI-FULL.

Acquiring Gold-Standard Labels. Both corpora contain human constructed abstractive summaries and extractive summaries on meeting level. Short abstracts, usually in one sentence, are constructed by meeting participants — *participant summaries*, and external annotators — *abstractive summaries*. Dialogue acts that contribute to important output of the meeting, e.g. decisions, are identified and used as extractive summaries, and some of them are also linked to the corresponding abstracts.

Since the corpora do not contain phrase-level importance annotation, we induce gold-standard labels for candidate phrases based on the following rule. A candidate phrase is considered as a positive sample if its head word is contained in any abstractive summary or participant summary. On average, 71.9 candidate phrases are identified per discussion for AMI-FULL with 31.3% labeled as positive, and 73.4 for ICSI-FULL with 24.0% of them as positive samples.

Furthermore, a subset of discussions in AMI-

[3] We use lpsolve: `http://lpsolve.sourceforge.net/5.5/`.

FULL are annotated with discourse structure and relations based on Twente Argumentation Schema (TAS) by Rienks et al. (2005)[4]. A tree-structured argument diagram (as shown in Figure 1) is created for each discussion or a part of the discussion. The nodes of the tree contain partial or complete speaker turns, and discourse relation types are labeled on the links between the nodes. In total, we have 129 discussions annotated with discourse labels. This dataset is called AMI-SUB hereafter.

Experimental Setup. 5-fold cross validation is used for all experiments. All real-valued features are uniformly normalized to [0,1]. For the joint learning algorithm, we use 10 epochs and carry out 50 sampling for MCMC for each training sample. The learning rate is set to 0.01. We run the learning algorithm for 20 times, and use the average of the learned weights as the final parameter values. For models trained with latent discourse relations, we fix the number of relations to 9.

Baselines and Comparisons. For both phrase-based content selection and discourse relation prediction tasks, we consider a baseline that always predicts the majority label (Majority). Previous work has shown that Support Vector Machines (SVMs)-based classifiers achieve state-of-the-art performance for keyphrase selection in meetings (Fernández et al., 2008; Wang and Cardie, 2013) and discourse parsing for formal text (Hernault et al., 2010). Therefore, we compare with linear SVM-based classifiers, trained with the same feature set of content features or discourse features. We fix the trade-off parameter to 1.0 for all SVM-based experiments. For discourse relation prediction, we use one-vs-rest strategy to build multiple binary classifiers.[5] We also compare with a state-of-the-art discourse parser (Ji et al., 2016), which employs neural language model to predict discourse relations.

5 Experimental Results

5.1 Phrase Selection and Discourse Labeling

Here we present the experimental results on phrase-based content selection and discourse relation prediction. We experiment with two variations of our joint model: one is trained on gold-standard discourse relations, the other is trained by treating discourse relations as latent models as described in Section 3.1. Remember that we have gold-standard argument diagrams on the AMI-SUB dataset, we can thus conduct experiments by assuming the *True Attachment Structure* is given for latent versions. When argument diagrams are not available, we build a tree among the turns in each discussion as follows. Two turns are attached if there is any adjacency pair between them. If one turn is attached to more than one previous turns, the closest one is considered. For the rest of the turns, they are attached to the preceding turn. This construction is applied on AMI-FULL and ICSI-FULL.

We also investigate whether joint learning and joint inference can produce better prediction per-

	Acc	F1
Comparisons		
Baseline (Majority)	60.1	37.5
SVM (w content features in § 3.4)	57.8	54.6
Our Models		
Joint-Learn + Joint-Inference	**63.2**∗	**62.6**∗
Joint-Learn + Separate-Inference	57.9	57.8
Separate-Learn	53.4	52.6
Our Models (Latent Discourse)		
w/ True Attachment Structure		
Joint-Learn + Joint-Inference	60.3∗	60.3∗
Joint-Learn + Separate-Inference	56.4	56.2
w/o True Attachment Structure		
Joint-Learn + Joint-Inference	56.4	56.4
Joint-Learn + Separate-Inference	52.7	52.3

Table 1: Phrase-based content selection performance on AMI-SUB with accuracy (acc) and F1. We display results of our models trained with gold-standard discourse relation labels and with latent discourse relations. For the later, we also show results based on *True Attachment Structure*, where the gold-standard attachments are known, and without the *True Attachment Structure*. Our models that significantly outperform SVM-based model are highlighted with $*$ ($p < 0.05$, paired t-test). Best result for each column is in **bold**.

	Acc	F1
Comparisons		
Baseline (Majority)	51.2	7.5
SVM (w discourse features in § 3.4)	51.2	22.8
Ji et al. (2016)	54.2	21.4
Our Models		
Joint-Learn + Joint-Inference	58.0∗	21.7
Joint-Learn + Separate-Inference	**59.2**∗	23.4
Separate-Learn	58.2∗	**25.1**

Table 2: Discourse relation prediction performance on AMI-SUB. Our models that significantly outperform SVM-based model and Ji et al. (2016) are highlighted with $*$ ($p < 0.05$, paired t-test). Best result for each column is in **bold**.

[4]There are 9 types of relations in TAS: POSITIVE, NEGATIVE, UNCERTAIN, REQUEST, SPECIALIZATION, ELABORATION, OPTION, OPTION EXCLUSION, and SUBJECT-TO.

[5]Multi-class classifier was also experimented with, but gave inferior performance.

	AMI-FULL		ICSI-FULL	
	Acc	**F1**	**Acc**	**F1**
Comparisons				
Baseline (Majority)	61.8	38.2	**75.3**	43.0
SVM (with content features in § 3.4)	58.6	56.7	66.2	53.1
Our Models (Latent Discourse)				
Joint-Learn + Joint-Inference	**63.4**∗	**63.0**∗	73.5∗	61.4∗
Joint-Learn + Separate-Inference	57.7	57.5	70.0∗	**62.7**∗

Table 3: Phrase-based content selection performance on AMI-FULL and ICSI-FULL. We display results of our models trained with latent discourse relations. Results that are significantly better than SVM-based model are highlighted with ∗ ($p < 0.05$, paired t-test).

formance. We consider joint learning with separate inference, where only content features or discourse features are used for prediction (Separate-Inference). We further study learning separate classifiers for content selection and discourse relations without joint features (Separate-Learn).

We first show the phrase selection and discourse relation prediction results on AMI-SUB in Tables 1 and 2. As shown in Table 1, our models, trained with gold-standard discourse relations or latent ones with true attachment structure, yield significant better accuracy and F1 scores than SVM-based classifiers trained with the same feature sets for phrase selection (paired t-test, $p < 0.05$). Our joint learning model with separate inference also outperforms neural network-based discourse parsing model (Ji et al., 2016) in Table 2.

Moreover, Tables 1 and 2 demonstrate that joint learning usually produces superior performance for both tasks than separate learning. Combined with joint inference, our model obtains the best accuracy and F1 on phrase selection. This indicates that leveraging the interplay between content and discourse boost the prediction performance. Similar results are achieved on AMI-FULL and ICSI-FULL in Table 3, where latent discourse relations without true attachment structure are employed for training.

5.2 Phrase-Based Extractive Summarization

We further evaluate whether the prediction of the content selection component can be used for summarizing the key points on discussion level. For each discussion, salient phrases identified by our model are concatenated in sequence for use as the summary. We consider two types of gold-standard summaries. One is utterance-level extractive summary, which consists of human labeled summary-worthy utterances. The other is abstractive summary, where we collect human abstract with at least one link from summary-worthy utterances.

Extractive Summaries as Gold-Standard							
		ROUGE-1			ROUGE-SU4		
	Len	Prec	Rec	F1	Prec	Rec	F1
Longest DA	30.9	64.4	15.0	23.1	58.6	9.3	15.3
Centroid DA	17.5	**73.9**	13.4	20.8	**62.5**	6.9	11.3
SVM	49.8	47.1	24.1	27.5	22.7	10.7	11.8
Liu et al. (2009)	62.4	40.4	39.2	36.2	15.5	15.2	13.5
Our Model	66.6	45.4	44.7	41.1∗	24.1∗	23.4∗	20.9∗
Our Model-latent	85.9	42.9	**49.3**	**42.4**∗	21.6	**25.7**∗	**21.3**∗
Abstractive Summaries as Gold-Standard							
		ROUGE1			ROUGE-SU4		
	Len	Prec	Rec	F1	Prec	Rec	F1
Longest DA	30.9	14.8	5.5	7.4	4.8	1.4	1.9
Centroid DA	17.5	**24.9**	5.6	8.5	**11.6**	1.4	2.2
SVM	49.8	13.3	9.7	9.5	4.4	2.4	2.4
Liu et al. (2009)	62.4	10.3	16.7	11.3	2.7	4.5	2.8
Our Model	66.6	12.6	18.9	**13.1**∗	3.8	5.5∗	**3.7**∗
Our Model-latent	85.9	11.4	**20.0**	12.4∗	3.3	**6.1**∗	3.5∗

Table 4: ROUGE scores for phrase-based extractive summarization evaluated against human-constructed utterance-level extractive summaries and abstractive summaries. Our models that statistically significantly outperform SVM and Liu et al. (2009) are highlighted with ∗ ($p < 0.05$, paired t-test). Best ROUGE score for each column is in **bold**.

We calculate scores based on ROUGE (Lin and Hovy, 2003), which is a popular tool for evaluating text summarization (Gillick et al., 2009; Liu and Liu, 2010). ROUGE-1 (unigrams) and ROUGE-SU4 (skip-bigrams with at most 4 words in between) are used. Following previous work on meeting summarization (Riedhammer et al., 2010; Wang and Cardie, 2013), we consider two dialogue act-level summarization baselines: (1) LONGEST DA in each discussion is selected as the summary, and (2) CENTROID DA, the one with the highest TF-IDF similarity with all DAs in the discussion. We also compare with an unsupervised keyword extraction approach by Liu et al. (2009), where word importance is estimated by its TF-IDF score, POS tag, and the salience of its corresponding sentence. With the same candidate phrases as in our model, we extend Liu et al. (2009) by scoring each phrase based on its average score of the words. Top phrases, with the same number of phrases output by our model, are included into the summaries. Finally, we compare with summaries consisting of salient phrases predicted by an SVM classifier trained with our content features.

From the results in Table 4, we can see that phrase-based extractive summarization methods can yield better ROUGE scores for recall and F1 than baselines that extract the whole sentences. Meanwhile, our system significantly out-

Meeting Clip:
D: can we uh power a light in this? can we get a strong enough battery to power a light?
A: um i think we could because the lcd panel requires power, and the lcd is a form of a light so that. . .
D: . . .it's gonna have to have something high-tech about it and that's gonna take battery power. . .
D: illuminate the buttons. yeah it glows.
D: well m i'm thinking along the lines of you're you're in the dark watching a dvd and you um you find the thing in the dark and you go like this . . . oh where's the volume button in the dark, and uh y you just touch it . . . and it lights up or something.

Abstract by Human:
What sort of battery to use. The industrial designer presented options for materials, components, and batteries and discussed the restrictions involved in using certain materials.

Longest DA:
well m i'm thinking along the lines of you're you're in the dark watching a dvd and you um you find the thing in the dark and you go like this.
Centroid DA:
can we uh power a light in this?
Our Method:
- power a light, a strong enough battery,
- requires power, a form,
- a really good battery, battery power,
- illuminate the buttons, glows,
- watching a dvd, the volume button, lights up or something

Figure 2: Sample summaries output by different systems for a meeting clip from AMI corpus (less relevant utterances in between are removed). Salient phrases by our system output are displayed for each turn of the clip, with duplicated phrases removed for brevity.

performs the SVM-based classifiers when evaluated on ROUGE recall and F1, while achieving comparable precision. Compared to Liu et al. (2009), our system also yields better results on all metrics.

Sample summaries by our model along with two baselines are displayed in Figure 2. Utterance-level extract-based baselines unavoidably contain disfluency and unnecessary details. Our phrase-based extractive summary is able to capture the key points from both the argumentation process and important outcomes of the conversation. This implies that our model output can be used as input for an abstractive summarization system. It can also facilitate the visualization of decision-making processes.

5.3 Further Analysis and Discussions

Features Analysis. We first discuss salient features with top weights learned by our joint model. For content features, main speaker tends to utter more salient content. Higher TF-IDF scores also indicate important phrases. If a phrase is mentioned in previous turn and repeated in the current turn, it is likely to be a key point. For discourse features, structure features matter the most. For instance, jointly modeling the discourse relation of the parent node along with the current node can lead to better inference. An example is that giving more details on the proposal (ELABORATION) tends to lead to POSITIVE feedback. Moreover, REQUEST usually appears close to the root of the argument diagram tree, while POSITIVE feedback is usually observed on leaves. Adjacency pairs also play an important role for discourse prediction. For joint features, features that composite "phrase mentioned in previous turn" and relation POSITIVE feedback or REQUEST yield higher weight, which are indicators for both key phrases and discourse relations. We also find that main speaker information composite with ELABORATION and UNCERTAIN are associated with high weights.

Error Analysis and Potential Directions. Taking a closer look at our prediction results, one major source of incorrect prediction for phrase selection is based on the fact that similar concepts might be expressed in different ways, and our model predicts inconsistently for different variations. For example, participants use both "thick" and "two centimeters" to talk about the desired shape of a remote control. However, our model does not group them into the same cluster and later makes different predictions. For future work, semantic similarity with context information can be leveraged to produce better clustering results. Furthermore, identifying discourse relations in dialogues is still a challenging task. For instance, "I wouldn't choose a plastic case" should be labeled as OPTION EXCLUSION, if the previous turns talk about different options. Otherwise, it can be labeled as NEGATIVE. Therefore, models that better handle semantics and context need to be considered.

6 Predicting Consistency of Understanding

As discussed in previous work (Mulder et al., 2002; Mercer, 2004), both content and discourse structure are critical for building shared understanding among discussants. In this section, we test whether our joint model can be utilized to predict the consistency among team members' under-

standing of their group decisions, which is defined as consistency of understanding (COU) in Kim and Shah (2016).

Kim and Shah (2016) establish gold-standard COU labels on a portion of AMI discussions, by comparing participant summaries to determine whether participants report the same decisions. If all decision points are consistent, the associated topic discussion is labeled as *consistent*; otherwise, the discussion is identified as *inconsistent*. Their annotation covers the AMI-SUB dataset. Therefore, we run the prediction experiments on AMI-SUB by using the same annotation. Out of total 129 discussions in AMI-SUB, 86 discussions are labeled as consistent and 43 are inconsistent.

We construct three types of features by using our model's predicted labels. Firstly, we learn two versions of our model based on the "consistent" discussions and the "inconsistent" ones in the training set, with learned parameters $\mathbf{w_{con}}$ and $\mathbf{w_{incon}}$. For a discussion in the test set, these two models output two probabilities $p_{con} = \max_{\mathbf{c},\mathbf{d}} P(\mathbf{c}, \mathbf{d}|\mathbf{x}, \mathbf{w_{con}})$ and $p_{incon} = \max_{\mathbf{c},\mathbf{d}} P(\mathbf{c}, \mathbf{d}|\mathbf{x}, \mathbf{w_{incon}})$. We use $p_{con} - p_{incon}$ as a feature.

Furthermore, we consider discourse relations of length one and two from the discourse structure tree. Intuitively, some discourse relations, e.g., ELABORATION followed by multiple POSITIVE feedback, imply consistent understanding.

The third feature is based on word entrainment, which has been shown to correlate with task success for groups (Nenkova et al., 2008). Using the formula in Nenkova et al. (2008), we compute the average word entrainment between the main speaker who utters the most words and all the other participants. The content words in the salient phrases predicted by our model is considered for entrainment computation.

Results. Leave-one-out is used for experiments. For training, our features are constructed from gold-standard phrase and discourse labels. Predicted labels by our model is used for constructing features during testing. SVM-based classifier is used for experimenting with different sets of features output by our model. A majority class baseline is constructed as well. We also consider an SVM classifier trained with ngram features (unigrams and bigrams). Finally, we compare with the state-of-the-art method in Kim and Shah (2016), where discourse-relevant features and head gesture features are utilized in Hidden Markov Models to predict the consistency label.

	Acc	F1
Comparisons		
Baseline (Majority)	66.7	40.0
Ngrams (SVM)	51.2	50.6
Kim and Shah (2016)	60.5	50.5
Features from Our Model		
Consistency Probability (Prob)	52.7	52.1
Discourse Relation (Disc)	63.6	57.1∗
Word Entrainment (Ent)	60.5∗	57.1∗
Prob + Disc+ Ent	**68.2**∗	**63.1**∗
Oracles		
Discourse Relation	69.8	62.7
Word Entrainment	61.2	57.8

Table 5: Consistency of Understanding (COU) prediction results on AMI-SUB. Results that statistically significantly outperform ngrams-based baseline and Kim and Shah (2016) are highlighted with $*$ ($p < 0.05$, paired t-test). For reference, we also show the prediction performance based on gold-standard discourse relations and phrase selection labels.

The results are displayed in Table 5. All SVMs trained with our features surpass the ngrams-based baseline. Especially, the discourse features, word entrainment feature, and the combination of the three, all significantly outperform the state-of-the-art system by Kim and Shah (2016).[6]

7 Conclusion

We presented a joint model for performing phrase-level content selection and discourse relation prediction in spoken meetings. Experimental results on AMI and ICSI meeting corpora showed that our model can outperform state-of-the-art methods for both tasks. Further evaluation on the task of predicting consistency-of-understanding in meetings demonstrated that classifiers trained with features constructed from our model output produced superior performance compared to the state-of-the-art model. This provides an evidence of our model being successfully applied in other prediction tasks in spoken meetings.

Acknowledgments

This work was supported in part by National Science Foundation Grant IIS-1566382 and a GPU gift from Nvidia. We thank three anonymous reviewers for their valuable suggestions on various aspects of this work.

[6]We also experiment with other popular classifiers, e.g. logistic regression or decision tree, and similar trend is respected.

References

Mohammad Hadi Bokaei, Hossein Sameti, and Yang Liu. 2016. Extractive Summarization of Multi-party Meetings Through Discourse Segmentation. *Natural Language Engineering* 22(01):41–72.

Trung H. Bui, Matthew Frampton, John Dowding, and Stanley Peters. 2009. Extracting Decisions from Multi-party Dialogue Using Directed Graphical Models and Semantic Similarity. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Stroudsburg, PA, USA, SIGDIAL '09, pages 235–243.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska, Iain McCowan, Wilfried Post, Dennis Reidsma, and Pierre Wellner. 2006. The AMI Meeting Corpus: A Pre-announcement. In *Proceedings of the Second International Conference on Machine Learning for Multimodal Interaction*. Springer-Verlag, Berlin, Heidelberg, MLMI'05, pages 28–39.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to Classify Email into "Speech Acts" . In Dekang Lin and Dekai Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Barcelona, Spain, pages 309–316.

Alfred Dielmann and Steve Renals. 2008. Recognition of Dialogue Acts in Multiparty Meetings Using a Switching DBN. *IEEE transactions on audio, speech, and language processing* 16(7):1303–1314.

Raquel Fernández, Matthew Frampton, John Dowding, Anish Adukuzhiyil, Patrick Ehlen, and Stanley Peters. 2008. Identifying Relevant Phrases to Summarize Decisions in Spoken Meetings. In *INTERSPEECH*. pages 78–81.

Michel Galley. 2006. A Skip-chain Conditional Random Field for Ranking Meeting Utterances by Importance. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '06, pages 364–372.

Debanjan Ghosh, Smaranda Muresan, Nina Wacholder, Mark Aakhus, and Matthew Mitsui. 2014. Analyzing Argumentative Discourse Units in Online Interactions. In *Proceedings of the First Workshop on Argumentation Mining*. pages 39–48.

Dan Gillick, Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tur. 2009. A Global Optimization Framework for Meeting Summarization. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, pages 4769–4772.

Dilek Hakkani-Tur. 2009. Towards Automatic Argument Diagramming of Multiparity Meetings. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, pages 4753–4756.

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

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. 2003. The ICSI Meeting Corpus. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*. IEEE, volume 1, pages I–I.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A Latent Variable Recurrent Neural Network for Discourse-Driven Language Models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 332–342.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Convolutional Neural Networks for Discourse Compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*. Association for Computational Linguistics, Sofia, Bulgaria, pages 119–126.

Joseph Kim and Julie A Shah. 2016. Improving Team's Consistency of Understanding in Meetings. *IEEE Transactions on Human-Machine Systems* 46(5):625–637.

Paul A Kirschner, Simon J Buckingham-Shum, and Chad S Carr. 2012. *Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-making*. Springer Science & Business Media.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '03, pages 423–430.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. pages 71–78.

Fei Liu and Yang Liu. 2010. Using Spoken Utterance Compression for Meeting Summarization: A Pilot Study. In *Spoken Language Technology Workshop (SLT), 2010 IEEE*. IEEE, pages 37–42.

Feifan Liu, Deana Pennell, Fei Liu, and Yang Liu. 2009. Unsupervised Approaches for Automatic Keyword Extraction Using Meeting Transcripts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Boulder, Colorado, pages 620–628.

Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, pages 1729–1744.

Vanessa Loza, Shibamouli Lahiri, Rada Mihalcea, and Po-Hsiang Lai. 2014. Building a Dataset for Summarization and Keyword Extraction from Emails. In *LREC*. pages 2441–2446.

Kathleen McKeown, Lokesh Shrestha, and Owen Rambow. 2007. Using Question-answer Pairs in Extractive Summarization of Email Conversations. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, pages 542–550.

Yashar Mehdad, Giuseppe Carenini, and Raymond T. Ng. 2014. Abstractive Summarization of Spoken and Written Conversations Based on Phrasal Queries. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1220–1230.

Neil Mercer. 2004. Sociocultural Discourse Analysis. *Journal of applied linguistics* 1(2):137–168.

Ingrid Mulder, Janine Swaak, and Joseph Kessels. 2002. Assessing Group Learning and Shared Understanding in Technology-mediated Interaction. *Educational Technology & Society* 5(1):35–47.

Gabriel Murray, Giuseppe Carenini, and Raymond Ng. 2010. Generating and Validating Abstracts of Meeting Conversations: A User Study. In *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics, Stroudsburg, PA, USA, INLG '10, pages 105–113.

Gabriel Murray, Steve Renals, Jean Carletta, and Johanna Moore. 2006. Incorporating Speaker and Discourse Features into Speech Summarization. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, pages 367–374.

Ani Nenkova, Agustin Gravano, and Julia Hirschberg. 2008. High Frequency Word Entrainment in Spoken Dialogue. In *Proceedings of the 46th annual meeting of the association for computational linguistics on human language technologies: Short papers*. Association for Computational Linguistics, pages 169–172.

Tatsuro Oya and Giuseppe Carenini. 2014. Extractive Summarization and Dialogue Act Modeling on Email Threads: An Integrated Probabilistic Approach. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 133.

Jérémy Perret, Stergos Afantenos, Nicholas Asher, and Mathieu Morey. 2016. Integer Linear Programming for Discourse Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 99–109.

Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tür. 2010. Long Story Short - Global Unsupervised Models for Keyphrase Based Meeting Summarization. *Speech Commun.* 52(10):801–815.

Rutger Rienks, Dirk Heylen, and E. van der Weijden. 2005. Argument Diagramming of Meeting Conversations. In A. Vinciarelli and J-M. Odobez, editors, *International Workshop on Multimodal Multiparty Meeting Processing, MMMP 2005, part of the 7th International Conference on Multimodal Interfaces, ICMI 2005*.

Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, Andrew McCallum, and Michael L Wick. 2011. Samplerank: Training Factor Graphs with Atomic Gradients. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 777–784.

David A Smith and Jason Eisner. 2008. Dependency Parsing by Belief Propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 145–156.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational linguistics* 26(3):339–373.

Lu Wang and Claire Cardie. 2012. Focused Meeting Summarization via Unsupervised Relation Extraction. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Seoul, South Korea.

Lu Wang and Claire Cardie. 2013. Domain-Independent Abstract Generation for Focused Meeting Summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1395–1405.

Argument Mining with Structured SVMs and RNNs

Vlad Niculae
Cornell University
vlad@cs.cornell.edu

Joonsuk Park
Williams College
jpark@cs.williams.edu

Claire Cardie
Cornell University
cardie@cs.cornell.edu

Abstract

We propose a novel factor graph model for argument mining, designed for settings in which the argumentative relations in a document do not necessarily form a tree structure. (This is the case in over 20% of the web comments dataset we release.) Our model jointly learns elementary unit type classification and argumentative relation prediction. Moreover, our model supports SVM and RNN parametrizations, can enforce structure constraints (e.g., transitivity), and can express dependencies between adjacent relations and propositions. Our approaches outperform unstructured baselines in both web comments and argumentative essay datasets.

1 Introduction

Argument mining consists of the automatic identification of argumentative structures in documents, a valuable task with applications in policy making, summarization, and education, among others. The argument mining task includes the tightly-knit subproblems of classifying propositions into elementary unit types and detecting argumentative relations between the elementary units. The desired output is a document argumentation graph structure, such as the one in Figure 1, where propositions are denoted by letter subscripts, and the associated argumentation graph shows their types and support relations between them.

Most annotation and prediction efforts in argument mining have focused on tree or forest structures (Peldszus and Stede, 2015; Stab and Gurevych, 2016), constraining argument structures to form one or more trees. This makes the problem computationally easier by enabling the use of maximum spanning tree–style parsing approaches. However, argumentation *in the wild* can be less well-formed. The argument put forth in Figure 1, for instance, consists of two components: a simple tree structure and a more complex graph structure (c jointly supports b and d). In this work, we design a flexible and highly expressive structured prediction model for argument mining, jointly learning to classify elementary units (henceforth *propositions*) and to identify the argumentative relations between them (henceforth *links*). By formulating argument mining as inference in a factor graph (Kschischang et al., 2001), our model (described in Section 4) can account for correlations between the two tasks, can consider second order link structures (e.g., in Figure 1, $c \rightarrow b \rightarrow a$), and can impose arbitrary constraints (e.g., transitivity).

To parametrize our models, we evaluate two alternative directions: linear structured SVMs

[Calling a debtor at work is counter-intuitive;]$_a$ [if collectors are continuously calling someone at work, other employees may report it to the debtor's supervisor.]$_b$ [Most companies have established rules about receiving or making personal calls during working hours.]$_c$ [If a collector or creditor calls a debtor on his/her cell phone and is informed that the debtor is at work, the call should be terminated.]$_d$ [No calls to employers should be allowed,]$_e$ [as this jeopardizes the debtor's job.]$_f$

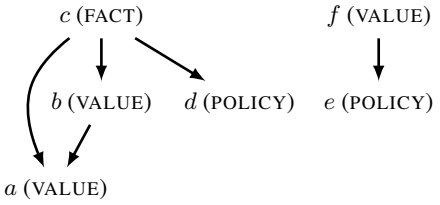


Figure 1: Example annotated CDCP comment.[1]

[1] We describe proposition types (FACT, etc.) in Section 3.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 985–995
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1091

(Tsochantaridis et al., 2005), and recurrent neural networks with structured loss, extending (Kiperwasser and Goldberg, 2016). Interestingly, RNNs perform poorly when trained with classification losses, but become competitive with the feature-engineered structured SVMs when trained within our proposed structured learning model.

We evaluate our approach on two argument mining datasets. Firstly, on our new *Cornell eRulemaking Corpus – CDCP*,[2] consisting of argument annotations on comments from an eRulemaking discussion forum, where links don't always form trees (Figure 1 shows an abridged example comment, and Section 3 describes the dataset in more detail). Secondly, on the UKP argumentative essays v2 (henceforth UKP), where argument graphs are annotated strictly as multiple trees (Stab and Gurevych, 2016). In both cases, the results presented in Section 5 confirm that our models outperform unstructured baselines. On UKP, we improve link prediction over the best reported result in (Stab and Gurevych, 2016), which is based on integer linear programming postprocessing. For insight into the strengths and weaknesses of the proposed models, as well as into the differences between SVM and RNN parameterizations, we perform an error analysis in Section 5.1. To support argument mining research, we also release our Python implementation, `Marseille`.[3]

2 Related work

Our factor graph formulation draws from ideas previously used independently in parsing and argument mining. In particular, maximum spanning tree (MST) methods for arc-factored dependency parsing have been successfully used by McDonald et al. (2005) and applied to argument mining with mixed results by Peldszus and Stede (2015). As they are not designed for the task, MST parsers cannot directly handle proposition classification or model the correlation between proposition and link prediction—a limitation our model addresses. Using RNN features in an MST parser with a structured loss was proposed by Kiperwasser and Goldberg (2016); their model can be seen as a particular case of our factor graph approach, limited to link prediction with a tree structure constraint. Our models support multi-task learning for proposition classification, parameterizing adjacent links with higher-order structures (e.g., $c \rightarrow b \rightarrow a$) and enforcing arbitrary constraints on the link structure, not limited to trees. Such higher-order structures and logic constraints have been successfully used for dependency and semantic parsing by Martins et al. (2013) and Martins and Almeida (2014); to our knowledge we are the first to apply them to argument mining, as well as the first to parametrize them with neural networks. Stab and Gurevych (2016) used an integer linear program to combine the output of independent proposition and link classifiers using a hand-crafted scoring formula, an approach similar to our baseline. Our factor graph method can combine the two tasks in a more principled way, as it fully learns the correlation between the two tasks without relying on hand-crafted scoring, and therefore can readily be applied to other argumentation datasets. Furthermore, our model can enforce the tree structure constraint, required on the UKP dataset, using MST cycle constraints used by Stab and Gurevych (2016), thanks to the AD^3 inference algorithm (Martins et al., 2015).

Sequence tagging has been applied to the related structured tasks of proposition identification and classification (Stab and Gurevych, 2016; Habernal and Gurevych, 2016; Park et al., 2015b); integrating such models is an important next step. Meanwhile, a new direction in argument mining explores *pointer networks* (Potash et al., 2016); a promising method, currently lacking support for tree structures and domain-specific constraints.

3 Data

We release a new argument mining dataset consisting of user comments about rule proposals regarding Consumer Debt Collection Practices (CDCP) by the Consumer Financial Protection Bureau collected from an eRulemaking website, `http://regulationroom.org`.

Argumentation structures found in web discussion forums, such as the eRulemaking one we use, can be more free-form than the ones encountered in controlled, elicited writing such as (Peldszus and Stede, 2015). For this reason, we adopt the model proposed by Park et al. (2015a), which does not constrain links to form tree structures, but unrestricted directed graphs. Indeed, over 20% of the comments in our dataset exhibit local structures that would not be allowable in a tree. Possible link types are *reason* and *evidence*, and propo-

[2] Dataset available at `http://joonsuk.org`.

[3] Available at `https://github.com/vene/marseille`.

sition types are split into five fine-grained categories: POLICY and VALUE contain subjective judgements/interpretations, where only the former specifies a specific course of action to be taken. On the other hand, TESTIMONY and FACT do not contain subjective expressions, the former being about personal experience, or "anecdotal." Lastly, REFERENCE covers URLs and citations, which are used to point to objective evidence in an online setting.

In comparison, the UKP dataset (Stab and Gurevych, 2016) only makes the syntactic distinction between CLAIM, MAJOR CLAIM, and PREMISE types, but it also includes *attack* links. The permissible link structure is stricter in UKP, with links constrained in annotation to form one or more disjoint directed trees within each paragraph. Also, since web arguments are not necessarily fully developed, our dataset has many argumentative propositions that are not in any argumentation relations. In fact, it isn't unusual for comments to have no argumentative links at all: 28% of CDCP comments have no links, unlike UKP, where all essays have complete argument structures. Such comments with no links make the problem harder, emphasizing the importance of capturing the *lack* of argumentative support, not only its presence.

3.1 Annotation results

Each user comment was annotated by two annotators, who independently annotated the boundaries and types of propositions, as well as the links among them.[4] To produce the final corpus, a third annotator manually resolved the conflicts,[5] and two automatic preprocessing steps were applied: we take the link transitive closure, and we remove a small number of nested propositions.[6] The resulting dataset contains 731 comments, consisting of about 3800 sentences (≈4700 propositions) and 88k words. Out of the 43k possible pairs of propositions, links are present between only 1300 (roughly 3%). In comparison, UKP has fewer documents (402), but they are longer, with a total of 7100 sentences (6100 propositions) and 147k words. Since UKP links only occur within the same paragraph and propositions not connected to the argument are removed in a preprocessing step, link prediction is less imbalanced in UKP, with 3800 pairs of propositions being linked out of a total of 22k (17%). We reserve a test set of 150 documents (973 propositions, 272 links) from CDCP, and use the provided 80-document test split from UKP (1266 propositions, 809 links).

[4]The annotators used the GATE annotation tool (Cunningham et al., 2011).

[5]Inter-annotator agreement is measured with Krippendorff's α (Krippendorff, 1980) with respect to elementary unit type (α=64.8%) and links (α=44.1%). A separate paper describing the dataset is under preparation.

[6]When two propositions overlap, we keep the one that results in losing the fewest links. For generality, we release the dataset without this preprocessing, and include code to reproduce it; we believe that handling nested argumentative units is an important direction for further research.

4 Structured learning for argument mining

4.1 Preliminaries

Binary and multi-class classification have been applied with some success to proposition and link prediction separately, but we seek a way to jointly learn the argument mining problem at the *document* level, to better model contextual dependencies and constraints. We therefore turn to *structured learning*, a framework that provides the desired level of expressivity.

In general, learning from a dataset of documents $x_i \in \mathcal{X}$ and their associated labels $y_i \in \mathcal{Y}$ involves seeking model parameters $\boldsymbol{w}$ that can "pick out" the best label under a scoring function f:

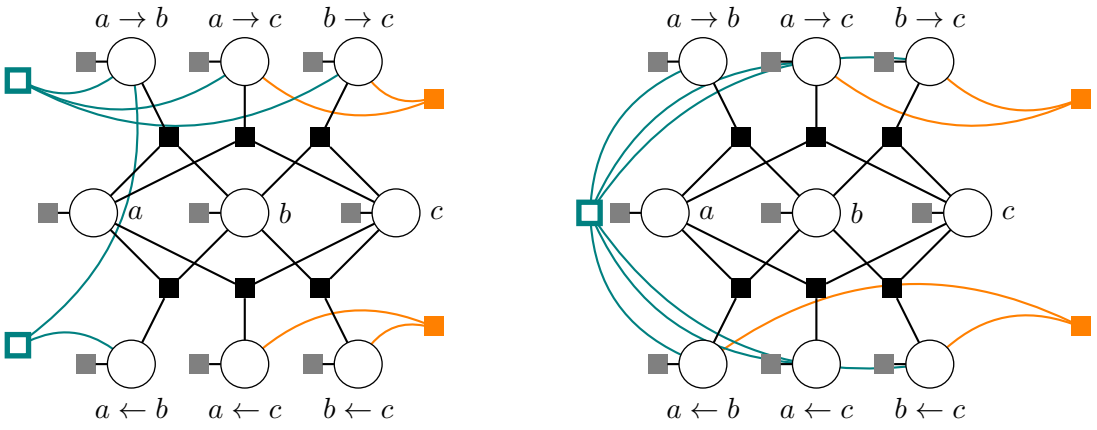
$$\hat{y} := \arg\max_{y \in \mathcal{Y}} f(x, y; \boldsymbol{w}). \quad (1)$$

Unlike classification or regression, where $\mathcal{X}$ is usually a feature space $\mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}$ (e.g., we predict an integer class index or a probability), in structured learning, more complex inputs and outputs are allowed. This makes the $\arg\max$ in Equation 1 impossible to evaluate by enumeration, so it is desirable to find models that decompose over smaller units and dependencies between them; for instance, as *factor graphs*. In this section, we give a factor graph description of our proposed structured model for argument mining.

4.2 Model description

An input document is a string of words with proposition offsets delimited. We denote the propositions in a document by $\{a, b, c, ...\}$ and the possible directed link between a and b as $a \to b$. The argument structure we seek to predict consists of the type of each proposition $y_a \in \mathcal{P}$ and a binary label for each link $y_{a \to b} \in \mathcal{R} = \{\texttt{on}, \texttt{off}\}$.[7]

[7]For simplicity and comparability, we follow Stab and Gurevych (2016) in using binary link labels even if links could be of different types. This can be addressed in our model by incorporating "labeled link" factors.



(a) CDCP (b) UKP

Figure 2: Factor graphs for a document with three propositions (a, b, c) and the six possible edges between them, and some of the factors used, illustrating differences and similarities between our models for the two datasets. Unary factors are light gray; compatibility factors are black. Factors not part of the basic model have curved edges: higher-order factors are orange and on the right; link structure factors are hollow, as that they don't have any parameters. Strict constraint factors are omitted for simplicity.

The possible proposition types $\mathcal{P}$ differ for the two datasets; such differences are documented in Table 1. As we describe the variables and factors constituting a document's factor graph, we shall refer to Figure 2 for illustration.

Unary potentials. Each proposition a and each link $a \to b$ has a corresponding random variable in the factor graph (the circles in Figure 2). To encode the model's belief in each possible value for these variables, we parametrize the *unary factors* (gray boxes in Figure 2) with unary potentials: $\boldsymbol{\phi}(a) \in \mathbb{R}^{|\mathcal{P}|}$ is a score of y_a for each possible proposition type. Similarly, link unary potentials $\boldsymbol{\phi}(a \to b) \in \mathbb{R}^{|\mathcal{R}|}$ are scores for $y_{a \to b}$ being `on`/`off`. Without any other factors, this would amount to independent classifiers for each task.

Compatibility factors. For every possible link $a \to b$, the variables $(a, b, a \to b)$ are bound by a dense factor scoring their joint assignment (the black boxes in Figure 2). Such a factor could automatically learn to encourage links from compatible types (e.g., from TESTIMONY to POLICY) or discourage links between less compatible ones (e.g., from FACT to TESTIMONY). In the simplest form, this factor would be parametrized as a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{|\mathcal{P}| \times |\mathcal{P}| \times |\mathcal{R}|}$, with t_{ijk} retaining the score of a source proposition of type i to be (k = `on`) or not to be (k = `off`) in a link with a proposition of type j. For more flexibility, we parametrize this factor with **compatibility features** depending only on simple structure: $\boldsymbol{t}_{ijk}$ becomes a vector, and the score of configuration (i, j, k) is given by $\boldsymbol{v}_{ab}^{\top} \boldsymbol{t}_{ijk}$ where $\boldsymbol{v}_{ab}$ consists of three binary features:

- **bias:** a constant value of 1, allowing $\boldsymbol{\mathcal{T}}$ to learn a base score for a label configuration (i, j, k), as in the simple form above,
- **adjacency:** when there are no other propositions between the source and the target,
- **order:** when the source precedes the target.

Second order factors. Local argumentation graph structures such as $a \to b \to c$ might be modeled better together rather than through separate link factors for $a \to b$ and $b \to c$. As in higher-order structured models for semantic and dependency parsing (Martins et al., 2013; Martins and Almeida, 2014), we implement three types of second order factors: **grandparent** $(a \to b \to c)$, **sibling** $(a \leftarrow b \to c)$, and **co-parent** $(a \to b \leftarrow c)$. Not all of these types of factors make sense on all datasets: as sibling structures cannot exist in directed trees, we don't use sibling factors on UKP. On CDCP, by transitivity, every grandparent structure implies a corresponding sibling, so it is sufficient to parametrize siblings. This difference between datasets is emphasized in Figure 2, where one example of each type of factor is pictured on the right side of the graphs (orange boxes with curved edges): on CDCP we illustrate a co-parent factor (top right) and a sibling factor (bot-

tom right), while on UKP we show a co-parent factor (top right) and a grandparent factor (bottom right). We call these factors *second order* because they involve two link variables, scoring the joint assignment of both links being on.

Valid link structure. The global structure of argument links can be further constrained using domain knowledge. We implement this using constraint factors; these have no parameters and are denoted by empty boxes in Figure 2. In general, well-formed arguments should be cycle-free. In the UKP dataset, links form a directed forest and can never cross paragraphs. This particular constraint can be expressed as a series of *tree factors*,[8] one for each paragraph (the factor connected to all link variables in Figure 2). In CDCP, links do not form a tree, but we use logic constraints to enforce transitivity (top left factor in Figure 2) and to prevent symmetry (bottom left); the logic formulas implemented by these factors are described in Table 1. Together, the two constraints have the desirable side effect of preventing cycles.

Strict constraints. We may include further domain-specific constraints into the model, to express certain disallowed configurations. For instance, proposition types that appear in CDCP data can be ordered by the level of objectivity (Park et al., 2015a), as shown in Table 1. In a well-formed argument, we would want to see links from more objective to equally or less objective propositions: it's fine to provide FACT as reason for VALUE, but not the other way around. While the training data sometimes violates this constraint, enforcing it might provide a useful inductive bias.

Inference. The $\arg\max$ in Equation 1 is a MAP over a factor graph with cycles and many overlapping factors, including logic factors. While exact inference methods are generally unavailable, our setting is perfectly suited for the Alternating Directions Dual Decomposition (AD^3) algorithm: approximate inference on expressive factor graphs with overlapping factors, logic constraints, and generic factors (e.g., directed tree factors) defined through maximization oracles (Martins et al., 2015). When AD^3 returns an integral solution, it is globally optimal, but when solutions are fractional, several options are available. At test time, for analysis, we retrieve exact solutions using the branch-and-bound method. At training time, however, fractional solutions can be used *as-is*; this makes better use of each iteration and actually increases the ratio of integral solutions in future iterations, as well as at test time, as proven by Meshi et al. (2016). We also find that after around 15 training iterations with fractional solutions, over 99% of inference calls are integral.

Learning. We train the models by minimizing the structured hinge loss (Taskar et al., 2004):

$$\sum_{(x,y)\in\mathcal{D}} \max_{y'\in\mathcal{Y}} (f(x,y';\boldsymbol{w}) + \rho(y,y')) - f(x,y;\boldsymbol{w}) \tag{2}$$

where ρ is a configurable misclassification cost. The $\max$ in Equation 2 is not the same as the one used for prediction, in Equation 1. However, when the cost function ρ decomposes over the variables, cost-augmented inference amounts to regular inference after augmenting the potentials accordingly. We use a weighted Hamming cost:

$$\rho(y,\hat{y}) := \sum_v \rho(y_v)\mathbb{I}[y_v = \hat{y}_v]$$

where v is summed over all variables in a document $\{a\} \cup \{a \to b\}$, and $\rho(y_v)$ is a misclassification cost. We assign uniform costs ρ to 1 for all mistakes except false-negative links, where we use higher cost proportional to the class imbalance in the training split, effectively giving more weight to positive links during training.

4.3 Argument structure SVM

One option for parameterizing the potentials of the unary and higher-order factors is with *linear models*, using proposition, link, and higher-order features. This gives birth to a linear structured SVM (Tsochantaridis et al., 2005), which, when using l_2 regularization, can be trained efficiently in the dual using the online block-coordinate Frank-Wolfe algorithm of Lacoste-Julien et al. (2013), as implemented in the `pystruct` library (Müller and Behnke, 2014). This algorithm is more convenient than subgradient methods, as it does not require tuning a learning rate parameter.

Features. For unary proposition and link features, we faithfully follow Stab and Gurevych (2016, Tables 9 and 10): proposition features are

[8] A tree factor regards each bound variable as an edge in a graph and assigns $-\infty$ scores to configurations that are not valid trees. For inference, we can use maximum spanning arborescence algorithms such as Chu-Liu/Edmonds.

Model part	CDCP dataset	UKP dataset
proposition types	REFERENCE $\succ$ TESTIMONY $\succ$ FACT $\succ$ VALUE $\succ$ POLICY	CLAIM, MAJOR CLAIM, PREMISE
links	all possible	within each paragraph
2nd order factors	siblings, co-parents	grandparents, co-parents
link structure	transitive acyclic: • $a \to b \;\&\; b \to c \implies a \to c$ • ATMOSTONE$(a \to b, b \to a)$	directed forest: • TREEFACTOR over each paragraph • zero-potential "root" links $a \to *$
strict constraints	link source must be as least as objective as the target: $a \to b \implies a \succeq b$	link source must be premise: $a \to b \implies a =$ PREMISE

Table 1: Instantiation of model design choices for each dataset.

lexical (unigrams and dependency tuples), structural (token statistics and proposition location), indicators (from hand-crafted lexicons), contextual, syntactic (subclauses, depth, tense, modal, and POS), probability, discourse (Lin et al., 2014), and average GloVe embeddings (Pennington et al., 2014). Link features are lexical (unigrams), syntactic (POS and productions), structural (token statistics, proposition statistics and location features), hand-crafted indicators, discourse triples, PMI, and shared noun counts.

Our proposed higher-order factors for grandparent, co-parent, and sibling structures require features extracted from a proposition triplet a, b, c. In dependency and semantic parsing, higher-order factors capture relationships between words, so sparse indicator features can be efficiently used. In our case, since propositions consist of many words, BOW features may be too noisy and too dense; so for simplicity we again take a cue from the link-specific features used by Stab and Gurevych (2016). Our higher-order factor features are: same sentence indicators (for all 3 and for each pair), proposition order (one for each of the 6 possible orderings), Jaccard similarity (between all 3 and between each pair), presence of any shared nouns (between all 3 and between each pair), and shared noun ratios: nouns shared by all 3 divided by total nouns in each proposition and each pair, and shared nouns between each pair with respect to each proposition. Up to vocabulary size difference, our total feature dimensionality is approximately 7000 for propositions and 2100 for links. The number of second order features is 35.

Hyperparameters. We pick the SVM regularization parameter $C \in \{0.001, 0.003, 0.01, 0.03, 0.1, 0.3\}$ by k-fold cross validation at document level, optimizing for the average between link and proposition F_1 scores.

4.4 Argument structure RNN

Neural network methods have proven effective for natural language problems even with minimal-to-no feature engineering. Inspired by the use of LSTMs (Hochreiter and Schmidhuber, 1997) for MST dependency parsing by Kiperwasser and Goldberg (2016), we parametrize the potentials in our factor graph with an LSTM-based neural network,[9] replacing MST inference with the more general AD^3 algorithm, and using relaxed solutions for training when inference is inexact.

We extract embeddings of all words with a corpus frequency > 1, initialized with GloVe word vectors. We use a deep bidirectional LSTM to encode contextual information, representing a proposition a as the average of the LSTM outputs of its words, henceforth denoted $\overleftrightarrow{\boldsymbol{a}}$.

Proposition potentials. We apply a multi-layer perceptron (MLP) with rectified linear activations to each proposition, with all layer dimensions equal except the final output layer, which has size $|\mathcal{P}|$ and is not passed through any nonlinearities.

Link potentials. To score a dependency $a \to b$, Kiperwasser and Goldberg (2016) pass the concatenation $[\overleftrightarrow{\boldsymbol{a}}; \overleftrightarrow{\boldsymbol{b}}]$ through an MLP. After trying this, we found slightly better performance by first passing *each* proposition through a slot-specific dense layer $\left(\overline{\boldsymbol{a}} := \sigma_{\text{src}}(\overleftrightarrow{\boldsymbol{a}}), \overline{\boldsymbol{b}} := \sigma_{\text{trg}}(\overleftrightarrow{\boldsymbol{b}})\right)$ followed by a *bilinear* transformation:

$$\phi_{\text{on}}(a \to b) := \overline{\boldsymbol{a}}^{\top} \boldsymbol{W} \overline{\boldsymbol{b}} + \boldsymbol{w}_{\text{src}}^{\top} \overline{\boldsymbol{a}} + \boldsymbol{w}_{\text{trg}}^{\top} \overline{\boldsymbol{b}} + w_0^{(\text{on})}.$$

Since the bilinear expression returns a scalar, but the link potentials must have a value for both the `on` and `off` states, we set the full potential to $\boldsymbol{\phi}(a \to b) := [\phi_{\text{on}}(a \to b), w_0^{(\text{off})}]$ where $w_0^{(\text{off})}$ is a learned scalar bias. We initialize $\boldsymbol{W}$ to the diagonal identity matrix.

[9] We use the `dynet` library (Neubig et al., 2017).

Second order potentials. Grandparent potentials $\phi(a \to b \to c)$ score two adjacent directed edges, in other words three propositions. We again first pass each proposition representation through a slot-specific dense layer. We implement a multilinear scorer analogously to the link potentials:

$$\phi(a \to b \to c) := \sum_{i,j,k} \overline{\boldsymbol{a}}_i \overline{\boldsymbol{b}}_j \overline{\boldsymbol{c}}_k w_{ijk}$$

where $\mathcal{W} = (w)_{ijk}$ is a third-order cube tensor. To reduce the large numbers of parameters, we implicitly represent $\mathcal{W}$ as a rank r tensor: $w_{ijk} = \sum_{s=1}^{r} u_{is}^{(1)} u_{js}^{(2)} u_{ks}^{(3)}$. Notably, this model captures only third-order interactions between the representation of the three propositions. To capture first-order "bias" terms, we could include slot-specific linear terms, e.g., $\boldsymbol{w}_a^\top \overline{\boldsymbol{a}}$; but to further capture quadratic *backoff* effects (for instance, if two propositions carry a strong signal of being siblings regardless of their parent), we would require quadratically many parameters. Instead of explicit lower-order terms, we propose *augmenting* $\overline{\boldsymbol{a}}, \overline{\boldsymbol{b}}$, and $\overline{\boldsymbol{c}}$ with a constant feature of 1, which has approximately the same effect, while benefiting from the parameter sharing in the low-rank factorization; an effect described by Blondel et al. (2016). Siblings and co-parents factors are similarly parametrized with their own tensors.

Hyperparameters. We perform grid search using k-fold document-level cross-validation, tuning the dropout probability in the dense MLP layers over $\{0.05, 0.1, 0.15, 0.2, 0.25\}$ and the optimal number of passes over the training data over $\{10, 25, 50, 75, 100\}$. We use 2 layers for the LSTM and the proposition classifier, 128 hidden units in all layers, and a multilinear decomposition with rank $r = 16$, after preliminary CV runs.

4.5 Baseline models

We compare our proposed models to equivalent independent unary classifiers. The unary-only version of a structured SVM is an l_2-regularized linear SVM.[10] For the RNN, we compute unary potentials in the same way as in the structured model, but apply independent hinge losses at each variable, instead of the global structured hinge loss. Since the RNN weights are shared, this is a form of multi-task learning. The baseline predictions can be interpreted as unary potentials, therefore we can simply round their output to the highest scoring labels, or we can, alternatively, perform test-time inference, imposing the desired structure.

5 Results

We evaluate our proposed models on both datasets. For model selection and development we used k-fold cross-validation at document level: on CDCP we set $k = 3$ to avoid small validation folds, while on UKP we follow Stab and Gurevych (2016) setting $k = 5$. We compare our proposed structured learning systems (the linear structured SVM and the structured RNN) to the corresponding baseline versions. We organize our experiments in three incremental variants of our factor graph: *basic*, *full*, and *strict*, each with the following components:[11]

component	basic	full	strict	(baseline)
unaries	✓	✓	✓	✓
compat. factors	✓	✓	✓	
compat. features		✓	✓	
higher-order		✓	✓	
link structure		✓	✓	✓
strict constraints			✓	✓

Following Stab and Gurevych (2016), we compute F_1 scores at proposition and link level, and also report their average as a summary of overall performance.[12] The results of a single prediction run on the test set are displayed in Table 2. The overall trend is that training using a structured objective is better than the baseline models, even when structured inference is applied on the baseline predictions. On UKP, for link prediction, the linear baseline can reach good performance when using inference, similar to the approach of Stab and Gurevych (2016), but the improvement in proposition prediction leads to higher overall F_1 for the structured models. Meanwhile, on the more difficult CDCP setting, performing inference on the baseline output is not competitive. While feature engineering still outperforms our RNN model, we find that RNNs shine on proposition classification, especially on UKP, and that structured training can make them competitive, reducing their observed lag on link prediction (Katiyar and Cardie, 2016), possibly through mitigating class imbalance.

[10] We train our SVM using SAGA (Defazio et al., 2014) in `lightning` (Blondel and Pedregosa, 2016).

[11] Components are described in Section 4. The baselines *with inference* support only unaries and factors with no parameters, as indicated in the last column.

[12] For link F_1 scores, however, we find it more intuitive to only consider retrieval of positive links rather than macro-averaged two-class scores.

	Baseline						Structured					
	SVM			RNN			SVM			RNN		
Metric	basic	full	strict	basic	full	strict	basic	full	strict	basic	full	strict
CDCP dataset												
Average	47.4	47.3	47.9	40.8	38.0	38.0	48.1	49.3	**50.0**	43.5	33.5	38.2
Link (272)	22.0	21.9	23.8	9.9	12.8	12.8	24.7	25.1	**26.7**	14.4	14.6	10.5
Proposition	72.7	72.7	72.0	71.8	63.2	63.2	71.6	**73.5**	73.2	72.7	52.4	65.9
VALUE (491)	75.3	75.3	74.4	74.1	74.8	74.8	73.4	75.7	76.4	73.7	73.1	69.7
POLICY (153)	78.7	78.7	78.5	74.3	72.2	72.2	72.3	77.3	76.8	73.9	74.4	76.8
TESTIMONY (204)	70.3	70.3	68.6	74.6	71.8	71.8	69.8	71.7	71.5	74.2	72.3	75.8
FACT (124)	39.2	39.2	38.3	35.8	30.5	30.5	42.4	42.5	41.3	41.5	42.2	40.5
REFERENCE (1)	100.0	100.0	100.0	100.0	66.7	66.7	100.0	100.0	100.0	100.0	0.0	66.7
UKP dataset												
Average	64.7	66.6	66.5	58.7	57.4	58.7	67.1	**68.9**	67.1	59.0	63.6	64.7
Link (809)	55.8	59.7	**60.3**	44.8	43.8	44.0	56.9	60.1	56.9	44.1	50.4	50.1
Proposition	73.5	73.5	72.6	72.6	70.9	73.3	77.2	77.6	77.3	74.0	76.9	**79.3**
MAJOR CLAIM (153)	76.7	76.7	77.6	81.4	75.1	81.3	77.0	78.2	80.0	83.6	84.6	88.3
CLAIM (304)	55.4	55.4	52.0	51.7	52.7	53.5	64.3	64.5	62.8	53.2	60.2	62.0
PREMISE (809)	88.4	88.4	88.3	84.8	84.8	85.2	90.3	90.2	89.2	85.0	85.9	87.6

Table 2: Test set F_1 scores for link and proposition classification, as well as their average, on the two datasets. The number of test instances is shown in parentheses; best scores on overall tasks are in bold.

5.1 Discussion and analysis

Contribution of compatibility features. The compatibility factor in our model can be visualized as conditional odds ratios given the source and target proposition types. Since there are only four possible configurations of the compatibility features, we can plot all cases in Figure 3, alongside the basic model. Not using compatibility features, the basic model can only learn whether certain configurations are more likely than others (e.g. a REFERENCE supporting another REFERENCE is unlikely, while a REFERENCE supporting a FACT is more likely; essentially a soft version of our domain-specific strict constraints. The full model with compatibility features is finer grained, capturing, for example, that links from REFERENCE to FACT are more likely when the reference comes *after*, or that links from VALUE to POLICY are extremely likely only when the two are adjacent.

Proposition errors. The confusion matrices in Figure 4 reveal that the most common confusion is misclassifying FACT as VALUE. The strongest difference between the various models tested is that the RNN-based models make this error less often. For instance, in the proposition:

> And the single most frequently used excuse of any debtor is "I didn't receive the letter/invoice/statement"

the pronouns in the nested quote may be mistaken for subjectivity, leading to the structured SVMs predictions of VALUE or TESTIMONY, while the basic structured RNN correctly classifies it as FACT.

Link errors. While structured inference certainly helps baselines by preventing invalid structures such as cycles, it still depends on local decisions, losing to fully structured training in cases where joint proposition and link decisions are needed. For instance, in the following conclusion of an UKP essay, the annotators found no links:

> In short, [the individual should finance his or her education]$_a$ because [it is a personal choice.]$_b$ Otherwise, [it would cause too much cost from taxpayers and the government.]$_c$

Indeed, no reasons are provided, but baseline are misled by the connectives: the SVM baseline outputs that b and c are PREMISES supporting the CLAIM a. The full structured SVM combines the two tasks and correctly recognizes the link structure.

Linear SVMs are still a very good baseline, but they tend to overgenerate links due to class imbalance, even if we use class weights during training. Surprisingly, RNNs are at the opposite end, being extremely conservative, and getting the highest precision among the models. On CDCP, where the number of true links is 272, the linear baseline with strict inference predicts 796 links with a precision of only 16%, while the strict structured RNN only predicts 52 links, with 33% precision; the example in Figure 5 illustrates this. In terms of higher-order structures, we find that using higher-order factors increases precision, at a cost in recall.

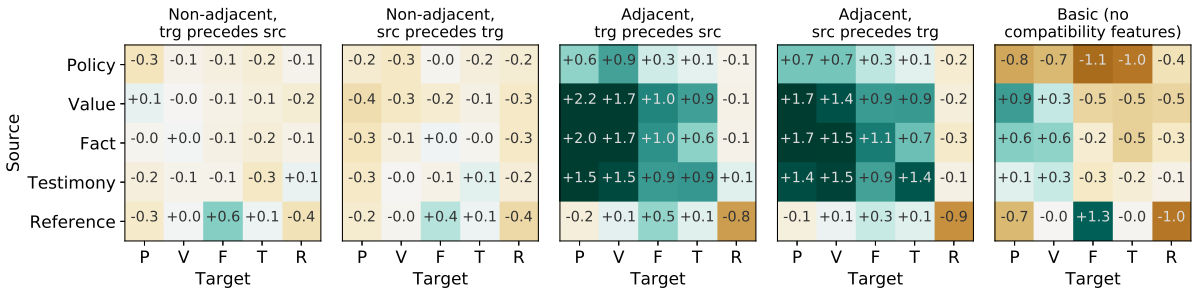


Figure 3: Learned conditional log-odds $\log \frac{p(\texttt{on}|\cdot)}{p(\texttt{off}|\cdot)}$, given the source and target proposition types and compatibility feature settings. First four figures correspond to the four possible settings of the compatibility features in the full structured SVM model. For comparison, the rightmost figure shows the same parameters in the basic structured SVM model, which does not use compatibility features.

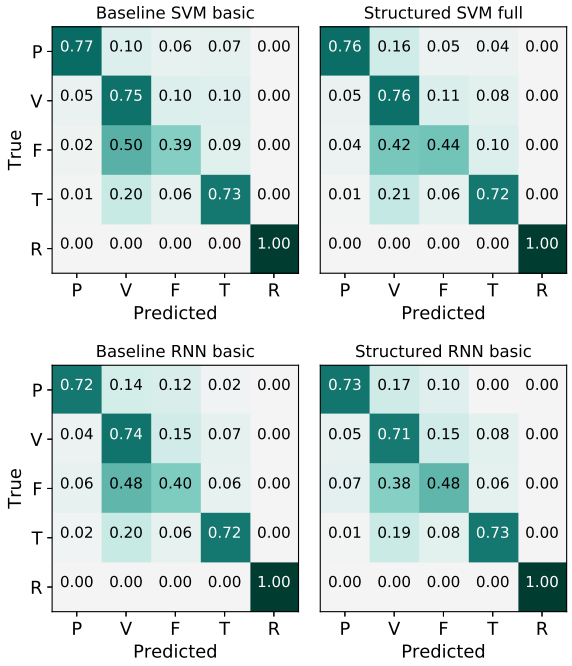


Figure 4: Normalized confusion matrices for proposition type classification.

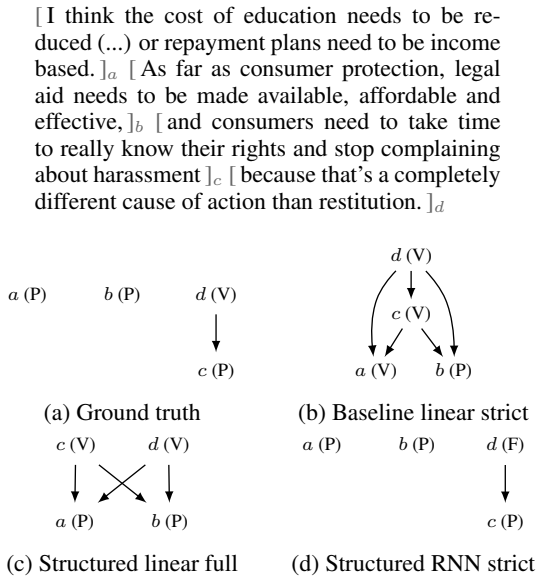


Figure 5: Predictions on a CDCP comment where the structured RNN outperforms the other models.

This is most beneficial for the 856 co-parent structures in the UKP test set: the full structured SVM has 53% F_1, while the basic structured SVM and the basic baseline get 47% and 45% respectively. On CDCP, while higher-order factors help, performance on siblings and co-parents is below 10% F_1 score. This is likely due to link sparsity and suggests plenty of room for further development.

6 Conclusions and future work

We introduce an argumentation parsing model based on AD^3 relaxed inference in expressive factor graphs, experimenting with both linear structured SVMs and structured RNNs, parametrized with higher-order factors and link structure constraints. We demonstrate our model on a new argumentation mining dataset with more permissive argument structure annotation. Our model also achieves state-of-the-art link prediction performance on the UKP essays dataset.

Future work. Stab and Gurevych (2016) found polynomial kernels useful for modeling feature interactions, but kernel structured SVMs scale poorly, we intend to investigate alternate ways to capture feature interactions. While we focus on monological argumentation, our model could be extended to dialogs, for which argumentation theory thoroughly motivates non-tree structures (Afantenos and Asher, 2014).

Acknowledgements

We are grateful to André Martins, Andreas Müller, Arzoo Katyiar, Chenhao Tan, Felix Wu, Jack Hessel, Justine Zhang, Mathieu Blondel, Tianze Shi, Tobias Schnabel, and the rest of the Cornell NLP seminar for extremely helpful discussions. We thank the anonymous reviewers for their thorough and well-argued feedback.

References

Stergos Afantenos and Nicholas Asher. 2014. Counter-argumentation and discourse: A case study. In *Proceedings of ArgNLP*.

Mathieu Blondel, Masakazu Ishihata, Akinori Fujino, and Naonori Ueda. 2016. Polynomial networks and factorization machines: New insights and efficient training algorithms. In *Proceedings of ICML*.

Mathieu Blondel and Fabian Pedregosa. 2016. Lightning: large-scale linear classification, regression and ranking in Python. https://doi.org/10.5281/zenodo.200504.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. 2011. *Text Processing with GATE (Version 6)*.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. 2014. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proceedings of NIPS*.

Ivan Habernal and Iryna Gurevych. 2016. Argumentation mining in user-generated web discourse. *Computational Linguistics* .

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

Arzoo Katiyar and Claire Cardie. 2016. Investigating LSTMs for joint extraction of opinion entities and relations. In *Proceedings of ACL*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *arXiv:1603.04351* preprint.

Klaus Krippendorff. 1980. *Content Analysis: An Introduction to Its Methodology*. Commtext. Sage.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47(2):498–519.

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. 2013. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proceedings of ICML*.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A PDTB-styled end-to-end discourse parser. *Natural Language Engineering* 20(02):151–184.

André FT Martins and Mariana SC Almeida. 2014. Priberam: A Turbo Semantic Parser with second order features. In *Proceedings of SemEval*.

André FT Martins, Miguel B Almeida, and Noah A Smith. 2013. Turning on the Turbo: Fast third-order non-projective Turbo Parsers. In *Proceedings of ACL*.

André FT Martins, Mário AT Figueiredo, Pedro MQ Aguiar, Noah A Smith, and Eric P Xing. 2015. AD3: Alternating directions dual decomposition for MAP inference in graphical models. *Journal of Machine Learning Research* 16:495–545.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of EMNLP*.

Ofer Meshi, Mehrdad Mahdavi, Adrian Weller, and David Sontag. 2016. Train and test tightness of LP relaxations in structured prediction. In *Proceedings of ICML*.

Andreas C Müller and Sven Behnke. 2014. PyStruct: learning structured prediction in Python. *Journal of Machine Learning Research* 15(1):2055–2060.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv:1701.03980* preprint.

Joonsuk Park, Cheryl Blake, and Claire Cardie. 2015a. Toward machine-assisted participation in eRulemaking: An argumentation model of evaluability. In *Proceedings of ICAIL*.

Joonsuk Park, Arzoo Katiyar, and Bishan Yang. 2015b. Conditional random fields for identifying appropriate types of support for propositions in online user comments. In *Proceedings of the 2nd Workshop on Argumentation Mining*. Association for Computational Linguistics, Denver, CO, pages 39–44.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2016. Here's my point: Argumentation mining with pointer networks. *arXiv:1612.08994* preprint.

Christian Stab and Iryna Gurevych. 2016. Parsing argumentation structures in persuasive essays. *arXiv:1604.07370* preprint.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin Markov networks. In *Proceedings of NIPS*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research* 6(Sep):1453–1484.

Neural Discourse Structure for Text Categorization

Yangfeng Ji and **Noah A. Smith**
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
{yangfeng,nasmith}@cs.washington.edu

Abstract

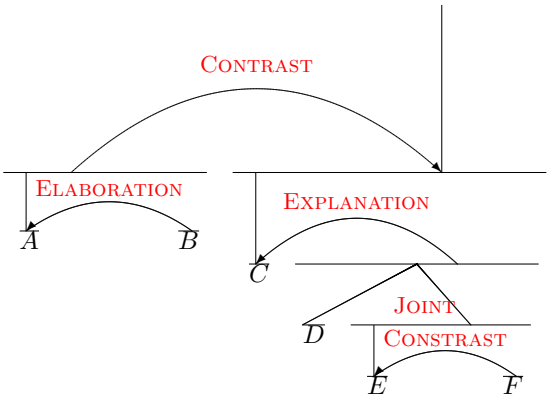
We show that discourse structure, as defined by Rhetorical Structure Theory and provided by an existing discourse parser, benefits text categorization. Our approach uses a recursive neural network and a newly proposed attention mechanism to compute a representation of the text that focuses on salient content, from the perspective of both RST and the task. Experiments consider variants of the approach and illustrate its strengths and weaknesses.

1 Introduction

Advances in text categorization have the potential to improve systems for analyzing sentiment, inferring authorship or author attributes, making predictions, and many more. Several past researchers have noticed that methods that reason about the relative salience or importance of passages within a text can lead to improvements (Ko et al., 2004). Latent variables (Yessenalina et al., 2010), structured-sparse regularizers (Yogatama and Smith, 2014), and neural attention models (Yang et al., 2016) have all been explored.

Discourse structure, which represents the organization of a text as a tree (for an example, see Figure 1), might provide cues for the importance of different parts of a text. Some promising results on sentiment classification tasks support this idea: Bhatia et al. (2015) and Hogenboom et al. (2015) applied hand-crafted weighting schemes to the sentences in a document, based on its discourse structure, and showed benefit to sentiment polarity classification.

In this paper, we investigate the value of discourse structure for text categorization more broadly, considering five tasks, through the use of a recursive neural network built on an automatically-derived document parse from a top-performing, open-source discourse parser, DPLP (Ji and Eisenstein, 2014). Our models learn to weight the importance of a document's sentences, based on their positions and relations in the discourse tree. We introduce a new, unnormalized attention mechanism to this end.



[Although the food was amazing]^{A} [and I was in love with the spicy pork burrito,]^{B} [the service was really awful.]^{C} [We watched our waiter serve himself many drinks.]^{D} [He kept running into the bathroom]^{E} [instead of grabbing our bill.]^{F}

Figure 1: A manually constructed example of the RST (Mann and Thompson, 1988) discourse structure on a text.

Experimental results show that variants of our model outperform prior work on four out of five tasks considered. Our method unsurprisingly underperforms on the fifth task, making predictions about legislative bills—a genre in which discourse conventions are quite different from those in the discourse parser's training data. Further experiments show the effect of discourse parse quality on text categorization performance, suggesting that future improvements to discourse parsing will pay off for text categorization, and validate our new attention mechanism.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 996–1005
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1092

Our implementation is available at https://github.com/jiyfeng/disco4textcat.

2 Background: Rhetorical Structure Theory

Rhetorical Structure Theory (RST; Mann and Thompson, 1988) is a theory of discourse that has enjoyed popularity in NLP. RST posits that a document can be represented by a tree whose leaves are elementary discourse units (EDUs, typically clauses or sentences). Internal nodes in the tree correspond to spans of sentences that are connected via discourse relations such as CONTRAST and ELABORATION. In most cases, a discourse relation links adjacent spans denoted "nucleus" and "satellite," with the former more essential to the writer's purpose than the latter.[1]

An example of a manually constructed RST parse for a restaurant review is shown in Figure 1. The six EDUs are indexed from A to F; the discourse tree organizes them hierarchically into increasingly larger spans, with the last CONTRAST relation resulting in a span that covers the whole review. Within each relation, the RST tree indicates the nucleus pointed by an arrow from its satellite (e.g., in the ELABORATION relation, A is the nucleus and B is the satellite).

The information embedded in RST trees has motivated many applications in NLP research, including document summarization (Marcu, 1999), argumentation mining (Azar, 1999), and sentiment analysis (Bhatia et al., 2015). In most applications, RST trees are built by automatic discourse parsing, due to the expensive cost of manual annotation. In this work, we use a state-of-the-art open-source RST-style discourse parser, DPLP (Ji and Eisenstein, 2014).[2]

We follow recent work that suggests transforming the RST tree into a dependency structure (Yoshida et al., 2014).[3] Figure 2(a) shows the corresponding dependency structure of the RST tree in Figure 1. It is clear that C is the root of the tree, and in fact this clause summarizes the review and suffices to categorize it as negative. This dependency representation of the RST tree offers a form of inductive bias for our neural model, helping it to discern the most salient parts of a text in order to assign it a label.

[1] There are also a few exceptions in which a relation can be realized with multiple nuclei.

[2] https://github.com/jiyfeng/DPLP

[3] The transformation is trivial and deterministic given the nucleus-satellite mapping for each relation. The procedure is analogous to the transformation of a headed phrase-structure parse in syntax into a dependency tree (e.g., Yamada and Matsumoto, 2003).

3 Model

Our model is a recursive neural network built on a discourse dependency tree. It includes a distributed representation computed for each EDU, and a composition function that combines EDUs and partial trees into larger trees. At the top of the tree, the representation of the complete document is used to make a categorization decision. Our approach is analogous to (and inspired by) the use of recursive neural networks on *syntactic* dependency trees, with word embeddings at the leaves (Socher et al., 2014).

3.1 Representation of Sentences

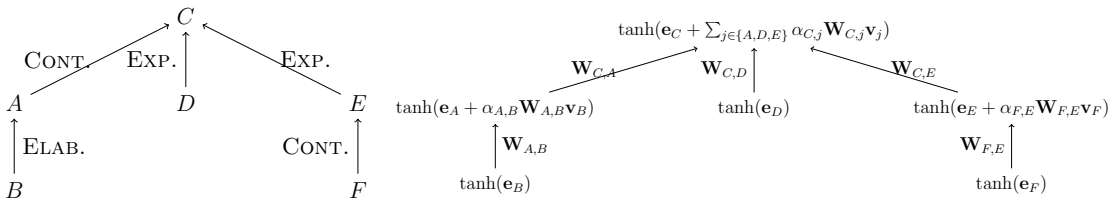
Let $\mathbf{e}$ be the distributed representation of an EDU. We use a bidirectional LSTM on the words' embeddings within each EDU (details of word embeddings are given in section 4), concatenating the last hidden state vector from the forward LSTM ($\overrightarrow{\mathbf{e}}$) with that of the backward LSTM ($\overleftarrow{\mathbf{e}}$) to get $\mathbf{e}$.

There is extensive recent work on architectures for embedding representations of sentences and other short pieces of text, including, for example, (bi)recursive neural networks (Paulus et al., 2014) and convolutional neural networks (Kalchbrenner et al., 2014). Future work might consider alternatives; we chose the bidirectional LSTM due to its effectiveness in many settings.

3.2 Full Recursive Model

Given the discourse dependency tree for an input text, our recursive model builds a vector representation through composition at each arc in the tree. Let $\mathbf{v}_i$ denote the vector representation of EDU i and its descendants. For the base case where EDU i is a leaf in the tree, we let $\mathbf{v}_i = \tanh(\mathbf{e}_i)$, which is the elementwise hyperbolic tangent function.

For an internal node i, the composition function considers a parent and all of its children, whose indices are denoted by $children(i)$. In defining this composition function, we seek for (i.) the contribution of the parent node $\mathbf{e}_i$ to be central; and (ii.) the contribution of each child node $\mathbf{e}_j$ be determined by its content as well as the discourse relation it holds with the parent. We therefore define



(a) dependency structure (b) recursive neural network structure

Figure 2: The dependency discourse tree derived from the example RST tree in Figure 1 (a) and the corresponding recursive neural network model on the tree (b).

$$\mathbf{v}_i = \tanh\left(\mathbf{e}_i + \sum_{j\in children(i)} \alpha_{i,j}\mathbf{W}_{r_{i,j}}\mathbf{v}_j\right), \quad (1)$$

where $\mathbf{W}_{r_{i,j}}$ is a relation-specific composition matrix indexed by the relation between i and j, $r_{i,j}$.

$\alpha_{i,j}$ is an "attention" weight, defined as

$$\alpha_{i,j} = \sigma\left(\mathbf{e}_i^\top \mathbf{W}_\alpha \mathbf{v}_j\right), \quad (2)$$

where σ is the elementwise sigmoid and $\mathbf{W}_\alpha$ contains attention parameters (these are relation-independent). Our attention mechanism differs from prior work (Bahdanau et al., 2015), in which attention weights are normalized to sum to one across competing candidates for attention. Here, $\alpha_{i,j}$ does not depend on node i's other children. This is motivated by RST, in which the presence of a node does not signify lesser importance to its siblings. Consider, for example, EDU D and text span E-F in Figure 1, which in parallel provide EXPLANATION for EDU C. This scenario differs from machine translation, where attention is used to implicitly and softly align output-language words to relatively few input-language words. It also differs from attention in composition functions used in syntactic parsing (Kuncoro et al., 2017), where attention can mimic head rules that follow from an endocentricity hypothesis of syntactic phrase representation.

Our recursive composition function, through the attention mechanism and the relation-specific weight matrices, is designed to learn how to differently weight EDUs for the categorization task. This idea of using a weighting scheme along with discourse structure is explored in prior works (Bhatia et al., 2015; Hogenboom et al., 2015), although they are manually designed, rather than learned from training data.

Once we have $\mathbf{v}_{root}$ of a text, the prediction of its category is given by softmax $(\mathbf{W}_o\mathbf{v}_{root} + \mathbf{b})$.

We refer to this model as the FULL model, since it makes use of the entire discourse dependency tree.

3.3 Unlabeled Model

The FULL model based on Equation 1 uses a dependency discourse tree with relations. Because alternate discourse relation labels have been proposed (e.g., Prasad et al., 2008), we seek to measure the effect of these labels. We therefore consider an UNLABELED model based only on the tree structure, without the relations:

$$\mathbf{v}_i = \tanh\left(\mathbf{e}_i + \sum_{j\in children(i)} \alpha_{i,j}\mathbf{v}_j\right). \quad (3)$$

Here, only attention weights are used to compose the children nodes' representations, significantly reducing the number of model parameters.

This UNLABELED model is similar to the depth weighting scheme introduced by Bhatia et al. (2015), which also uses an unlabeled discourse dependency tree, but our attention weights are computed by a function whose parameters are learned. This approach sits squarely between Bhatia et al. (2015) and the flat document structure used by Yang et al. (2016); the UNLABELED model still uses discourse to bias the model toward some content (that which is closer to the tree's root).

3.4 Simpler Variants

We consider two additional baselines that are even simpler. The first, ROOT, uses the discourse dependency structure only to select the root EDU, which is used to represent the entire text: $\mathbf{v}_{root} = \mathbf{e}_{root}$. No composition function is needed. This model variant is motivated by work on document summarization (Yoshida et al., 2014), where the

most central EDU is used to represent the whole text.

The second variant, ADDITIVE, uses all the EDUs with a simple composition function, and does not depend on discourse structure at all: $\mathbf{v}_{root} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i$, where N is the total number of EDUs. This serves as a baseline to test the benefits of discourse, controlling for other design decisions and implementation choices. Although sentence representations $\mathbf{e}_i$ are built in a different way from the work of Yang et al. (2016), this model is quite similar to their HN-AVE model on building document representations.

4 Implementation Details

The parameters of all components of our model (top-level classification, composition, and EDU representation) are learned end-to-end using standard methods. We implement our learning procedure with the `DyNet` package (Neubig et al., 2017).

Preprocessing. For all datasets, we use the same preprocessing steps, mostly following recent work on language modeling (e.g., Mikolov et al., 2010). We lowercased all the tokens and removed tokens that contain only punctuation symbols. We replaced numbers in the documents with a special number token. Low-frequency word types were replaced by UNK; we reduce the vocabulary for each dataset until approximately 5% of tokens are mapped to UNK. The vocabulary sizes after preprocessing are also shown in Table 1.

Discourse parsing. Our model requires the discourse structure for each document. We used DPLP, the RST parser from Ji and Eisenstein (2014), which is one of the best discourse parsers on the RST discourse treebank benchmark (Carlson et al., 2001). It employs a greedy decoding algorithm for parsing, producing 2,000 parses per minute on average on a single CPU. DPLP provides discourse segmentation, breaking a text into EDUs, typically clauses or sentences, based on syntactic parses provided by Stanford CoreNLP. RST trees are converted to dependencies following the method of Yoshida et al. (2014). DPLP as distributed is trained on 347 *Wall Street Journal* articles from the Penn Treebank (Marcus et al., 1993).

Word embeddings. In cases where there are 10,000 or fewer training examples, we used pretrained GloVe word embeddings (Pennington et al., 2014), following previous work on neural discourse processing (Ji and Eisenstein, 2015). For larger datasets, we randomly initialized word embeddings and trained them alongside other model parameters.

Learning and hyperparameters. Online learning was performed with the optimization method and initial learning rate as hyperparameters. To avoid the exploding gradient problem, we used the norm clipping trick with a threshold of $\tau = 5.0$. In addition, dropout rate 0.3 was used on both input and hidden layers to avoid overfitting. We performed grid search over the word vector representation dimensionality, the LSTM hidden state dimensionality (both $\{32, 48, 64, 128, 256\}$), the initial learning rate ($\{0.1, 0.01, 0.001\}$), and the update method (SGD and Adam, Kingma and Ba, 2015). For each corpus, the highest-accuracy combination of these hyperparameters is selected using development data or ten-fold cross validation, which will be specified in section 5.

5 Datasets

We selected five datasets of different sizes and corresponding to varying categorization tasks. Some information about these datasets is summarized in Table 1.

Sentiment analysis on Yelp reviews. Originally from the Yelp Dataset Challenge in 2015, this dataset contains 1.5 million examples. We used the preprocessed dataset from Zhang et al. (2015), which has 650,000 training and 50,000 test examples. The task is to predict an ordinal rating (1–5) from the text of the review. To select the best combination of hyperparameters, we randomly sampled 10% training examples as the development data. We compared with hierarchical attention networks (Yang et al., 2016), which use the normalized attention mechanism on both word and sentence layers with a flat document structure, and provide the state-of-the-art result on this corpus.

Framing dimensions in news articles. The Media Frames Corpus (MFC; Card et al., 2015) includes around 4,200 news articles about immigration from 13 U.S. newspapers over the years 1980–2012. The annotations of these articles are in terms of a set of 15 general-purpose labels, such as ECONOMICS and MORALITY, designed to categorize the emphasis framing applied to the

Dataset	Task	Classes	Number of docs.				Vocab. size
			Total	Training	Development	Test	
Yelp	Sentiment	5	700K	650K	–	50K	10K
MFC	Frames	15	4.2K	–	–	–	7.5K
Debates	Vote	2	1.6K	1,135	105	403	5K
Movies	Sentiment	2	2.0K	–	–	–	5K
Bills	Survival	2	52K	46K	–	6K	10K

Table 1: Information about the five datasets used in our experiments. To compare with prior work, we use different experimental settings. For Yelp and Bill corpora, we use 10% of the training examples as development data. For MFC and Movies corpora, we use 10-fold cross validation and report averages across all folds.

immigration issue within the articles. We focused on predicting the single *primary* frame of each article. The state-of-the-art result on this corpus is from Card et al. (2016), where they used logistic regression together with unigrams, bigrams and Bamman-style personas (Bamman et al., 2014) as features. The best feature combination in their model alongside other hyperparameters was identified by a Bayesian optimization method (Bergstra et al., 2015). To select hyperparameters, we used a small set of examples from the corpus as a development set. Then, we report average accuracy across 10-fold cross validation as in (Card et al., 2016).

Congressional floor debates. The corpus was originally collected by Thomas et al. (2006), and the data split we used was constructed by Yessenalina et al. (2010). The goal is to predict the vote ("yea" or "nay") for the speaker of each speech segment. The most recent work on this corpus is from Yogatama and Smith (2014), which proposed structured regularization methods based on linguistic components, e.g., sentences, topics, and syntactic parses. Each regularization method induces a linguistic bias to improve text classification accuracy, where the best result we repeated here is from the model with sentence regularizers.

Movie reviews. This classic movie review corpus was constructed by Pang and Lee (2004) and includes 1,000 positive and 1,000 negative reviews. On this corpus, we used the standard ten-fold data split for cross validation and reported the average accuracy across folds. We compared with the work from both Bhatia et al. (2015) and Hogenboom et al. (2015), which are two recent works on discourse for sentiment analysis. Bhatia et al. (2015) used a hand-crafted weighting scheme to bias the bag-of-word representations on sentences. Hogenboom et al. (2015) also considered manually-designed weighting schemes and a lexicon-based model as classifier, achieving performance inferior to fully-supervised methods like Bhatia et al. (2015) and ours.

Congressional bill corpus. This corpus, collected by Yano et al. (2012), includes 51,762 legislative bills from the 103rd to 111th U.S. Congresses. The task is to predict whether a bill will survive based on its content. We randomly sampled 10% training examples as development data to search for the best hyperparameters. To our knowledge, the best published results are due to Yogatama and Smith (2014), which is the same baseline as for the congressional floor debates corpus.

6 Experiments

We evaluated all variants of our model on the five datasets presented in section 5, comparing in each case to the published state of the art as well as the most relevant works.

Results. See Table 2. On four out of five datasets, our UNLABELED model (line 8) outperforms past methods. In the case of the very large Yelp dataset, our FULL model (line 9) gives even stronger performance, but not elsewhere, suggesting that it is overparameterized for the smaller datasets. Indeed, on the MFC and Movies tasks, the discourse-ignorant ADDITIVE outperforms the FULL model. On these datasets, the selected FULL model had nearly 20 times as many parameters as the UNLABELED model, which in turn had twice as many parameters as the ADDITIVE.

Method	Yelp	MFC	Debates	Movies	Bills
Prior work					
1. Yang et al. (2016)	71.0	—	—	—	—
2. Card et al. (2016)	—	56.8	—	—	—
3. Yogatama and Smith (2014)	—	—	74.0	—	88.5
4. Bhatia et al. (2015)	—	—	—	82.9	—
5. Hogenboom et al. (2015)	—	—	—	71.9	—
Variants of our model					
6. ADDITIVE	68.5	**57.6**	69.0	82.7	80.1
7. ROOT	54.3	51.2	60.3	68.7	70.5
8. UNLABELED	**71.3**	**58.4**	**75.7**	**83.1**	78.4
9. FULL	**71.8**	56.3	**74.2**	79.5	77.0

Table 2: Test-set accuracy across five datasets. Results from prior work are reprinted from the corresponding publications. Boldface marks performance stronger than the previous state of the art.

This finding demonstrates the benefit of explicit discourse structure—even the output from an imperfect parser—for text categorization in some genres. This benefit is supported by both UNLABELED and FULL, since both of them use discourse structures of texts. The advantage of using discourse information varies on different genres and different corpus sizes. Even though the discourse parser is trained on news text, it still offers benefit to restaurant and movie reviews and to the genre of congressional debates. Even for news text, if the training dataset is small (e.g., MFC), a lighter-weight variant of discourse (UNLABELED) is preferred.

Legislative bills, which have technical legal content and highly specialized conventions (see the supplementary material for an example), are arguably the most distant genre from news among those we considered. On that task, we see discourse working against accuracy. Note that the corpus of bills is more than ten times larger than three cases where our UNLABELED model outperformed past methods, suggesting that the drop in performance is not due to lack of data.

It is also important to notice that the ROOT model performs quite poorly in all cases. This implies that discourse structure is not simply helping by finding a single EDU upon which to make the categorization decision.

Qualitative analysis. Figure 3 shows some example texts from the Yelp Review corpus with their discourse structures produced by DPLP, where the weights were generated with the FULL model. Figures 3(a) and 3(b) are two successful examples of the FULL model. Figure 3(a) shows a simple case with respect to the discourse structure. Figure 3(b) is slightly different—the text in this example may have more than one reasonable discourse structure, e.g., $2D$ could be a child of $2C$ instead of $2A$. In both cases, discourse structures help the FULL model bias to the important sentences.

Figure 3(c), on the other hand, presents a negative example, where DPLP failed to identify the most salient sentence $3F$. In addition, the weights produced by the FULL model do not make much sense, which we suspect the model was confused by the structure. Figure 3(c) also presents a manually-constructed discourse structure on the same text for reference. A more accurate prediction is expected if we use this manually-constructed discourse structure, because it has the appropriate dependency between sentences. In addition, the annotated discourse relations are able to select the right relation-specific composition matrices in FULL model, which are consistent with the training examples.

Effect of parsing performance. A natural question is whether further improvements to RST discourse parsing would lead to even greater gains in text categorization. While advances in discourse parsing are beyond the scope of this paper, we can gain some insight by exploring degradation to the DPLP parser. An easy way to do this is to train it on subsets of the RST discourse treebank. We repeated the conditions described above for our FULL model, training DPLP on 25%, 50%, and 75% of the training set (randomly selected in

From DPLP:

$$1A \xrightarrow[\text{Elaboration}]{0.66} 1B \xrightarrow[\text{Cause}]{0.67} 1C$$

[This store is somewhat convenient but I can never find any workers,]^{1A} [it drives me crazy.]^{1B} [I never come here on the weekends or around holidays anymore.]^{1C}

(a) true label: 2, predicted label: 2

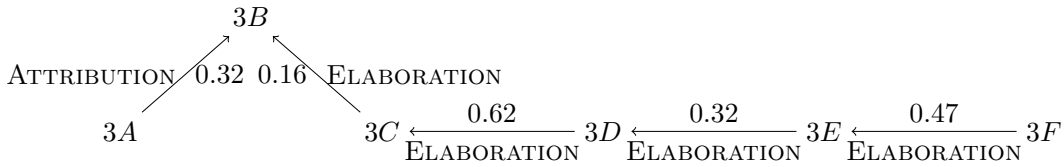
From DPLP:

$$2C \xrightarrow[\text{Evaluation}]{0.70} 2B \xrightarrow[\text{Evaluation}]{0.87} 2A \xleftarrow[\text{Elaboration}]{0.61} 2D$$

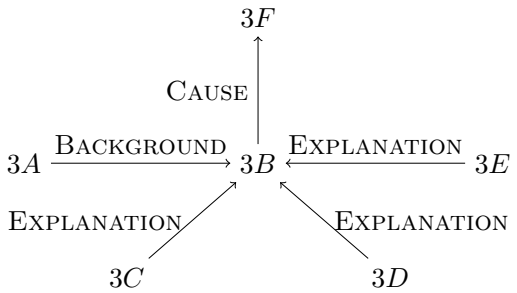
[I love these people.]^{2A} [They are very friendly and always ask about my life.]^{2B} [They remember things I tell them then ask about it the next time I'm in.]^{2C} [Patrick and Lily are the best but everyone there is wonderful in their own ways.]^{2D}

(b) true label: 5, predicted label: 5

From DPLP:



Manually constructed:



[We use to visit this pub 10 years ago because they had a nice english waitress and excellent fish and chips for the price.]^{3A} [However we went back a few weeks ago and were disappointed.]^{3B} [The price of the fish and chip dinner went up and they cut the portion in half.]^{3C} [No one assisted us in putting two tables together we had to do it ourselves.]^{3D} [Two guests wanted a good English hot tea and they didn't brew it in advance.]^{3E} [So we've decided there are newer and better places to eat fish and chips especially up in north phoenix.]^{3F}

(c) true label: 1, predicted label: 3

Figure 3: Some example texts (with light revision for readability) from the Yelp Review corpus and their corresponding dependency discourse parses from DPLP (Ji and Eisenstein, 2014). The numbers on dependency edges are attention weights produced by the Full model.

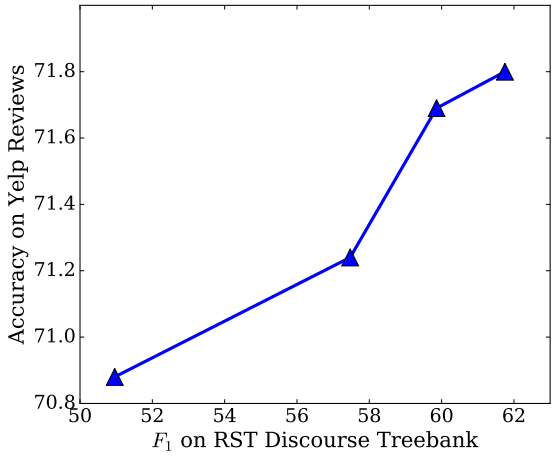


Figure 4: Varying the amount of training data for the discourse parser, we can see how parsing F_1 performance affects accuracy on the Yelp review task.

each case) before re-parsing the data for the sentiment analysis task. We did not repeat the hyperparameter search. In Figure 4, we plot accuracy of the classifier (y-axis) against the F_1 performance of the discourse parser (x-axis). Unsurprisingly, lower parsing performance implies lower classification accuracy. Notably, if the RST discourse treebank were reduced to 25% of its size, our method would underperform the discourse-ignorant model of Yang et al. (2016). While we cannot extrapolate with certainty, these findings suggest that further improvements to discourse parsing, through larger annotated datasets or improved models, could lead to greater gains.

Attention mechanism. In section 3, we contrasted our new attention mechanism (Equation 2), which is inspired by RST's lack of "competition" for salience among satellites, with the attention mechanism used in machine translation (Bahdanau et al., 2015). We consider here a variant of our model with normalized attention:

$$\boldsymbol{\alpha}'_i = \text{softmax}\left(\begin{bmatrix} \vdots \\ \mathbf{v}_j^\top \\ \vdots \end{bmatrix}_{j \in children(i)} \mathbf{W}_\alpha \cdot \mathbf{e}_i\right). \quad (4)$$

The result here is a vector $\boldsymbol{\alpha}'_i$, with one element for each child node $j \in children(i)$, and which sums to one.

On Yelp dateset, this variant of the FULL model achieves 70.3% accuracy (1.5% absolute behind our FULL model), giving empirical support to our theoretically-motivated design decision not to normalize attention. Of course, further architecture improvements may yet be possible.

Discussion. Our findings in this work show the benefit of using discourse structure for text categorization. Although discourse structure strongly improves the performance on most of corpora in our experiments, its benefit is limited particularly by two factors: (1) the state-of-the-art performance on RST discourse parsing; and (2) domain mismatch between the training corpus for a discourse parser and the domain where the discourse parser is used. For the first factor, discourse parsing is still an active research topic in NLP, and may yet improve. The second factor suggests exploring domain adaptation methods or even direct discourse annotation for genres of interest.

7 Related Work

Early work on text categorization often treated text as a bag of words (e.g., Joachims, 1998; Yang and Pedersen, 1997). Representation learning, for example through matrix decomposition (Deerwester et al., 1990) or latent topic variables (Ramage et al., 2009), has been considered to avoid overfitting in the face of sparse data.

The assumption that all parts of a text should influence categorization equally persists even as more powerful representation learners are considered. Zhang et al. (2015) treat a text as a sequence of characters, proposing to a deep convolutional neural network to build text representation. Xiao and Cho (2016) extended that architecture by inserting a recurrent neural network layer between the convolutional layer and the classification layer.

In contrast, our contributions follow Ko et al. (2004), who sought to weight the influence of different parts of an input text on the task. Two works that sought to learn the importance of sentences in a document are Yessenalina et al. (2010) and Yang et al. (2016). The former used a latent variable for the informativeness of each sentence, and the latter used a neural network to learn an attention function. Neither used any linguistic bias, relying only on task supervision to discover the latent variable distribution or attention function. Our work builds the neural network directly on a discourse dependency tree, favoring the most central EDUs over the others but giving the model the ability to overcome this bias.

Another way to use linguistic information was

presented by Yogatama and Smith (2014), who used a bag-of-words model. The novelty in their approach was a data-driven regularization method that encouraged the model to collectively ignore groups of features found to coocur. Most related to our work is their "sentence regularizer," which encouraged the model to try to ignore training-set sentences that were not informative for the task. Discourse structure was not considered.

Discourse for sentiment analysis. Recently, discourse structure has been considered for sentiment analysis, which can be cast as a text categorization problem. Bhatia et al. (2015) proposed two discourse-motivated models for sentiment polarity prediction. One of the models is also based on discourse dependency trees, but using a handcrafted weighting scheme. Our method's attention mechanism automates the weighting.

8 Conclusion

We conclude that automatically-derived discourse structure can be helpful to text categorization, and the benefit increases with the accuracy of discourse parsing. We did not see a benefit for categorizing legislative bills, a text genre whose discourse structure diverges from that of news. These findings motivate further improvements to discourse parsing, especially for new genres.

Acknowledgments

We thank anonymous reviewers and members of Noah's ARK for helpful feedback on this work. We thank Dallas Card and Jesse Dodge for helping prepare the Media Frames Corpus and the Congressional bill corpus. This work was made possible by a University of Washington Innovation Award.

References

Moshe Azar. 1999. Argumentative text as rhetorical structure: An application of rhetorical structure theory. *Argumentation* 13(1):97–114.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

David Bamman, Brendan O'Connor, and Noah A Smith. 2014. Learning latent personas of film characters. In *ACL*.

James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D. Cox. 2015. Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery* 8(1).

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from RST discourse parsing. In *EMNLP*.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The Media Frames Corpus: Annotations of frames across issues. In *ACL*.

Dallas Card, Justin Gross, Amber E. Boydstun, and Noah A. Smith. 2016. Analyzing framing through the casts of characters in the news. In *EMNLP*.

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

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41(6):391.

Alexander Hogenboom, Flavius Frasincar, Franciska de Jong, and Uzay Kaymak. 2015. Using rhetorical structure in sentiment analysis. *Communications of the ACM* 58(7):69–77.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for document-level discourse parsing. In *ACL*.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association of Computational Linguistics* 3:329–344.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *ECML*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. ArXiv:1404.2188.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Youngjoong Ko, Jinwoo Park, and Jungyun Seo. 2004. Improving text categorization using the importance of sentences. *Information Processing & Management* 40(1):65–79.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *EACL*.

William Mann and Sandra Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text* 8(3):243–281.

Daniel Marcu. 1999. Discourse trees are good indicators of importance in text. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 123–136.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. ArXiv:1701.03980.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 271.

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In *NIPS*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *LREC*.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *EMNLP*.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics* 2:207–218.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *EMNLP*.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. ArXiv:1602.00367.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *IWPT*.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *ICML*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*.

Tae Yano, Noah A. Smith, and John D. Wilkerson. 2012. Textual predictors of bill survival in congressional committees. In *NAACL*.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document sentiment classification. In *EMNLP*.

Dani Yogatama and Noah A. Smith. 2014. Linguistic structured sparsity in text categorization. In *ACL*.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *EMNLP*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Adversarial Connective-exploiting Networks for Implicit Discourse Relation Classification

Lianhui Qin[1,2], **Zhisong Zhang**[1,2], **Hai Zhao**[1,2,*], **Zhiting Hu**[3], **Eric P. Xing**[3]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
[3]Carnegie Mellon University
{qinlianhui, zzs2011}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn,
{zhitingh, epxing}@cs.cmu.edu

Abstract

Implicit discourse relation classification is of great challenge due to the lack of connectives as strong linguistic cues, which motivates the use of annotated implicit connectives to improve the recognition. We propose a feature imitation framework in which an implicit relation network is driven to learn from another neural network with access to connectives, and thus encouraged to extract similarly salient features for accurate classification. We develop an adversarial model to enable an adaptive imitation scheme through competition between the implicit network and a rival feature discriminator. Our method effectively transfers discriminability of connectives to the implicit features, and achieves state-of-the-art performance on the PDTB benchmark.

1 Introduction

Discourse relations connect linguistic units such as clauses and sentences to form coherent semantics. Identification of discourse relations can benefit a variety of downstream applications including question answering (Liakata et al., 2013), machine translation (Li et al., 2014), text summarization (Gerani et al., 2014), opinion spam detection (Chen and Zhao, 2015), and so forth.

*Corresponding authors. This paper was partially supported by Cai Yuanpei Program (CSC No. 201304490199 and No. 201304490171), National Natural Science Foundation of China (No. 61170114, No. 61672343 and No. 61272248), National Basic Research Program of China (No. 2013CB329401), Major Basic Research Program of Shanghai Science and Technology Committee (No. 15JC1400103), Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04), and Key Project of National Society Science Foundation of China (No. 15-ZDA041).

Connectives (e.g., *but*, *so*, etc) are one of the most critical linguistic cues for identifying discourse relations. When explicit connectives are present in the text, a simple frequency-based mapping is sufficient to achieve over 85% classification accuracy (Xue et al., 2016; Li et al., 2016). In contrast, *implicit discourse relation recognition* has long been seen as a challenging problem, with the best accuracy so far still lower than 50% (Chen et al., 2015). In the implicit case, discourse relations are not lexicalized by connectives, but to be inferred from relevant sentences (i.e., *arguments*). For example, the following two adjacent sentences Arg1 and Arg2 imply relation *Cause* (i.e., Arg2 is the cause of Arg1).

> [*Arg1*]: Never mind.
>
> [*Arg2*]: You already know the answer.
>
> [*Implicit connective*]: Because
>
> [*Discourse relation*]: Cause

Various attempts have been made to directly infer underlying relations by modeling the semantics of the arguments, ranging from feature-based methods (Lin et al., 2009; Pitler et al., 2009) to the very recent end-to-end neural models (Chen et al., 2016a; Qin et al., 2016c). Despite impressive performance, the absence of strong explicit connective cues has made the inference extremely hard and hindered further improvement. In fact, even the human annotators would make use of connectives to aid relation annotation. For instance, the popular Penn Discourse Treebank (PDTB) benchmark data (Prasad et al., 2008) was annotated by first inserting a connective expression (i.e., *implicit connective*, as shown in the above example) manually, and determining the abstract relation by combining both the implicit connective and contextual semantics.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1006–1017
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1093

Therefore, the huge performance gap between explicit and implicit parsing (namely, 85% vs 50%), as well as the human annotation practice, strongly motivates to incorporate connective information to guide the reasoning process. This paper aims to advance implicit parsing by making use of annotated implicit connectives available in training data. Few recent work has explored such combination. Zhou et al. (2010) developed a two-step approach by first predicting implicit connectives whose sense is then disambiguated to obtain the relation. However, the pipeline approach usually suffers from error propagation, and the method itself has relied on hand-crafted features which do not necessarily generalize well. Other research leveraged explicit connective examples for data augmentation (Rutherford and Xue, 2015; Braud and Denis, 2015; Ji et al., 2015; Braud and Denis, 2016). Our work is orthogonal and complementary to this line.

In this paper, we propose a novel neural method that incorporates implicit connectives in a principled *adversarial* framework. We use deep neural models for relation classification, and take the intuition that, sentence arguments integrated with connectives would enable highly discriminative neural features for accurate relation inference, and an ideal implicit relation classifier, even though without access to connectives, should mimic the connective-augmented reasoning behavior by extracting similarly salient features. We therefore setup a secondary network in addition to the implicit relation classifier, building upon connective-augmented inputs and serving as a feature learning model for the implicit classifier to emulate.

Methodologically, however, feature imitation in our problem is challenging due to the semantic gap induced by adding the connective cues. It is necessary to develop an adaptive scheme to flexibly drive learning and transfer discriminability. We devise a novel adversarial approach which enables a self-calibrated imitation mechanism. Specifically, we build a discriminator which distinguishes between the features by the two counterpart networks. The implicit relation network is then trained to correctly classify relations and simultaneously to fool the discriminator, resulting in an adversarial framework. The adversarial mechanism has been an emerging method in different context, especially for image generation (Goodfellow et al., 2014) and domain adaptation (Ganin et al., 2016; Chen et al., 2016c). Our adversarial framework is unique to address neural feature emulation between two models. Besides, to the best of our knowledge, this is the first adversarial approach in the context of discourse parsing. Compared to previous connective exploiting work (Zhou et al., 2010; Xu et al., 2012), our method provides a new integration paradigm and an end-to-end procedure that avoids inefficient feature engineering and error propagation.

Our method is evaluated on the PDTB 2.0 benchmark in a variety of experimental settings. The proposed adversarial model greatly improves over standalone neural models and previous best-performing approaches. We also demonstrate that our implicit recognition network successfully imitates and extracts crucial hidden representations.

We begin by briefly reviewing related work in section 2. Section 3 presents the proposed adversarial model. Section 4 shows substantially improved experimental results over previous methods. Section 5 discusses extensions and future work.

2 Related Work

2.1 Implicit Discourse Relation Recognition

There has been a surge of interest in implicit discourse parsing since the release of PDTB (Prasad et al., 2008), the first large discourse corpus distinguishing implicit examples from explicit ones. A large set of work has focused on direct classification based on observed sentences, including structured methods with linguistically-informed features (Lin et al., 2009; Pitler et al., 2009; Zhou et al., 2010), end-to-end neural models (Qin et al., 2016b,c; Chen et al., 2016a; Liu and Li, 2016), and combined approaches (Ji and Eisenstein, 2015; Ji et al., 2016). However, the lacking of connective cues makes learning purely from contextual semantics full of challenges.

Prior work has attempted to leverage connective information. Zhou et al. (2010) also incorporate implicit connectives, but in a pipeline manner by first predicting the implicit connective with a language model and determining discourse relation accordingly. Instead of treating implicit connectives as intermediate prediction targets which can suffer from error propagation, we use the connectives to induce highly discriminative features to guide the learning of an implicit network, serving as an adaptive regularization mechanism for en-

hanced robustness and generalization. Our framework is also end-to-end, avoiding costly feature engineering. Another notable line aims at adapting explicit examples for data synthesis (Biran and McKeown, 2013; Rutherford and Xue, 2015; Braud and Denis, 2015; Ji et al., 2015), multi-task learning (Lan et al., 2013; Liu et al., 2016), and word representation (Braud and Denis, 2016). Our work is orthogonal and complementary to these methods, as we use implicit connectives which have been annotated for implicit examples.

2.2 Adversarial Networks

Deep neural networks have gained impressive success in various natural language processing tasks (Wang et al., 2016; Zhang et al., 2016b; Cai et al., 2017), in which adversarial networks have been shown especially effective in deep generative modeling (Goodfellow et al., 2014) and domain adaptation (Ganin et al., 2016). Generative adversarial nets (Goodfellow et al., 2014) learn to produce realistic samples through competition between a generator and a real/fake discriminator. Professor forcing (Lamb et al., 2016) applies a similar idea to improve long-term generation of a recurrent neural language model. Other approaches (Chen et al., 2016b; Hu et al., 2017; Liang et al., 2017) extend the framework for controllable image/text generation. Li et al. (2015); Salimans et al. (2016) propose feature matching which trains generators to match the statistics of real/fake examples. Their features are extracted by the discriminator rather than the classifier networks as in our case. Our work differs from the above since we consider the context of discriminative modeling. Adversarial domain adaptation forces a neural network to learn domain-invariant features using a classifier that distinguishes the domain of the network's input data based on the hidden feature. Our adversarial framework is distinct in that besides the implicit relation network we construct a second neural network serving as a teacher model for feature emulation.

To the best of our knowledge, this is the first to employ the idea of adversarial learning in the context of discourse parsing. We propose a novel connective exploiting scheme based on feature imitation, and to this end derive a new adversarial framework, achieving substantial performance gain over existing methods. The proposed approach is generally applicable to other tasks for utilizing any indicative side information. We give more discussions in section 5.

3 Adversarial Method

Discourse connectives are key indicators for discourse relation. In the annotation procedure of the PDTB implicit relation benchmark, annotators inserted implicit connective expressions between adjacent sentences to lexicalize abstract relations and help with final decisions. Our model aims at making full use of the provided implicit connectives at training time to regulate learning of implicit relation recognizer, encouraging extraction of highly discriminative semantics from raw arguments, and improving generalization at test time. Our method provides a novel adversarial framework that leverages connective information in a flexible adaptive manner, and is efficiently trained end-to-end through standard back-propagation.

The basic idea of the proposed approach is simple. We want our implicit relation recognizer, which predicts the underlying relation of sentence arguments without discourse connective, to have prediction behaviors close to a *connective-augmented* relation recognizer which is provided with a discourse connective in addition to the arguments. The connective-augmented recognizer is in analogy to an annotator with the help of connectives as in the human annotation process, and the implicit recognizer would be improved by learning from such an "informed" annotator. Specifically, we want the latent features extracted by the two models to match as closely as possible, which explicitly transfers the discriminability of the connective-augmented representations to implicit ones.

To this end, instead of manually selecting a closeness metric, we take advantage of the adversarial framework by constructing a two-player zero-sum game between the implicit recognizer and a rival discriminator. The discriminator attempts to distinguish between the features extracted by the two relation models, while the implicit relation model is trained to maximize the accuracy on implicit data, and at the same time to confuse the discriminator.

In the next we first present the overall architecture of the proposed approach (section 3.1), then develop the training procedure (section 3.2). The components are realized as deep (convolutional) neural networks, with detailed modeling choices

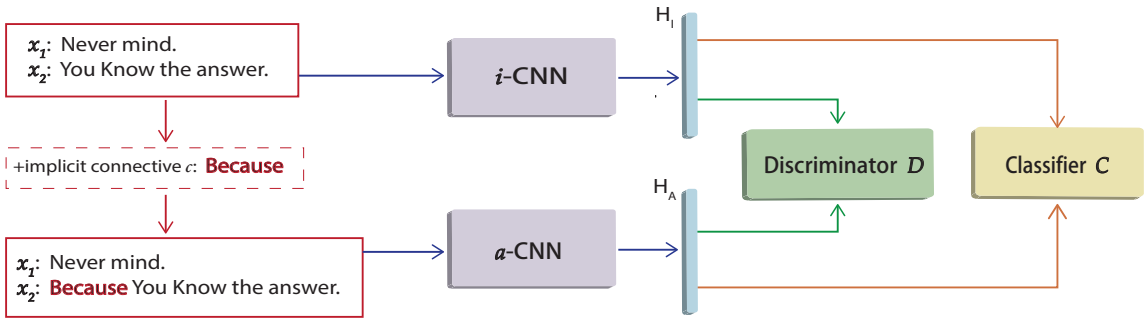


Figure 1: Architecture of the proposed method. The framework contains three main components: 1) an implicit relation network *i-CNN* over raw sentence arguments, 2) a connective-augmented relation network *a-CNN* whose inputs are augmented with implicit connectives, and 3) a discriminator distinguishing between the features by the two networks. The features are fed to the final classifier for relation classification. The discriminator and *i-CNN* form an adversarial pair for feature imitation. At test time, the implicit network *i-CNN* with the classifier is used for prediction.

discussed in section 3.3.

3.1 Model Architecture

Let $(\boldsymbol{x}, y)$ be a pair of input and output of implicit relation classification, where $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ is a pair of sentence arguments, and y is the underlying discourse relation. Each training example also includes an annotated implicit connective c that best expresses the relation. Figure 1 shows the architecture of our framework.

The neural model for implicit relation classification (*i-CNN* in the figure) extracts latent representation from the arguments, denoted as $H_I(\boldsymbol{x}_1, \boldsymbol{x}_2)$, and feeds the feature into a classifier C for final prediction $C(H_I(\boldsymbol{x}_1, \boldsymbol{x}_2))$. For ease of notation, we will also use $H_I(\boldsymbol{x})$ to denote the latent feature on data $\boldsymbol{x}$.

The second relation network (*a-CNN*) takes as inputs the sentence arguments along with an implicit connective, to induce the connective-augmented representation $H_A(\boldsymbol{x}_1, \boldsymbol{x}_2, c)$, and obtains relation prediction $C(H_A(\boldsymbol{x}_1, \boldsymbol{x}_2, c))$. Note that the same final classifier C is used for both networks, so that the feature representations by the two networks are ensured to be within the same semantic space, enabling feature emulation as presented shortly.

We further pair the implicit network with a rival discriminator D to form our adversarial game. The discriminator is to differentiate between the reasoning behaviors of the implicit network *i-CNN* and the augmented network *a-CNN*. Specifically, D is a binary classifier that takes as inputs a latent feature H derived from either *i-CNN* or *a-CNN* given appropriate data (where implicit connectives is either missing or present, respectively). The output $D(H)$ estimates the probability that H comes from the connective-augmented *a-CNN* rather than *i-CNN*.

3.2 Training Procedure

The system is trained through an alternating optimization procedure that updates the components in an interleaved manner. In this section, we first present the training objective for each component, and then give the overall training algorithm.

Let $\boldsymbol{\theta}_D$ denote the parameters of the discriminator. The training objective of D is straightforward, i.e., to maximize the probability of correctly distinguishing the input features:

$$\max_{\boldsymbol{\theta}_D} \mathcal{L}_D = \mathbb{E}_{(\boldsymbol{x},c,y)\sim\text{data}}\Big[\log D(H_A(\boldsymbol{x}, c); \boldsymbol{\theta}_D) + \log(1 - D(H_I(\boldsymbol{x}); \boldsymbol{\theta}_D))\Big], \quad (1)$$

where $\mathbb{E}_{(\boldsymbol{x},c,y)\sim\text{data}}[\cdot]$ denotes the expectation in terms of the data distribution.

We denote the parameters of the implicit network *i-CNN* and the classifier C as $\boldsymbol{\theta}_I$ and $\boldsymbol{\theta}_C$, respectively. The model is then trained to (a) correctly classify relations in training data and (b) produce salient features close to connective-augmented ones. The first objective can be fulfilled by minimizing the usual cross-entropy loss:

$$\mathcal{L}_{I,C}(\boldsymbol{\theta}_I, \boldsymbol{\theta}_C) = \mathbb{E}_{(\boldsymbol{x},y)\sim\text{data}}\Big[J\big(C(H_I(\boldsymbol{x}; \boldsymbol{\theta}_I); \boldsymbol{\theta}_C), y\big)\Big], \quad (2)$$

Algorithm 1 Adversarial Model for Implicit Recognition

Input: Training data $\{(\boldsymbol{x}, c, y)_n\}$
Parameters: λ_1, λ_2 – balancing parameters

1: Initialize $\{\boldsymbol{\theta}_I, \boldsymbol{\theta}_C\}$ and $\{\boldsymbol{\theta}_A\}$ by minimizing Eq.(2) and Eq.(4), respectively
2: **repeat**
3: Train the discriminator through Eq.(1)
4: Train the relation models through Eq.(5)
5: **until** convergence

Output: Adversarially enhanced implicit relation network *i-CNN* with classifier C for prediction

where $J(\boldsymbol{p}, y) = -\sum_k \mathbb{I}(y = k) \log p_k$ is the cross-entropy loss between predictive distribution $\boldsymbol{p}$ and ground-truth label y. We achieve objective (b) by minimizing the discriminator's chance of correctly telling apart the features:

$$\mathcal{L}_I(\boldsymbol{\theta}_I) = \mathbb{E}_{\boldsymbol{x}\sim\text{data}}\Big[\log\big(1 - D(H_I(\boldsymbol{x}; \boldsymbol{\theta}_I))\big)\Big]. \quad (3)$$

The parameters of the augmented network *a-CNN*, denoted as $\boldsymbol{\theta}_A$, can be learned by simply fitting to the data, i.e., minimizing the cross-entropy loss as follows:

$$\mathcal{L}_A(\boldsymbol{\theta}_A) = \mathbb{E}_{(\boldsymbol{x},c,y)\sim\text{data}}\Big[J\big(C(H_A(\boldsymbol{x}, c; \boldsymbol{\theta}_A)), y\big)\Big]. \quad (4)$$

As mentioned above, here we use the same classifier C as for the implicit network, forcing a unified feature space of both networks. We combine the above objectives Eqs.(2)-(4) of the relation classifiers and minimize the joint loss:

$$\min_{\boldsymbol{\theta}_I, \boldsymbol{\theta}_A, \boldsymbol{\theta}_C} \mathcal{L}_{I,A,C} = \mathcal{L}_{I,C}(\boldsymbol{\theta}_I, \boldsymbol{\theta}_C) + \lambda_1 \mathcal{L}_I(\boldsymbol{\theta}_I) + \lambda_2 \mathcal{L}_A(\boldsymbol{\theta}_A), \quad (5)$$

where λ_1 and λ_2 are two balancing parameters calibrating the weights of the classification losses and the feature-regulating loss. In practice, we pre-train the implicit and augmented networks independently by minimizing Eq.(2) and Eq.(4), respectively. In the adversarial training process, we found setting $\lambda_2 = 0$ gives stable convergence. That is, the connective-augmented features are fixed after the pre-training stage.

Algorithm 1 summarizes the training procedure, where we interleave the optimization of Eq.(1) and Eq.(5) at each iteration. More practical details are provided in section 4. We instantiate all modules as neural networks (section 3.3) which are differentiable, and perform the optimization efficiently through standard stochastic gradient descent and back-propagation.

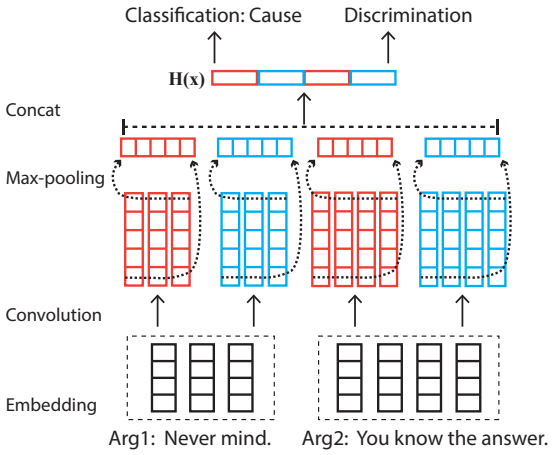


Figure 2: Neural structure of *i-CNN*. Two sets of convolutional filters are shown, with the corresponding features in red and blue, respectively. The weights of the filters on two input arguments are tied.

Through Eq.(1) and Eq.(3), the discriminator and the implicit relation network follow a minimax competition, which drives both to improve until the implicit feature representations are close to the connective-augmented latent representations, encouraging the implicit network to extract highly discriminative features from raw sentence arguments for relation classification. Alternatively, we can see Eq.(3) as an *adaptive* regularization on the implicit model, which, compared to pre-fixed regularizors such as ℓ_2-regularization, provides a more flexible, self-calibrated mechanism to improve generalization ability.

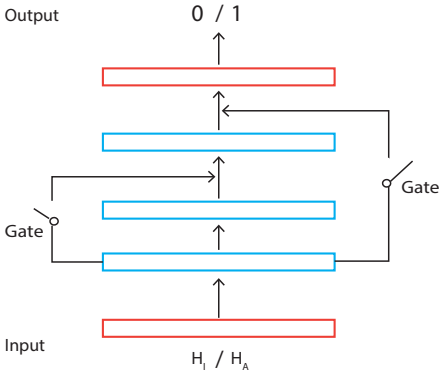


Figure 3: Neural structure of the discriminator D.

3.3 Component Structures

We have presented our adversarial framework for implicit relation classification. We now discuss the model realization of each component. All components of the framework are parameterized with neural networks. Distinct roles of the modules in the framework lead to different modeling choices.

Relation Classification Networks Figure 2 illustrates the structure of the implicit relation network *i-CNN*. We use a convolutional network as it is a common architectural choice for discourse parsing. The network takes as inputs the word vectors of the tokens in each sentence argument, and maps each argument to intermediate features through a shared convolutional layer. The resulting representations are then concatenated and fed into a max pooling layer to select most salient features as the final representation. The final classifier C is a simple fully-connected layer followed by a softmax classifier.

The connective-augmented network *a-CNN* has a similar structure as *i-CNN*, wherein implicit connective is appended to the second sentence as input. The key difference from *i-CNN* is that here we adopt *average k-max* pooling, which takes the average of the top-k maximum values in each pooling window. The reason is to prevent the network from solely selecting the connective induced features (which are typically the most salient features) which would be the case when using max pooling, but instead force it to also attend to contextual features derived from the arguments. This facilitates more homogeneous output features of the two networks, and thus facilitates feature imitation. In all the experiments we fixed $k = 2$.

Discriminator The discriminator is a binary classifier to identify the correct source of an input feature vector. To make it a strong rival to the feature imitating network (*i-CNN*), we model the discriminator as a multi-layer perceptron (MLP) enhanced with gated mechanism for efficient information flow (Srivastava et al., 2015; Qin et al., 2016c), as shown in Figure 3.

4 Experiments

We demonstrate the effectiveness of our approach both quantitatively and qualitatively with extensive experiments. We evaluate prediction performance on the PDTB benchmark in different settings. Our method substantially improves over a diverse set of previous models, especially in the practical multi-class classification task. We perform in-depth analysis of the model behaviors, and show our adversarial framework successfully enables the implicit relation model to imitate and learn discriminative features.

4.1 Experiment Setup

We use PDTB 2.0[1], one of the largest manually annotated discourse relation corpus. The dataset contains 16,224 implicit relation instances in total, with three levels of senses: Level-1 *Class*, Level-2 *Type*, and Level-3 *Subtypes*. The 1st level consists of four major relation *Class*es: COMPARISON, CONTINGENCY, EXPANSION and TEMPORAL. The 2nd level contains 16 *Type*s.

To make extensive comparison with prior work of implicit discourse relation classification, we evaluate on two popular experimental settings: 1) multi-class classification for 2nd-level types (Lin et al., 2009; Ji and Eisenstein, 2015), and 2) one-versus-others binary classifications for 1st-level classes (Pitler et al., 2009). We describe the detailed configurations in the following respective sections. We will focus our analysis on the multi-class classification setting, which is most realistic in practice and serves as a building block for a complete discourse parser such as that for the shared tasks of CoNLL-2015 and 2016 (Xue et al., 2015, 2016).

Model Training Here we provide the detailed architecture configurations of each component we used in the experiments.

- Throughout the experiments *i-CNN* and *a-CNN* contain 3 sets of convolutional filters with the filter sizes selected on the dev set. Table 1 lists the filter configurations of the convolutional layer in *i-CNN* and *a-CNN* in different tasks. As described in section 3.3, following the convolutional layer is a max pooling layer in *i-CNN*, and an average k-max pooling layer with $k = 2$ in *a-CNN*.
- The final single-layer classifier *C* contains 512 neurons with *tanh* activation function.
- The discriminator *D* consists of 4 fully-connected layers, with 2 gated pathways from layer 1 to layer 3 and layer 4 (see Figure 3).

[1] http://www.seas.upenn.edu/∼pdtb/

Task	Filter sizes	Filter number
PDTB-Lin	2, 4, 8	3×256
PDTB-Ji	2, 5, 10	3×256
One-vs-all	2, 5, 10	3×1024

Table 1: The convolutional architectures of *i-CNN* and *a-CNN* in different tasks (section 4). For example, in PDTB-Lin, we use 3 sets of filters, each of which is of size 2, 4, and 8, respectively; and each set has 256 filters.

The size of each layer is set to 1024 and is fixed in all the experiments.

- We set the dimension of the input word vectors to 300 and initialize with pre-trained word2vec (Mikolov et al., 2013). The maximum length of sentence argument is set to 80. Truncation and zero-padding are applied when necessary.

All experiments were performed on a TITAN-X GPU and 128GB RAM, with neural implementation based on Tensorflow[2].

For adversarial model training, it is critical to keep balance between the progress of the two players. We use a simple strategy which at each iteration optimizes the discriminator and the implicit relation network on a randomly-sampled mini-batch. We found this is enough to stabilize the training. The neural parameters are trained using AdaGrad (Duchi et al., 2011) with an initial learning rate of 0.001. For the balancing parameters in Eq.(5), we set $\lambda_1 = 0.1$, while $\lambda_2 = 0$. That is, after the initialization stage the weights of the connective-augmented network *a-CNN* are fixed. This has been shown capable of giving stable and good predictive performance for our system.

4.2 Implicit Relation Classification

We will mainly focus on the general multi-class classification problem in two alternative settings adopted in prior work, showing the superiority of our model over previous state of the arts. We perform in-depth comparison with carefully designed baselines, providing empirical insights into the working mechanism of the proposed framework. For broader comparisons we also report the performance in the one-versus-all setting.

[2]https://www.tensorflow.org

	Model	PDTB-Lin	PDTB-Ji
1	Word-vector	34.07	36.86
2	CNN	43.12	44.51
3	Ensemble	42.17	44.27
4	Multi-task	43.73	44.75
5	ℓ_2-reg	44.12	45.33
6	Lin et al. (2009)	40.20	-
7	Lin et al. (2009) +Brown clusters	-	40.66
8	Ji and Eisenstein (2015)	-	44.59
9	Qin et al. (2016a)	43.81	45.04
10	Ours	**44.65**	**46.23**

Table 2: Accuracy (%) on the test sets of the PDTB-Lin and PDTB-Ji settings for **multi-class classification**. Please see the text for more details.

Multi-class Classifications

We first adopt the standard PDTB splitting convention following (Lin et al., 2009), denoted as PDTB-Lin, where sections 2-21, 22, and 23 are used as training, dev, and test sets, respectively. The most frequent 11 types of relations are selected in the task. During training, instances with more than one annotated relation types are considered as multiple instances, each of which has one of the annotations. At test time, a prediction that matches one of the gold types is considered as correct. The test set contains 766 examples. More details are in (Lin et al., 2009). An alternative, slightly different multi-class setting is used in (Ji and Eisenstein, 2015), denoted as PDTB-Ji, where sections 2-20, 0-1, and 21-22 are used as training, dev, and test sets, respectively. The resulting test set contains 1039 examples. We also evaluate in this setting for thorough comparisons.

Table 2 shows the classification accuracy in both of the settings. We see that our model (Row 10) achieves state-of-the-art performance, greatly outperforming previous methods (Rows 6-9) with various modeling paradigms, including the linguistic feature-based model (Lin et al., 2009), pure neural methods (Qin et al., 2016c), and combined approach (Ji and Eisenstein, 2015).

To obtain better insights into the working mechanism of our method, we further compare with a set of carefully selected baselines as shown in Rows 1-5. 1) "Word-vector" sums over the word vectors for sentence representation, showing the base effect of word embeddings. 2) "CNN" is a standalone convolutional net having the exact same architecture with our implicit rela-

tion network. Our model trained within the proposed framework provides significant improvement, showing the benefits of utilizing implicit connectives at training time. 3) "Ensemble" has the same neural architecture with the proposed framework except that the input of *a-CNN* is not augmented with implicit connectives. This essentially is an ensemble of two implicit recognition networks. We see that the method performs even inferior to the single CNN model. This further confirms the necessity of exploiting connective information. 4) "Multi-task" is the convolutional net augmented with an additional task of simultaneously predicting the implicit connectives based on the network features. As a straightforward way of incorporating connectives, we see that the method slightly improves over the stand-alone CNN, while falling behind our approach with a large margin. This indicates that our proposed feature imitation is a more effective scheme for making use of implicit connectives. 5) At last, "ℓ_2-reg" also implements feature mimicking by imposing an ℓ_2 distance penalty between the implicit relation features and connective-augmented features. We see that the simple model has obtained improvement over previous best-performing systems in both settings, further validating the idea of imitation. However, in contrast to the fixed ℓ_2 regularization, our adversarial framework provides an adaptive mechanism, which is more flexible and performs better as shown in the table.

Model	COMP.	CONT.	EXP.	TEMP.
Pitler et al. (2009)	21.96	47.13	-	16.76
Qin et al. (2016c)	**41.55**	**57.32**	71.50	35.43
Zhang et al. (2016a)	35.88	50.56	71.48	29.54
Zhou et al. (2010)	31.79	47.16	70.11	20.30
Liu and Li (2016)	36.70	54.48	70.43	**38.84**
Chen et al. (2016a)	40.17	54.76	-	31.32
Ours	40.87	54.56	**72.38**	36.20

Table 3: Comparisons of F_1 scores (%) for binary classification.

One-versus-all Classifications

We also report the results of four one-versus-all binary classifications for more comparisons with prior work. We follow the conventional experimental setting (Pitler et al., 2009) by selecting sections 2-20, 21-22, and 0-1 as training, dev, and test sets. Table 4 lists the statistics of the data.

Following previous work, Table 3 reports the F1 scores. Our method outperforms most of the prior systems in all the tasks. We achieve state-of-the-art performance in recognition of the Expansion relation, and obtain comparable scores with the best-performing methods in each of the other relations, respectively. Notably, our feature imitation scheme greatly improves over (Zhou et al., 2010) which leverages implicit connectives as an intermediate prediction task. This provides additional evidence for the effectiveness of our approach.

Relation	Train	Dev	Test
Comparison	1942/1942	197/986	152/894
Contigency	3342/3342	295/888	279/767
Expansion	7004/7004	671/512	574/472
Temporal	760/760	64/1119	85/961

Table 4: Distributions of positive and negative instances from the train/dev/test sets in four binary relation classification tasks.

4.3 Qualitative Analysis

We now take a closer look into the modeling behavior of our framework, by investigating the process of the adversarial game during training, as well as the feature imitation effects.

Figure 4 demonstrates the training progress of different components. The *a-CNN* network keeps high predictive accuracy as implicit connectives are given, showing the importance of connective cues. The rise-and-fall patterns in the accuracy of the discriminator clearly show its competition with the implicit relation network *i-CNN* as training goes. At first few iterations the accuracy of the discriminator increases quickly to over 0.9, while at late stage the accuracy drops to around 0.6, showing that the discriminator is getting confused by *i-CNN* (an accuracy of 0.5 indicates full confusion). The *i-CNN* network keeps improving in terms of implicit relation classification accuracy, as it is gradually fitting to the data and simultaneously learning increasingly discriminative features by mimicking *a-CNN*. The system exhibits similar learning patterns in the two different settings, showing the stability of the training strategy.

We finally visualize the output feature vectors of *i-CNN* and *a-CNN* using the t-SNE method (Maaten and Hinton, 2008) in Figure 5. Without feature imitation, the extracted features by the two networks are clearly separated (Figure 5(a)). In contrast, as shown in Figures 5(b)-(c), the feature vectors are increasingly mixed as training proceeds. Thus our framework has suc-

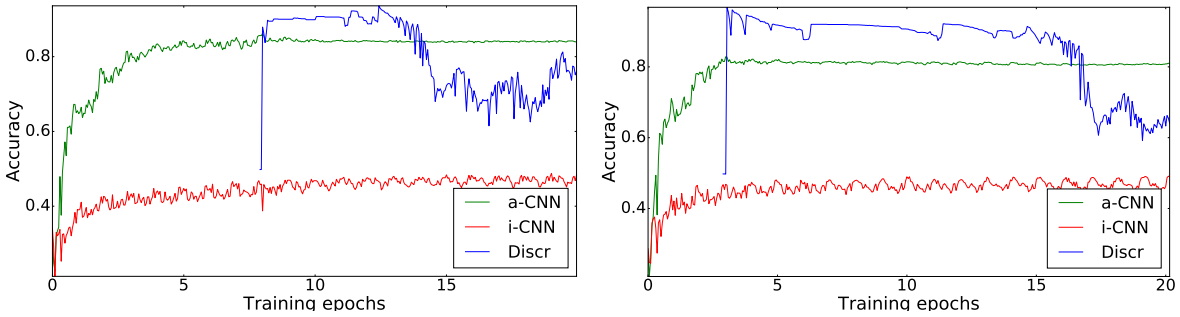


Figure 4: (Best viewed in colors.) Test-set performance of three components over training epochs. Relation networks *a-CNN* and *i-CNN* are measured with multi-class classification accuracy (with or without implicit connectives, respectively), while the discriminator is evaluated with binary classification accuracy. **Top**: the PDTB-Lin setting (Lin et al., 2009), where first 8 epochs are for initialization stage (thus the discriminator is fixed and not shown); **Bottom**: the PDTB-Ji setting (Ji and Eisenstein, 2015), where first 3 epochs are for initialization.

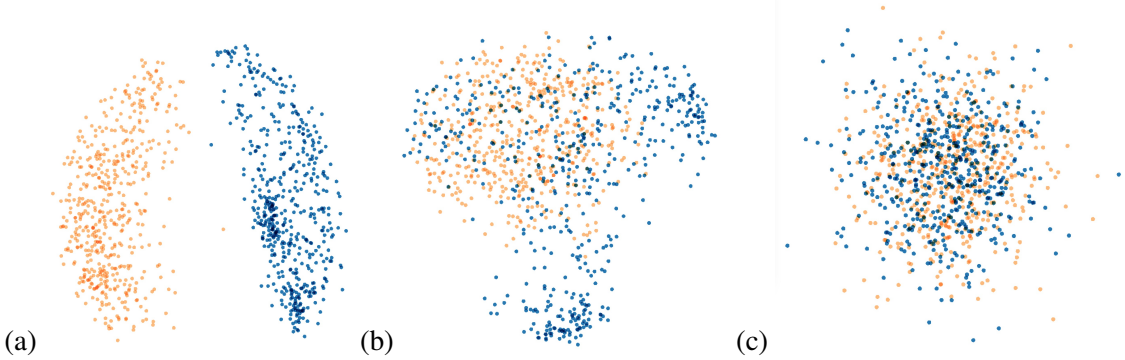


Figure 5: (Best viewed in colors.) Visualizations of the extracted hidden features by the implicit relation network *i-CNN* (blue) and connective-augmented relation network *a-CNN* (orange), in the multi-class classification setting (Lin et al., 2009). **(a)** Two networks are trained without adversary (with shared classifier); **(b)** Two networks are trained within our framework at epoch 10; **(c)** at epoch 20. The implicit relation network successfully imitates the connective-augmented features through the adversarial game. Visualization is conducted with the t-SNE algorithm (Maaten and Hinton, 2008).

cessfully driven *i-CNN* to induce similar representations with *a-CNN*, even though connectives are not present.

5 Discussions

We have developed an adversarial neural framework that facilitates an implicit relation network to extract highly discriminative features by mimicking a connective-augmented network. Our method achieved state-of-the-art performance for implicit discourse relation classification. Besides implicit connective examples, our model can naturally exploit enormous explicit connective data to further improve discourse parsing.

The proposed adversarial feature imitation scheme is also generally applicable to other context to incorporate indicative side information available at training time for enhanced inference. Our framework shares a similar spirit of the iterative knowledge distillation method (Hu et al., 2016a,b) which train a "student" network to mimic the classification behavior of a knowledge-informed "teacher" network. Our approach encourages imitation on the feature level instead of the final prediction level. This allows our approach to apply to regression tasks, and more interestingly, the context in which the student and teacher networks have different prediction outputs, e.g., performing different tasks, while transferring knowledge between each other can be beneficial. Besides, our adversarial mechanism provides an adaptive metric to measure and drive the imitation procedure.

References

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL, Volume 2: Short Papers)*. Sofia, Bulgaria, pages 69–73.

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal, pages 2201–2211.

Chloé Braud and Pascal Denis. 2016. Learning connective-based word representations for implicit discourse relation identification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, Texas, pages 203–213.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada.

Changge Chen, Peilu Wang, and Hai Zhao. 2015. Shallow discourse parsing using constituent parsing tree. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task (CONLL)*. Beijing, China, pages 37–41.

Changge Chen and Hai Zhao. 2015. Deceptive opinion spam detection using deep level linguistic feature. In *The 4th CCF Conference on Natural Language Processing and Chinese Computing (NLPCC 2015), LNCS*. Nanchang, China, volume 9362, pages 465–474.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016a. Implicit discourse relation detection via a deep architecture with gated relevance network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL Volume 1: Long Papers)*. Berlin, Germany, pages 1726–1735.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016b. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*. pages 2172–2180.

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Weinberger, and Claire Cardie. 2016c. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614* .

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research* 17(59):1–35.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, T. Raymond Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1602–1613.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. pages 2672–2680.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric P Xing. 2016a. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 2410–2420.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Controllable text generation. *arXiv preprint arXiv:1703.00955* .

Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P Xing. 2016b. Deep neural networks with massive learned knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, USA, pages 1670–1679.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics (TACL)* 3:329–344.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*. San Diego, California, pages 332–342.

Yangfeng Ji, Gongbo Zhang, and Jacob Eisenstein. 2015. Closing the gap: Domain adaptation from explicit to implicit discourse relations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal, pages 2219–2224.

Alex M Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*. pages 4601–4609.

Man Lan, Yu Xu, and Zhengyu Niu. 2013. Leveraging synthetic discourse data via multi-task learning for implicit discourse relation recognition. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL, Volume 1: Long Papers)*. Sofia, Bulgaria, pages 476–485.

Junyi Jessy Li, Marine Carpuat, and Ani Nenkova. 2014. Assessing the discourse factors that influence the quality of machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL, Volume 2: Short Papers)*. Baltimore, Maryland, pages 283–288.

Yujia Li, Kevin Swersky, and Richard S Zemel. 2015. Generative moment matching networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Lille, France, pages 1718–1727.

Zhongyi Li, Hai Zhao, Chenxi Pang, Lili Wang, and Huan Wang. 2016. A constituent syntactic parse tree based discourse parser. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task (CONLL)*. Berlin, Germany, pages 60–64.

Maria Liakata, Simon Dobnik, Shyamasree Saha, Colin Batchelor, and Dietrich Rebholz-Schuhmann. 2013. A discourse-driven content model for summarising scientific articles evaluated in a complex question answering task. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Seattle, Washington, USA, pages 747–757.

Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan, and Eric P Xing. 2017. Recurrent topic-transition GAN for visual paragraph generation. *arXiv preprint arXiv:1703.07022* .

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Singapore, pages 343–351.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, Texas, pages 1224–1233.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. *arXiv preprint arXiv:1603.02776* .

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9:2579–2605.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (3)*. South Lake Tahoe, Nevada, USA, pages 3111–3119.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of he Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. Suntec, Singapore, pages 683–691.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The Penn discourse treebank 2.0. In *The sixth international conference on Language Resources and Evaluation (LREC)*. Marrakech, Morocco, pages 2961–2968.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016a. Implicit discourse relation recognition with context-aware character-enhanced embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan, pages 1914–1924.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016b. Shallow discourse parsing using convolutional neural network. In *Proceedings of the CoNLL-16 shared task*. Berlin, Germany, pages 70–77.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016c. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, Texas, pages 2263–2270.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*. Denver, Colorado, pages 799–808.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. pages 2226–2234.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* .

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2016. Learning distributed word representations for bidirectional LSTM recurrent neural network. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. San Diego, California, pages 527–533.

Yu Xu, Man Lan, Yue Lu, Zheng Yu Niu, and Chew Lim Tan. 2012. Connective prediction using machine learning for implicit discourse relation classification. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*. Brisbane, Australia, pages 1–8.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The CoNLL-2015 shared task on shallow discourse parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*. Beijing, China, pages 1–16.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Bonnie Webber, Attapol Rutherford, Chuan Wang, and Hongmin Wang. 2016. The CoNLL-2016 shared task on shallow discourse parsing. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*. Berlin, Germany, pages 1–19.

Biao Zhang, Deyi Xiong, jinsong su, Qun Liu, Rongrong Ji, Hong Duan, and Min Zhang. 2016a. Variational neural discourse relation recognizer. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, Texas, pages 382–391.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016b. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 1382–1392.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics (CoLING 2010)*. Beijing, China, pages 1507–1514.

Don't understand a measure? Learn it: Structured Prediction for Coreference Resolution optimizing its measures

Iryna Haponchyk* and **Alessandro Moschitti**
*DISI, University of Trento, 38123 Povo (TN), Italy
Qatar Computing Research Institute, HBKU, 34110, Doha, Qatar
{gaponchik.irina,amoschitti}@gmail.com

Abstract

An assential aspect of structured prediction is the evaluation of an output structure against the gold standard. Especially in the loss-augmented setting, the need of finding the max-violating constraint has severely limited the expressivity of effective loss functions. In this paper, we trade off exact computation for enabling the use of more complex loss functions for coreference resolution (CR). Most noteworthily, we show that such functions can be (i) automatically learned also from controversial but commonly accepted CR measures, e.g., MELA, and (ii) successfully used in learning algorithms. The accurate model comparison on the standard CoNLL–2012 setting shows the benefit of more expressive loss for Arabic and English data.

1 Introduction

In recent years, interesting structured prediction methods have been developed for coreference resolution (CR), e.g., (Fernandes et al., 2014; Björkelund and Kuhn, 2014; Martschat and Strube, 2015). These models are supposed to output clusters but, to better control the exponential nature of the problem, the clusters are converted into tree structures. Although this simplifies the problem, optimal solutions are associated with an exponential set of trees, requiring to maximize over such a set. This originated latent models (Yu and Joachims, 2009) optimizing the so-called loss-augmented objective functions.

In this setting, loss functions need to be factorizable together with the feature representations for finding the max-violating constraints. The consequence is that only simple loss functions, basically just counting incorrect edges, were applied in previous work, giving up expressivity for simplicity. This is a critical limitation as domain experts consider more information than just counting edges.

In this paper, we study the use of more expressive loss functions in the structured prediction framework for CR, although some findings are clearly applicable to more general settings. We attempted to optimize the complicated official MELA measure[1] (Pradhan et al., 2012) of CR within the learning algorithm. Unfortunately, MELA is the average of measures, among which CEAF_e has an excessive computational complexity preventing its direct use. To solve this problem, we defined a model for learning MELA from data using a fast linear regressor, which can be then effectively used in structured prediction algorithms. We defined features to learn such a loss function, e.g., different link counts or aggregations such as Precision and Recall. Moreover, we designed methods for generating training data from which our regression loss algorithm (RL) can generalize well and accurately predict MELA values on unseen data.

Since RL is not factorizable[2] over a mention graph, we designed a latent structured perceptron (LSP) that can optimize non-factorizable loss functions on CR graphs. We tested LSP using RL and other traditional loss functions using the same setting of the CoNLL–2012 Shared Task, thus enabling an exact comparison with previous work. The results confirmed that RL can be effectively learned and used in LSP, although the improvement was smaller than expected, considering that our RL provides the algorithm with a more accurate feedback.

Thus, we analyzed the theory behind this pro-

[1] Received most consensus in the NLP community.
[2] We have not found yet a possible factorization.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1018–1028
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1094

cess by also contributing to the definition of the properties of loss optimality. These show that the available loss functions, e.g., by Fernandes et al.; Yu and Joachims, are enough for optimizing MELA on the training set, at least when the data is separable. Thus, in such conditions, we cannot expect a very large improvement from RL.

To confirm such a conjecture, we tested the models in a more difficult setting, in terms of separability. We used different feature sets of a smaller size and found out that in such conditions, RL requires less epochs for converging and produces better results than the other simpler loss functions. The accuracy of RL-based model, using 16 times less features, decreases by just 0.3 points, still improving the state of the art in structured prediction. Accordingly, in the Arabic setting, where the available features are less discriminative, our approach highly improves the standard LSP.

2 Related Work

There is a number of works attempting to directly optimize coreference metrics. The solution proposed by Zhao and Ng (2010) consists in finding an optimal weighting (by beam search) of training instances, which would maximize the target coreference metric. Their models, optimizing MUC and B^3, deliver a significant improvement on the MUC and ACE corpora. Uryupina et al. (2011) benefited from applying genetic algorithms for the selection of features and architecture configuration by multi-objective optimization of MUC and the two CEAF variants. Our approach is different in that the evaluation measure (its approximation) is injected directly into the learning algorithm. Clark and Manning (2016) optimize B^3 directly as well within a mention-ranking model. For the efficiency reasons, they omit optimization of CEAF, which we enable in this work.

$SVM^{cluster}$ – a structured output approach by Finley and Joachims (2005) – enables optimization to any clustering loss function (including non-decomposable ones). The authors experimentally show that optimizing particular loss functions results into a better classification accuracy in terms of the same functions. However, these are in general fast to compute, which is not the MELA case.

While Finley and Joachims are compelled to perform approximate inference to overcome the intractability of finding an optimal clustering, the latent variable structural approaches – SVM of Yu and Joachims (2009) and perceptron of Fernandes et al. (2014) – render exact inference possible by introducing auxiliary graph structures. The modeling of Fernandes et al. (also referred to as the *antecedent tree* approach) is exploited in the works of Björkelund and Kuhn (2014), Martschat and Strube (2015), and Lassalle and Denis (2015). Like us, the first couples such approach with approximate inference but for enabling the use of non-local features. The current state-of-the-art model of Wiseman et al. (2016) also employs a greedy inference procedure as it has global features from an RNN as a non-decomposable term in the inference objective.

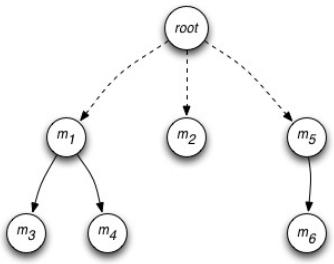


Figure 1: Latent tree used for structural learning

3 Structure Output Learning for CR

We consider online learning algorithms for linking structured input and output patterns. More formally, such algorithms find a linear mapping $f(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \Phi(\mathbf{x}, \mathbf{y}) \rangle$, where $f : X \times Y \rightarrow \mathbb{R}$, $\mathbf{w}$ is a linear model, $\Phi(\mathbf{x}, \mathbf{y})$ is a combined feature vector of input variables X and output variables Y. The predicted structure is derived with the $\operatorname{argmax}_{\mathbf{y} \in Y} f(\mathbf{x}, \mathbf{y})$. In the next sections, we show how to learn $\mathbf{w}$ for CR using structured perceptron. Additionally, we provide a characterization of effective loss functions for separable cases.

3.1 Modeling CR

In this framework, CR is essentially modeled as a clustering problem, where an input-output example is described by a tuple $(\mathbf{x}, \mathbf{y}, \mathbf{h})$, $\mathbf{x}$ is a set of entity mentions contained in a text document, $\mathbf{y}$ is set of the corresponding mention clusters, and $\mathbf{h}$ is a latent variable, i.e., an auxiliary structure that can represent the clusters of $\mathbf{y}$. For example, given the following text:

Although *(she)*$_{m_1}$ was supported by *(President Obama)*$_{m_2}$, *(Mrs. Clinton)*$_{m_3}$ missed *(her)*$_{m_4}$ *(chance)*$_{m_5}$, *(which)*$_{m_6}$ *looked* very good before counting votes.

the clusters of the entity mentions are represented by the latent tree in Figure 1, where its nodes are

Algorithm 1 Latent Structured Perceptron

1: Input: $X = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, $\mathbf{w}_0$, C, T
2: $\mathbf{w} \leftarrow \mathbf{w}_0$; $t \leftarrow 0$
3: **repeat**
4: **for** $i = 1, ..., n$ **do**
5: $\mathbf{h}_i^* \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i, \mathbf{y}_i)}{\text{argmax}} \langle \mathbf{w}_t, \Phi(\mathbf{x}_i, \mathbf{h}) \rangle$
6: $\hat{\mathbf{h}}_i \leftarrow \underset{\mathbf{h} \in H(\mathbf{x}_i)}{\text{argmax}} \langle \mathbf{w}_t, \Phi(\mathbf{x}_i, \mathbf{h}) \rangle + C \times \Delta(\mathbf{y}_i, \mathbf{h}_i^*, \mathbf{h})$
7: **if** $\Delta(\mathbf{y}_i, \mathbf{h}^*{}_i, \hat{\mathbf{h}}_i) > 0$ **then**
8: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Phi(\mathbf{x}_i, \mathbf{h}_i^*) - \Phi(\mathbf{x}_i, \hat{\mathbf{h}}_i)$
9: **end if**
10: **end for**
11: $t \leftarrow t + 1$
12: **until** $t < nT$
13: $\mathbf{w} \leftarrow \frac{1}{t} \sum_{i=1}^{t} \mathbf{w}_i$
return $\mathbf{w}$

mentions and the subtrees connected to the additional root node form distinct clusters. The tree $\mathbf{h}$ is called a latent variable as it is consistent with $\mathbf{y}$, i.e., it contains only links between mention nodes that corefer or fall into the same cluster according to $\mathbf{y}$. Clearly, an exponential set of trees, H, can be associated with one and the same clustering $\mathbf{y}$. Using only one tree to represent a clustering makes the search for optimal mention clusters tractable. In particular, structured prediction algorithms select $\mathbf{h}$ that maximizes the model learned at time t as shown in the next section.

3.2 Latent Structured Perceptron (LSP)

The LSP model proposed by Sun et al. (2009) and specialized for solving CR tasks by Fernandes et al. (2012) is described by Alg. 1.

Given a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, initial $\mathbf{w}_0$[3], a trade off parameter C, and the maximum number of epochs T, LSP iterates the following operations: Line 5 finds a latent tree $\mathbf{h}_i^*$ that maximizes $\langle \mathbf{w}_t, \Phi(\mathbf{x}_i, \mathbf{h}) \rangle$ for the current example $(\mathbf{x}_i, \mathbf{y}_i)$. It basically finds the max ground truth tree with respect to the current $\mathbf{w}_t$. Finding such max requires an exploration over the tree set $H(\mathbf{x}_i, \mathbf{y}_i)$, which only contains arcs between mentions that corefer according to the gold standard clustering $\mathbf{y}_i$. Line 6 seeks for the max-violating tree $\hat{\mathbf{h}}_i$ in $H(\mathbf{x}_i)$, which is the set of all candidate trees using any possible combination of arcs. Line 7 tests if the produced tree $\hat{\mathbf{h}}_i$ has some mistakes with respect to the gold clustering $\mathbf{y}_i$, using loss function $\Delta(\mathbf{y}_i, \mathbf{h}_i^*, \hat{\mathbf{h}}_i)$. Note that some models define a loss exploiting also the current best latent tree $\mathbf{h}_i^*$. If the test is verified, the model is updated with the vector $\Phi(\mathbf{x}_i, \mathbf{h}_i^*) - \Phi(\mathbf{x}_i, \hat{\mathbf{h}}_i)$.

[3]Either 0 or a random vector.

Fernandes et al. (2012) used exactly the directed trees we showed as latent structures and applied Edmonds' spanning tree algorithm (Edmonds, 1967) for finding the max. Their model achieved the best results in the CoNLL–2012 Shared Task, a challenge for CR systems (Pradhan et al., 2012). Their selected loss function also plays an important role as shown in the following.

3.3 Loss functions

When defining a loss, it is very important to preserve the factorization of the model components along the latent tree edges since this leads to efficient maximization algorithms (see Section 5).

Fernandes et al. uses a loss function that (i) compares a predicted tree $\hat{\mathbf{h}}$ against the gold tree $\mathbf{h}^*$ and (ii) factorizes over the edges in the way the model does. Its equation is:

$$\Delta_F(\mathbf{h}^*, \hat{\mathbf{h}}) = \sum_{i=1}^{M} \mathbb{1}_{\hat{\mathbf{h}}(i) \neq \mathbf{h}^*(i)} (1 + r \cdot \mathbb{1}_{\mathbf{h}^*(i)=0}), \quad (1)$$

where $\mathbf{h}^*(i)$ and $\hat{\mathbf{h}}(i)$ output the parent of the mention node i in the gold and predicted tree, respectively, whereas $\mathbb{1}_{\mathbf{h}^*(i) \neq \hat{\mathbf{h}}(i)}$ just checks if the parents are different, and if yes, penalty of 1 (or $1 + r$ if the gold parent is the root) is added.

Yu and Joachims's loss is based on undirected tree without a root and on the gold clustering $\mathbf{y}$. It is computed as:

$$\Delta_{YJ}(\mathbf{y}, \hat{\mathbf{h}}) = n(\mathbf{y}) - k(\mathbf{y}) + \sum_{\mathbf{e} \in \hat{\mathbf{h}}} l(\mathbf{y}, \mathbf{e}), \quad (2)$$

where $n(\mathbf{y})$ is the number of graph nodes, $k(\mathbf{y})$ is the number of clusters in $\mathbf{y}$, and $l(\mathbf{y}, \mathbf{e})$ assigns -1 to any edge $\mathbf{e}$ that connects nodes from the same cluster in $\mathbf{y}$, and r otherwise.

In our experiments, we adopt both loss functions, however, in contrast to Fernandes et al., we always measure Δ_F against the gold label $\mathbf{y}$ and not against the current $\mathbf{h}^*$, i.e., in the way it is done by Martschat and Strube (2015), who employ an equivalent LSP model in their work.

3.4 On optimality of simple loss functions

The above loss functions are rather simple and mainly based on counting the number of mistaken edges. Below, we show that such simple loss functions achieve training data separation (if it exists) of a general task measure reaching its max on their 0 mistakes. The latter is a desirable characteristic of many measures used in CR and NLP research.

Proposition 1 (Sufficient condition for optimality of loss functions for learning graphs)**.** *Let* $\Delta(\mathbf{y}, \mathbf{h}^*, \hat{\mathbf{h}}) \geq 0$ *be a simple, edge-factorizable loss function, which is also monotone in the number of edge errors, and let* $\mu(\mathbf{y}, \hat{\mathbf{h}})$ *be any graph-based measure maximized by no edge errors. Then, if the training set is linearly separable LSP optimizing* Δ *converges to the* μ *optimum.*

Proof. If the data is linearly separable the perceptron converges $\Rightarrow \Delta(\mathbf{y}_i, \mathbf{h^*}_i, \hat{\mathbf{h}}_i) = 0, \forall \mathbf{x}_i$. The loss is factorizable, i.e.,

$$\Delta(\mathbf{y}_i, \mathbf{h^*}_i, \hat{\mathbf{h}}_i) = \sum_{\mathbf{e} \in \hat{\mathbf{h}}_i} l(\mathbf{y}_i, \mathbf{h^*}_i, \mathbf{e}), \qquad (3)$$

where $l(\cdot)$ is an edge loss function. Thus, $\sum_{\mathbf{e} \in \hat{\mathbf{h}}_i} l(\mathbf{y}_i, \mathbf{h^*}_i, \mathbf{e}) = 0$. The latter equation and monotonicity imply $l(\mathbf{y}_i, \mathbf{h^*}_i, \mathbf{e}) = 0, \forall \mathbf{e} \in \hat{\mathbf{h}}_i$, i.e., there are no edge mistakes, otherwise by fixing such edges, we would have a smaller Δ, i.e., negative, contradicting the initial positiveness hypothesis. Thus, no edge mistake in any $\mathbf{x}_i$ implies that $\mu(\mathbf{y}, \hat{\mathbf{h}})$ is maximized on the training set. □

Corollary 1. $\Delta_F(\mathbf{h}^*, \hat{\mathbf{h}})$ and $\Delta_{YJ}(\mathbf{y}, \hat{\mathbf{h}})$ are both optimal loss functions for graphs.

Proof. Equations 1 and 2 show that both are 0 when applied to a clustering with no mistake on the edges. Additionally, for each edge mistake more, both loss functions increase, implying monotonicity. Thus, they satisfy all the assumptions of Proposition 1. □

The above characteristic suggests that Δ_F and Δ_{YJ} can optimize any measure that reasonably targets no mistakes as its best outcome. Clearly, this property does not guarantee loss functions to be suitable for a given task measure, e.g., the latter may have different max points and behave rather discontinuously. However, a common practice in NLP is to optimize the maximum of a measure, e.g., in case of Precision and Recall, or Accuracy, therefore, loss functions able to at least achieve such an optimum are preferable.

4 Automatically learning a loss function

How to measure a complex task such as CR has generated a long and controversial discussion in the research community. While such a debate is progressing, the most accepted and used measure is the so-called Mention, Entity, and Link Average (MELA) score. As it will be clear from the description below, MELA is not easily interpretable and not robust to the mention identification effect (Moosavi and Strube, 2016). Thus, loss functions showing the optimality property may not be enough to optimize it. Our proposal is to use a version of MELA transformed in a loss function optimized by an LSP algorithm with inexact inference. However, the computational complexity of the measure prevents to carry out an effective learning. Our solution is thus to learn MELA with a fast linear regressor, which also produces a continuos version of the measure.

4.1 Measures for CR

MELA is the unweighted average of MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998) and $CEAF_e$ (CEAF variant with entity-based similarity) (Luo, 2005; Cai and Strube, 2010) scores, having heterogeneous nature.

MUC is based on the number of correctly predicted links between mentions. The number of links required for obtaining the key entity set K is $\sum_{k_i \in K}(|k_i| - 1)$, where k_i are key entities in K (cardinality of each entity minus one). MUC recall computes what fraction of these were predicted, and the predicted were as many as $\sum_{k_i \in K}(|k_i| - |p(k_i)|) = \sum_{k_i \in K}(|k_i| - 1 - (|p(k_i)| - 1))$, where $p(k_i)$ is a partition of the key entity k_i formed by intersecting it with the corresponding response entities $r_j \in R$, s.t., $k_i \cap r_j \neq \emptyset$. This number equals to the number of the key links minus the number of missing links, required to unite the parts of the partition $p(k_i)$ to obtain k_i.

B^3 computes Precision and Recall individually for each mention. For mention m: $Recall_m = \frac{|k_i^m \cap r_j^m|}{|k_i^m|}$, where k_i^m and r_j^m, subscripted with m, denote, correspondingly, the key and response entities into which m falls. The over-document Recall is then an average of these taken with respect to the number of the key mentions. The MUC and B^3 Precision is computed by interchanging the roles of the key and response entities.

$CEAF_e$ computes similarity between key and system entities after finding an optimal alignment between them. Using $\psi(k_i, r_j) = \frac{2|k_i \cap r_j|}{|k_i| + |r_j|}$ as the entity similarity measure, it finds an optimal one-to-one map $g^* : K \rightarrow R$, which maps every key entity to a response entity, maximazing an overall similarity $\Psi(g) = \sum_{k_i \in K} \psi(k_i, g(k_i))$ of the example. This is solved as a bipartite matching problem by the Kuhn-Munkres algorithm. Then Preci-

Algorithm 2 Finding a Max-violating Spanning Tree

1: Input: training example $(\mathbf{x}, \mathbf{y})$; graph $G(\mathbf{x})$ with vertices V denoting mentions; set of the incoming candidate edges, $E(\mathbf{v})$, $\mathbf{v} \in V$; weight vector $\mathbf{w}$
2: $\mathbf{h}^* \leftarrow \emptyset$
3: **for** $\mathbf{v} \in V$ **do**
4: $\quad \mathbf{e}^* = \underset{\mathbf{e} \in E(\mathbf{v})}{\operatorname{argmax}} \langle \mathbf{w}, \mathbf{e} \rangle + C \times l(\mathbf{y}, \mathbf{e})$
5: $\quad \mathbf{h}^* = \mathbf{h}^* \cup \mathbf{e}^*$
6: **end for**
7: **return** max-violating tree $\mathbf{h}^*$
8: (clustering $\mathbf{y}^*$ is induced by the tree $\mathbf{h}^*$)

sion and Recall are $\frac{\Psi(g^*)}{\sum_{r_j \in R} \psi(r_j, r_j)}$ and $\frac{\Psi(g^*)}{\sum_{k_i \in K} \psi(k_i, k_i)}$, respectively.

MELA computation is rather expensive mostly because of CEAF$_e$. Its complexity is bounded by $\mathcal{O}(Ml^2 \log l)$ (Luo, 2005), where M and l are, correspondingly, the maximum and minimum number of entities in $\mathbf{y}$ and $\hat{\mathbf{y}}$. Computing CEAF$_e$ is especially slow for the candidate outputs $\hat{\mathbf{y}}$ with a low quality of prediction, i.e, when l is big, and the coherence with the gold $\mathbf{y}$ is scarse.

Finally, B^3 and CEAF$_e$ are strongly influenced by the mention identification effect (Moosavi and Strube, 2016). Thus, Δ_F and Δ_{YJ} may output identical values for different clusterings that can have a big gap in terms of MELA.

4.2 Features for learning measures

As computational reasons prevent to use MELA in LSP (see our inexact search algorithm in Section 5), we study methods for approximating it with a linear regressor. For this purpose, we define nine features, which count either exact or simplified versions of Precision, Recall and F1 of each of the three metric-components of MELA. Clearly, neither Δ_F nor Δ_{YJ} provide the same values.

Apart from the computational complexity, the difficulty of evaluating the quality of the predicted clustering $\hat{\mathbf{y}}$ during training is also due to the fact that CR is carried out on automatically detected mentions, while it needs to be compared against a gold standard clustering of a gold mention set. However, we can use simple information about automatic mentions and how they relate to gold mentions and gold clusters. In particular, we use four numbers: (i) correctly detected automatic mentions, (ii) links they have in the gold standard, (iii) gold mentions, and (iv) gold links. The last one enables the precise computation of Precision, Recall and F1-measure values of MUC; the required partitions $p(k_i)$ of key entities are also available at training time as they contain only automatic mentions. These are the first three features that we design. Likewise for B^3, the feature values can be derived using (ii) and (iii).

For computing CEAF$_e$ heuristics, we do not perform cluster alignment to find an optimal $\Psi(g^*)$. Instead of $\Psi(g^*)$, which can be rewritten as $\sum_{m \in K \cap R} \frac{2}{|k_i^m| + |g^*(k_i^m)|}$ if summing up over the mentions not the entities, we simply use $\tilde{\Psi} = \sum_{m \in K \cap R} \frac{2}{|k_i^m| + |r_j^m|}$, pretending that for each m its key k_i^m and response r_j^m entities are aligned. $\sum_{r_j \in R} \psi(r_j, r_j)$ and $\sum_{k_i \in K} \psi(k_i, k_i)$ in the denominators of the Precision and Recall are the number of predicted and gold clusters, correspondingly. The imprecision of the CEAF$_e$ related features is expected to be leveraged when put together with the exact B^3 and MUC values into the regression learning using the exact MELA values (implicitly exact CEAF$_e$ values as well).

4.3 Generating training and test data

The features described above can be used to characterize the clustering variables $\hat{\mathbf{y}}$. For generating training data, we collected all the max-violating $\hat{\mathbf{y}}$ produced during LSP$_F$ (using Δ_F) learning and associate them with their correct MELA scores from the scorer. This way, we can have both training and test data for our regressor. In our experiments, for the generation purpose, we decided to run LSP$_F$ on each document separately to obtain more variability in $\hat{\mathbf{y}}$'s. We use a simple linear SVM to learn a model $\mathbf{w}_\rho$. Considering that MELA$(\mathbf{y}, \hat{\mathbf{y}})$ score lies in the interval $[100, 0]$, a simple approximation of the loss could be:

$$\Delta_\rho(\mathbf{y}, \hat{\mathbf{y}}) = 100 - \mathbf{w}_\rho \cdot \phi(\mathbf{y}, \hat{\mathbf{y}}). \quad (4)$$

Below, we show its improved version and an LSP for learning with it based on inexact search.

5 Learning with learned loss functions

Our experiments will demonstrate that Δ_ρ can be accurately learned from data. However, the features we used for this are not factorizable over the edges of the latent trees. Thus, we design a new LSP algorithm that can use our learned loss in an approximated max search.

5.1 A general inexact algorithm for CR

If the loss function can be factorized over tree edges (see Equation 3) the max-violating constraint in Line 6 of Alg. 1 can be efficiently found by exact decoding, e.g., using Edmonds' algorithm as in Fernandes et al. (2014) or Kruskal's as

Algorithm 3 Inexact Inference of a Max-violating Spanning Tree with a Global Loss

1: Input: training example $(\mathbf{x}, \mathbf{y})$; graph $G(\mathbf{x})$ with vertices V denoting mentions; set of the incoming candidate edges, $E(\mathbf{v})$, $\mathbf{v} \in V$; $\mathbf{w}$, ground truth tree $\mathbf{h}^*$
2: $\hat{\mathbf{h}} \leftarrow \emptyset$
3: $score \leftarrow 0$
4: **repeat**
5: $\quad prev\_score = score$
6: $\quad score = 0$
7: $\quad$ **for** $\mathbf{v} \in V$ **do**
8: $\quad\quad \mathbf{h} = \hat{\mathbf{h}} \setminus \mathbf{e}(\mathbf{v})$
9: $\quad\quad \hat{\mathbf{e}} = \underset{\mathbf{e} \in E(\mathbf{v})}{\text{argmax}} \langle \mathbf{w}, \mathbf{e} \rangle + C \times \Delta(\mathbf{y}, \mathbf{h}^*, \mathbf{h} \cup \mathbf{e})$
10: $\quad\quad \hat{\mathbf{h}} = \mathbf{h} \cup \hat{\mathbf{e}}$
11: $\quad\quad score = score + \langle \mathbf{w}, \hat{\mathbf{e}} \rangle$
12: $\quad$ **end for**
13: $\quad score = score + \Delta(\mathbf{y}, \mathbf{h}^*, \hat{\mathbf{h}})$
14: **until** $score = prev\_score$
15: **return** max-violating tree $\hat{\mathbf{h}}$

in Yu and Joachims (2009). The candidate graph, by construction, does not contain cycles, and the inference by Edmonds' algorithm does technically the same as the "best-left-link" inference algorithm by Chang et al. (2012). This can be schematically represented in Alg. 2.

When we deal with Δ_ρ, Alg. 2 cannot be longer applied as our new loss function is non-factorizable. Thus, we designed a greedy solution, Alg. 3, which still uses the spanning tree algorithm, though, it is not guaranteed to deliver the max-violating constraint. However, finding even a suboptimal solution optimizing a more accurate loss function may achieve better performance both in terms of speed and accuracy.

We reformulate Step 4 of Alg. 2, where a max-violating incoming edge $\hat{\mathbf{e}}$ is identified for a vertex $\mathbf{v}$. The new max-violating inference objective contains now a global loss measured on the partial structure $\hat{\mathbf{h}}$ built up to now plus a candidate edge $\mathbf{e}$ for a vertex $\mathbf{v}$ in consideration (Line 10 of Alg. 3). On a high level, this resembles the inference procedure of Wiseman et al. (2016), who use it for optimizing global features coming from an RNN. Differently though, after processing all the vertices, we repeat the procedure until the score of $\hat{\mathbf{h}}$ no longer improves.

Note that Björkelund and Kuhn (2014) perform inexact search on the same latent tree structures to extend the model to non-local features. In contrast to our approach, they use beam search and accumulate the early updates.

In addition to the design of an algorithm enabling the use of our Δ_ρ, there are other intricacies caused by the lack of factorization that need to be taken into account (see the next section).

Samples		# examples	MSE	SCC
Train	Test			
S_1	S_2	$6,011$	2.650	99.68
S_2	S_1	$5,496$	2.483	99.70

Table 1: Accuracy of the loss regressor on two different sets of examples generated from different documents samples.

5.2 Approaching factorization properties

The Δ_ρ defined by Equation 4 approximately falls into the interval $[0, 100]$. However, the simple optimal loss functions, Δ_F and Δ_{YJ}, output a value dependent on the size of the input training document in terms of edges (as they factorize in terms of edges). Since this property cannot be learned from MELA by our regression algorithm, we calibrate our loss with respect to the number of correctly predicted mentions, c, in that document, obtaining $\Delta'_\rho = \frac{c}{100}\Delta_\rho$.

Finally, another important issue is connected to the fact that on the way as we incrementally construct a max-violating tree according to Alg. 3, Δ_ρ decreases (and MELA grows), as we add more mentions to the output, traversing the tree nodes $\mathbf{v}$. Thus, to equalize the contribution of the loss among the candidate edges of different nodes, we also scale the loss of the candidate edges of the node $\mathbf{v}$ having order i in the document, according to the formula $\Delta''_\rho = \frac{i}{|V|}\Delta'_\rho$. This can be interpreted as giving more weight to the hard-to-classify instances – an important issue alleviated by Zhao and Ng (2010). Towards the end of the document, the probability of correctly predicting an incoming edge for a node generally decreases, as increases the number of hypotheses.

6 Experiments

In our experiments, we first show that our regressor for learning MELA approximates it rather accurately. Then, we examine the impact of our Δ_ρ on state-of-the-art systems in comparison with other loss functions. Finally, we show that the impact of our model is amplified when learning in smaller feature spaces.

6.1 Setup

Data We conducted our experiments on English and Arabic parts of the corpus from CoNLL 2012-Shared Task[4]. The English data contains 2,802, 343, and 348 documents in the training,

[4] conll.cemantix.org/2012/data.html

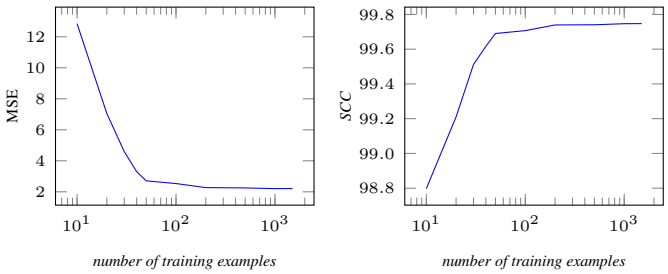


Figure 2: Regressor Learning curves.

dev. and test parts, respectively. The Arabic data includes 359, 44, and 44 documents for training, dev. and test sets, respectively.

Models We implement our version of LSP, where LSP_F, LSP_{YJ}, and LSP_ρ use the loss functions, Δ_F, Δ_{YJ}, and Δ_ρ, defined in Section 3.3 and 5.2, respectively. We used cort[5] – coreference toolkit by Martschat and Strube (2015) both to preprocess the English data and to extract candidate mentions and features (the basic set). For Arabic, we used mentions and features from BART[6] (Uryupina et al., 2012). We extended the initial feature set for Arabic with the feature combinations proposed by Durrett and Klein (2013), those permitted by the available initial features.

Parametrization All the perceptron models require tuning of a regularization parameter C. LSP_F and LSP_{YJ} – also tuning of a specific loss parameter r. We select the parameters on the entire dev. set by training on 100 random documents from the training set. We pick up $C \in \{1.0, 100.0, 1000.0, 2000.0\}$, the r values for LSP_F from the interval $[0.5, 2.5]$ with step 0.5, and the r values for LSP_{YJ} – from $\{0.05, 0.1, 0.5\}$. Ultimately, for English, we used $C = 1000.0$ in all the models; $r = 1.0$ in LSP_F and $r = 0.1$ in LSP_{YJ}. And wider ranges of parameter values were considered for Arabic, due to the lower mention detection rate: $C = 1000.0$, $r = 6.0$ for LSP_F, $C = 1000.0$, $r = 0.01$ for LSP_{YJ}, and $C = 5000.0$ – for LSP_ρ. A standard previous work setting for the number of epochs T of LSP is 5 (Martschat and Strube, 2015). Fernandes et al. (2014) noted that $T = 50$ was sufficient for convergence. We selected the best T from 1 to 50 on the dev. set.

Evaluation measure We used MUC, B^3, CEAF_e and their average MELA for evaluation, computed by the version 8 of the official CoNLL scorer.

[5] http://smartschat.de/software

[6] http://www.bart-coref.org/

Model	**Selected** ($N = 1M$)			**All** ($N \sim 16.8M$)		
	Dev.	Test	T_{best}	Dev.	Test	T_{best}
LSP_F	63.72	62.19	49	64.05	63.05	41
LSP_{YJ}	63.72	62.44	29	64.32	62.76	13
LSP_ρ	64.12	63.09	27	64.30	63.37	18
M&S AT	–	–	–	62.31	61.24	5
M&S MR	–	–	–	63.52	62.47	5
B&K	–	–	–	62.52	61.63	–
Fer	–	–	–	60.57	60.65	–

Table 2: Results of our and previous work models evaluated on the dev. and test sets following the exact CoNLL-2012 English setting, using all training documents with All and $1M$ features. T_{best} is evaluated on the dev. set.

6.2 Learning loss functions

For learning MELA, we generated training and test examples from LSP_F according to the procedure described in Section 4.3. In the first experiment, we trained the $\mathbf{w}_\rho$ model on a set of examples S_1, generated from a sample of 100 English documents and tested on a set of examples S_2, generated from another sample of the same size, and vice versa. The results in Table 1 show that with just $5,000/6,000$, the Mean Squared Error (MSE) is roughly between $\sim 2.4 - 2.7$: these are rather small numbers considering that the regression output values in the interval $[0, 100]$. Squared Correlation Coefficient (SCC) reaches a correlation of about 99.7%, demonstrating that our regression approach is effective in estimating MELA.

Additionally, Figure 2 shows the regression learning curves evaluated with MSE and SCC. The former rapidly decreases and, with about $1,000$ examples, reaches a plateau of around 2.3. The latter shows a similar behaviour, approaching a correlation of about 99.8% with real MELA.

6.3 State of the art and model comparison

We first experimented with the standard CoNLL setting to compare the LSP accuracy in terms of MELA using the three different loss functions, i.e., LSP_F, LSP_{YJ} and LSP_ρ. In particular, we used all the documents of the training set and all $N \sim 16.8$M features from cort, and tested on the both dev. and test sets. The results are reported in Columns All of Table 2.

We note first that our Δ_ρ is effective as it stays on a par with Δ_F and Δ_{YJ} on the dev. set. This is interesting as Corollary 1 shows that such functions can optimize MELA, the reported values refer to the optimal epoch numbers. Also, LSP_ρ improves the other models on the test set by 0.3 percent points (statistical significant at the 93% level of confidence).

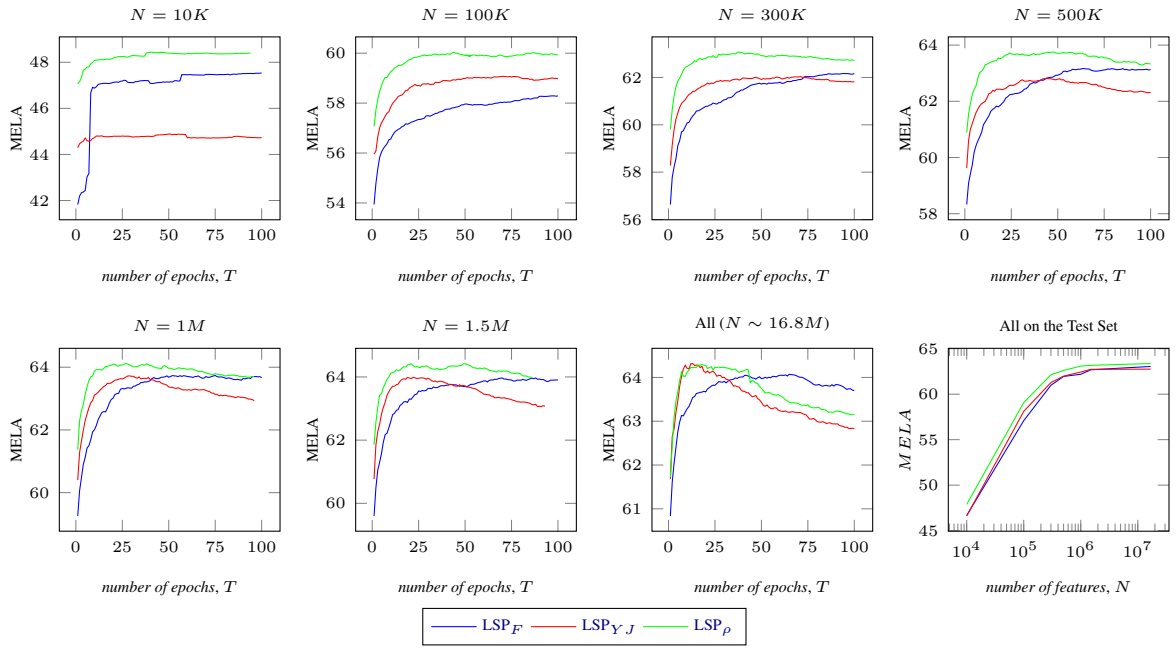


Figure 3: Results of LSP models on the dev. set using different number of features, N. The last plot reports MELA score on the test set of the models using the optimal number of epochs tuned on the dev. set.

#Feat.	Model	Test Set			
		MUC	B^3	$CEAF_e$	$MELA$
All	LSP$_F$	72.66	59.94	56.54	63.05
	LSP$_{YJ}$	72.18	59.31	55.82	62.76
	LSP$_\rho$	72.34	60.36	57.40	63.37
1M	LSP$_F$	71.95	59.03	55.59	62.19
	LSP$_{YJ}$	72.35	59.54	56.38	62.44
	LSP$_\rho$	72.09	60.11	57.07	63.09

Table 3: Results on the test set using the same setting of Table 2 and the measures composing MELA.

Secondly, all the three models improve the state of the art on CR using LSP, i.e., by Martschat and Strube (2015) using antecedent trees (M&S AT) or mention ranking (M&S MR), Björkelund and Kuhn (2014) using a global feature model (B&K) and Fernandes et al. (2014) (Fer). Noted that all the LSP models were trained on the training set only, without retraining on the training and dev. sets together, thus our scores can be improved.

Thirdly, Table 3 shows the breakdown of the MELA results in terms of its components on the test set. Interestingly, LSP$_\rho$ is noticeably better in terms of B^3 and CEAF$_e$, while LSP with simple losses, as expected, deliver higher MUC score.

Finally, the overall improvement of Δ_ρ is not impressive. This mainly depends on the optimality of the competing loss functions, which in a setting of $\sim$ 16.8M features, satisfy the separability condition of Proposition 1.

6.4 Learning in more challenging conditions

In these experiments, we verify the hypothesis that when the optimality property is partially or totally missing Δ_ρ is more visibly superior to Δ_F and Δ_{YJ}. As we do not want to degrade their effectiveness, the only condition dependent on the setting is the data inseparability or at least harder to be separated. These conditions can be obtained by reducing the size of the feature space. However, since we aim at testing conditions, where Δ_ρ is practically useful, we filter out less important features, preserving the model accuracy (at least when the selection is not extremely harsh). For this purpose, we use a feature selection approach using a basic binary classifier trained to discriminate between correct and incorrect mention pairs. It is typically used in non structured CR methods and has a nice property of using the same features of LSP (we do not use global features in our study). We carried out a selection using the absolute values of the model weights of the classifier for ranking features and then selecting those having higher rank (Haponchyk and Moschitti, 2017).

The MELA produced by our models using all the training data is presented in Figure 3. The first 7 plots show learning curves in terms of LSP epochs for different feature sets with increasing size N, evaluated on the dev. set. We note that: firstly, the fewer features are available, the better LSP$_\rho$ curves are than those of LSP$_F$ and LSP$_{YJ}$ in terms of accuracy and convergence speed. The intuition is that finding a separation of the training set (generalizing well) becomes more challenging (e.g., with 10k features, the data is not linearly sep-

arable) thus a loss function which is closer to the real measure provides some advantages.

Secondly, when using all features, LSP$_\rho$ is still overall better than the other models but clearly the latter can achieve the same MELA on the dev. set.

Thirdly, the last plot shows the MELA produced by LSP models on the test set, when trained with the best epoch derived from the dev. set (previous plots). We observe that LSP$_\rho$ is constantly better than the other models, though decreasing its effect as the feature number increases.

Next, in Column 1 (Selected) of Table 2, we report the model MELA using 1 million features. We note that LSP$_\rho$ improves the other models by at least 0.6 percent points, achieving the same accuracy as the best of its competitors, i.e., LSP$_F$, using all the features.

Finally, Δ_ρ does not satisfy Proposition 1, therefore, generally, we do not know if it can optimize any μ-type measure over graphs. However, being learned to optimize MELA, it clearly separates data maximizing such a measure. We empirically verified this by checking the MELA score obtained on the training set: we found that LSP$_\rho$ always optimizes MELA, iterating for fewer epochs than the other loss functions.

6.5 Generalization to other languages

Here, we test the effectiveness of the proposed method on Arabic using all available data and features. The results in Table 4 reveal an indisputable superiority of LSP$_\rho$ over the counterparts optimizing simple loss functions. They support the results of the previous section as we had to deal with the insufficiency of the expert-based features for Arabic. In such an uneasy case, LSP$_\rho$ was able to improve over LSP$_F$ by more than 4.7 points.

We also tested the loss model $\mathbf{w}_\rho$ trained for the experiments on the English data (resp. setting All of Section 6.3) in LSP$_\rho$ on Arabic. This corresponds to LSP$_\rho^{EN}$ model. Notably, it performs even better, 1.5 points more, than LSP$_\rho$ using a loss learned from Arabic examples. This suggests a nice property of data invariance of Δ_ρ. The improvement delivered by the "English" $\mathbf{w}_\rho$ is due to the fact that it was trained on the data which is richer: (i) quantitatively, since coming from almost 8 times more training documents in comparison to Arabic and (ii) qualitatively, in a sense of diversity with respect to the RL target value. Indeed, the Arabic data is much less separable than the English data and this prevents to have examples where MELA values are higher.

Model	**All** ($N \sim 395K$)		
	Dev.	Test	T_{best}
LSP$_F$	31.20	33.19	10
LSP$_{YJ}$	27.70	28.51	13
LSP$_\rho$	36.91	37.91	6
LSP$_\rho^{EN}$	**38.47**	**39.56**	12
Uryupina et al., 2012	–	37.54	–
B&K	46.67	48.72	–
Fer	–	45.18	–

Table 4: Results of our and baseline models evaluated on the dev. and test sets following the exact CoNLL-2012 Arabic setting, using all training documents. T_{best} is evaluated on the dev. set.

7 Conclusions

In this paper, we studied the use of complex loss functions in structured prediction for CR. Given the scale of our investigation, we limited our study to LSP, which is anyway considered state of the art. We derived several findings: (i) for the first time, up to our knowledge, we showed that a complex measure, such as MELA, can be learned by a linear regressor (RL) with high accuracy and effective generalization. (ii) The latter was essential for designing a new LSP based on inexact search and RL. (iii) We showed that an automatically learned loss can be optimized and provides state-of-the-art performance in a real setting, including thousands of documents and millions of features, such as CoNLL–2012 Shared Task. (iv) We defined a property of optimal loss functions for CR, which shows that in separable cases, such losses are enough to get the state of the art. However, as soon as separability becomes more complex simple loss functions lose optimality and RL becomes more accurate and faster. (v) Our MELA approximation provides a loss that is data invariant which, once learned, can be optimized in LSP on different datasets and in different languages.

Our study opens several future directions, ranging from defining algorithms based on automatically learned loss functions to learning more effective measures from expert examples.

Acknowledgements

We would like to thank Olga Uryupina for providing us with the preprocessed data from BART for Arabic. This work has been supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action). Many thanks to the anonymous reviewers for their valuable suggestions.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the Linguistic Coreference Workshop at the First International Conference on Language Resources and Evaluation*. Granada, Spain, pages 563–566.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 47–57. http://www.aclweb.org/anthology/P/P14/P14-1005.

Jie Cai and Michael Strube. 2010. Evaluation metrics for end-to-end coreference resolution systems. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Stroudsburg, PA, USA, SIGDIAL '10, pages 28–36. http://dl.acm.org/citation.cfm?id=1944506.1944511.

Kai-Wei Chang, Rajhans Samdani, Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. Illinois-coref: The ui system in the conll-2012 shared task. In *Joint Conference on EMNLP and CoNLL - Shared Task*. Association for Computational Linguistics, Jeju Island, Korea, pages 113–117. http://www.aclweb.org/anthology/W12-4513.

Kevin Clark and Christopher D. Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 643–653. http://www.aclweb.org/anthology/P16-1061.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Jack Edmonds. 1967. Optimum branchings. *Journal of research of National Bureau of standards* pages 233–240.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on EMNLP and CoNLL - Shared Task*. Association for Computational Linguistics, Jeju Island, Korea, pages 41–48. http://www.aclweb.org/anthology/W12-4502.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2014. Latent trees for coreference resolution. *Computational Linguistics* 40(4):801–835.

Thomas Finley and Thorsten Joachims. 2005. Supervised clustering with support vector machines. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*. ACM, New York, NY, USA, pages 217–224. https://doi.org/10.1145/1102351.1102379.

Iryna Haponchyk and Alessandro Moschitti. 2017. A practical perspective on latent structured prediction for coreference resolution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 143–149. http://www.aclweb.org/anthology/E17-2023.

Emmanuel Lassalle and Pascal Denis. 2015. Joint anaphoricity detection and coreference resolution with constrained latent structures. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'15, pages 2274–2280. http://dl.acm.org/citation.cfm?id=2886521.2886637.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '05, pages 25–32. https://doi.org/10.3115/1220575.1220579.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics* 3:405–418.

Nafise Sadat Moosavi and Michael Strube. 2016. Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 632–642. http://www.aclweb.org/anthology/P16-1060.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*. Association for Computational Linguistics, Jeju Island, Korea, page 1–40. http://www.aclweb.org/anthology/W12-4501.

Xu Sun, Takuya Matsuzaki, Daisuke Okanohara, and Jun'ichi Tsujii. 2009. Latent variable perceptron algorithm for structured classification. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'09, pages 1236–1242. http://dl.acm.org/citation.cfm?id=1661445.1661643.

Olga Uryupina, Alessandro Moschitti, and Massimo Poesio. 2012. Bart goes multilingual:

The unitn/essex submission to the conll-2012 shared task. In *Joint Conference on EMNLP and CoNLL - Shared Task*. Association for Computational Linguistics, Stroudsburg, PA, USA, CoNLL '12, pages 122–128. http://dl.acm.org/citation.cfm?id=2391181.2391198.

Olga Uryupina, Sriparna Saha, Asif Ekbal, and Massimo Poesio. 2011. Multi-metric optimization for coreference: The unitn/iitp/essex submission to the 2011 conll shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL Shared Task '11, pages 61–65. http://dl.acm.org/citation.cfm?id=2132936.2132944.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference*. pages 45–52.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 994–1004. http://aclweb.org/anthology/N/N16/N16-1114.pdf.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '09, pages 1169–1176. https://doi.org/10.1145/1553374.1553523.

Shanheng Zhao and Hwee Tou Ng. 2010. Maximum metric score training for coreference resolution. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, Beijing, China, pages 1308–1316. http://www.aclweb.org/anthology/C10-1147.

Bayesian Modeling of Lexical Resources for Low-Resource Settings

Nicholas Andrews and **Mark Dredze** and **Benjamin Van Durme** and **Jason Eisner**
Department of Computer Science and Human Language Technology Center of Excellence
Johns Hopkins University
3400 N. Charles St., Baltimore, MD 21218 USA
{noa,eisner,mdredze,vandurme}@jhu.edu

Abstract

Lexical resources such as dictionaries and gazetteers are often used as auxiliary data for tasks such as part-of-speech induction and named-entity recognition. However, discriminative training with lexical features requires annotated data to reliably estimate the lexical feature weights and may result in overfitting the lexical features at the expense of features which generalize better. In this paper, we investigate a more robust approach: we stipulate that the lexicon is the result of an assumed generative process. Practically, this means that we may treat the lexical resources as *observations* under the proposed generative model. The lexical resources provide training data for the generative model without requiring separate data to estimate lexical feature weights. We evaluate the proposed approach in two settings: part-of-speech induction and low-resource named-entity recognition.

1 Introduction

Dictionaries and gazetteers are useful in many natural language processing tasks. These lexical resources may be derived from freely available sources (such as Wikidata and Wiktionary) or constructed for a particular domain. Lexical resources are typically used to complement existing annotations for a given task (Ando and Zhang, 2005; Collobert et al., 2011). In this paper, we focus instead on low-resource settings where task annotations are unavailable or scarce. Specifically, we use lexical resources to guide part-of-speech induction (§4) and to bootstrap named-entity recognizers in low-resource languages (§5).

Given their success, it is perhaps surprising that incorporating gazetteers or dictionaries into discriminative models (e.g. conditional random fields) may sometimes *hurt performance*. This phenomena is called *weight under-training*, in which lexical features—which detect whether a name is listed in the dictionary or gazetteer—are given excessive weight at the expense of other useful features such as spelling features that would generalize to unlisted names (Smith et al., 2005; Sutton et al., 2006; Smith and Osborne, 2006). Furthermore, discriminative training with lexical features requires sufficient annotated training data, which poses challenges for the unsupervised and low-resource settings we consider here.

Our observation is that Bayesian modeling provides a principled solution. The lexicon is itself a dataset that was generated by some process. Practically, this means that lexicon entries (words or phrases) may be treated as additional observations. As a result, these entries provide information about how names are spelled. The presence of the lexicon therefore now *improves* training of the spelling features, rather than *competing with* the spelling features to help explain the labeled corpus.

A downside is that generative models are typically less feature-rich than their globally normalized discriminative counterparts (e.g. conditional random fields). In designing our approach—the *hierarchical sequence memoizer (HSM)*—we aim to be reasonably expressive while retaining practically useful inference algorithms. We propose a Bayesian nonparametric model to serve as a generative distribution responsible for both lexicon and corpus data. The proposed model *memoizes* previously used lexical entries (words or phrases) but backs off to a character-level distribution when generating novel types (Teh, 2006; Mochihashi et al., 2009). We propose an efficient inference algorithm for the proposed model using particle Gibbs sampling (§3). Our code is available at https://github.com/noa/bayesner.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1029–1039
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1095

2 Model

Our goal is to fit a model that can automatically annotate text. We observe a supervised or unsupervised training corpus. For each label y in the annotation scheme, we also observe a lexicon of strings of type y. For example, in our tagging task (§4), a dictionary provides us with a list of words for each part-of-speech tag y. (These lists need not be disjoint.) For named-entity recognition (NER, §5), we use a list of words or phrases for each named-entity type y (PER, LOC, ORG, etc.).[1]

2.1 Modeling the lexicon

We may treat the lexicon for type y, of size m_y, as having been produced by a set of m_y IID draws from an unknown distribution P_y over the words or named entities of type y. It therefore provides some evidence about P_y. We will later assume that P_y is also used when generating mentions of these words or entities in text. Thanks to this sharing of P_y, if $x = \texttt{Washington}$ is listed in the gazetteer of locations ($y = \text{LOC}$), we can draw the same conclusions as if we had seen a LOC-labeled instance of `Washington` in a supervised corpus.

Generalizing this a bit, we may suppose that one observation of string x in the lexicon is equivalent to c labeled tokens of x in a corpus, where the constant $c > 0$ is known as a *pseudocount*. In other words, observing a lexicon of m_y distinct types $\{x_1, \ldots, x_{m_y}\}$ is equivalent to observing a labeled *pseudocorpus* of cm_y tokens. Notice that given such an observation, the prior probability of any candidate distribution P_y is reweighted by the likelihood $\frac{(cm_y)!}{(c!)^{m_y}} \cdot (P_y(x_1)P_y(x_2)\cdots P_y(x_{m_y}))^c$. Therefore, this choice of P_y can have relatively high posterior probability only to the extent that it assigns high probability to all of the lexicon types.

2.2 Discussion

We employ the above model because it has reasonable qualitative behavior and because computationally, it allows us to condition on observed lexicons as easily as we condition on observed corpora. However, we caution that as a generative model of the lexicon, it is deficient, in the sense that it allocates probability mass to events that cannot actually correspond to any lexicon. After all, drawing cm_y IID tokens from P_y is highly unlikely to result in exactly c tokens of each of m_y different types, and yet a run of our system will always assume that precisely this happened to produce each observed lexicon! To avoid the deficiency, one could assume that the lexicon was generated by rejection sampling: that is, the gazetteer author repeatedly drew samples of size cm_y from P_y until one was obtained that had this property, and then returned the set of distinct types in that sample as the lexicon for y. But this is hardly a realistic description of how gazetteers are actually constructed. Rather, one imagines that the gazetteer author simply harvested a lexicon of frequent types from P_y or from a corpus of tokens generated from P_y. For example, a much better generative story is that the lexicon was constructed as the first m_y distinct types to appear $\geq c$ times in an unbounded sequence of IID draws from P_y. When $c = 1$, this is equivalent to modeling the lexicon as m_y draws *without replacement* from P_y.[2] Unfortunately, draws without replacement are no longer IID or exchangeable: order matters. It would therefore become difficult to condition inference and learning on an observed lexicon, because we would need to explicitly sum or sample over the possibilities for the latent *sequence* of tokens (or stick segments). We therefore adopt the simpler deficient model.

A version of our lexicon model (with $c = 1$) was previously used by Dreyer and Eisner (2011, Appendix C), who observed a list of verb paradigm types rather than word or entity-name types.

2.3 Prior distribution over P_y

We assume *a priori* that P_y was drawn from a Pitman-Yor process (PYP) (Pitman and Yor, 1997). Both the lexicon and the ordinary corpus are observations that provide information about P_y. The PYP is defined by three parameters: a concentration parameter α, a discount parameter d, and a base distribution H_y. In our case, H_y is a distribution over $\mathcal{X} = \Sigma^*$, the set of possible strings over a finite character alphabet Σ.

For example, H_{LOC} is used to choose new place names, so it describes what place names tend to

[1] Dictionaries and knowledge bases provide more information than we use in this paper. For instance, Wikidata also provides a wealth of attributes and other metadata for each entity s. In principle, this additional information could also be helpful in estimating $P_y(s)$; we leave this intriguing possibility for future work.

[2] If we assume that P_y was drawn from a Pitman-Yor process prior (as in §2.3) using the stick-breaking method (Pitman, 1996), it is also equivalent to modeling the lexicon as the set of labels of the *first* m_y stick segments (which tend to have high probability).

look like in the language. The draw $P_{\text{LOC}} \sim \text{PYP}(d, \alpha, H_{\text{LOC}})$ is an "adapted" version of H_{LOC}. It is P_{LOC} that determines how often each name is mentioned in text (and whether it is mentioned in the lexicon). Some names such as `Washington` that are merely plausible under H_{LOC} are far more frequent under P_{LOC}, presumably because they were chosen as the names of actual, significant places. These place names were randomly drawn from H_{LOC} as part of the procedure for drawing P_y.

The expected value of P_y is H (i.e., H is the mean of the PYP distribution), but if α and d are small, then a typical draw of P_y will be rather different from H, with much of the probability mass falling on a subset of the strings.

At training or test time, when deciding whether to label a corpus token of $x = \texttt{Washington}$ as a place or person, we will be interested in the relative values of $P_{\text{LOC}}(x)$ and $P_{\text{PER}}(x)$. In practice, we do not have to represent the unknown infinite object P_y, but can integrate over its possible values. When $P_y \sim \text{PYP}(d, \alpha, H_y)$, then a sequence of draws $X_1, X_2, \ldots \sim P_y$ is distributed according to a Chinese restaurant process, via

$$\begin{aligned} P_y(X_{i+1} = x \mid X_1, \ldots, X_i) &= \frac{\text{customers}(x) - d \cdot \text{tables}(x)}{\alpha + i} \\ &\quad + \frac{\alpha + d \cdot \sum_{x'} \text{tables}(x')}{\alpha + i} H_y(x) \end{aligned} \tag{1}$$

where $\text{customers}(x) \leq i$ is the number of times that x appeared among $X_1, \ldots, X_i$, and $\text{tables}(x) \leq \text{customers}(x)$ is the number of those times that x was drawn from H_y (where each $P_y(X_i \mid \cdots)$ defined by (1) is interpreted as a mixture distribution that *sometimes* uses H_y).

2.4 Form of the base distribution H_y

By fitting H_y on corpus and lexicon data, we learn what place names or noun strings tend to look like in the language. By simultaneously fitting P_y, we learn which ones are commonly mentioned. Recall that under our model, tokens are drawn from P_y but the underlying types are drawn from H_y, e.g., H_y is responsible for (at least) the first token of each type.

A simple choice for H_y is a Markov process that emits characters in $\Sigma \cup \{\$\}$, where $\$$ is a distinguished stop symbol that indicates the end of the string. Thus, the probability of producing $\$$ controls the typical string length under H_y.

We use a more sophisticated model of strings—a sequence memoizer (SM), which is a (hierarchical) Bayesian treatment of variable-order Markov modeling (Wood et al., 2009). The SM allows dependence on an unbounded history, and the probability of a given sequence (string) can be found efficiently much as in equation (1).

Given a string $x = a_1 \cdots a_J \in \Sigma^*$, the SM assigns a probability to it via

$$\begin{aligned} H_y(\boldsymbol{a}_{1:J}) &= \Big(\prod_{j=1}^{J} H_y(a_j \mid \boldsymbol{a}_{1:j-1}) \Big) H_y(\$ \mid \boldsymbol{a}_{1:J}) \\ &= \Big(\prod_{j=1}^{J} H_{y,\boldsymbol{a}_{1:j-1}}(a_j) \Big) H_{y,\boldsymbol{a}_{1:J}}(\$) \end{aligned} \tag{2}$$

where $H_{y,\boldsymbol{u}}(a)$ denotes the conditional probability of character a given the left context $\boldsymbol{u} \in \Sigma^*$. Each $H_{y,\mathbf{u}}$ is a distribution over Σ, defined recursively as

$$\begin{aligned} H_{y,\epsilon} &\sim \text{PYP}(d_\epsilon, \alpha_\epsilon, \mathcal{U}_\Sigma) \\ H_{y,\boldsymbol{u}} &\sim \text{PYP}(d_{|\mathbf{u}|}, \alpha_{|\mathbf{u}|}, H_{y,\sigma(\mathbf{u})}) \end{aligned} \tag{3}$$

where ϵ is the empty sequence, $\mathcal{U}_\Sigma$ is the uniform distribution over $\Sigma \cup \{\$\}$, and $\sigma(\mathbf{u})$ drops the first symbol from $\mathbf{u}$. The discount and concentration parameters $(d_{|\mathbf{u}|}, \alpha_{|\mathbf{u}|})$ are associated with the lengths of the contexts $|\boldsymbol{u}|$, and should generally be larger for longer (more specific) contexts, implying stronger backoff from those contexts.[3]

Our inference procedure is largely indifferent to the form of H_y, so the SM is not the only option. It would be possible to inject more assumptions into H_y, for instance via structured priors for morphology or a grammar of name structure. Another possibility is to use a parametric model such as a neural language model (e.g., Jozefowicz et al. (2016)), although this would require an inner-loop of gradient optimization.

2.5 Modeling the sequence of tags y

We now turn to modeling the corpus. We assume that each sentence is generated via a sequence of latent labels $\boldsymbol{y} = \boldsymbol{y}_{1:T} \in \mathcal{Y}^*$.[4] The observations

[3] We fix these hyperparameters using the values suggested in (Wood et al., 2009; Gasthaus and Teh, 2010), which we find to be quite robust in practice. One could also resample their values (Blunsom and Cohn, 2010); we experimented with this but did not observe any consistent advantage to doing so in our setting.

[4] The label sequence is terminated by a distinguished end-of-sequence label, again written as $\$$.

$\boldsymbol{x}_{1:T}$ are then generated conditioned on the label sequence via the corresponding P_y distribution (defined in §2.3). All observations with the *same* label y are drawn from the same P_y, and thus this *subsequence* of observations is distributed according to the Chinese restaurant process (1).

We model $\boldsymbol{y}$ using another sequence memoizer model. This is similar to other hierarchical Bayesian models of latent sequences (Goldwater and Griffiths, 2007; Blunsom and Cohn, 2010), but again, it does not limit the Markov order (the number of preceding labels that are conditioned on). Thus, the probability of a sequence of latent types is computed in the same way as the base distribution in §2.4, that is,

$$p(\boldsymbol{y}_{1:T}) := \Big(\prod_{t=1}^{T} G_{\boldsymbol{y}_{1:t-1}}(y_t)\Big) G_{\boldsymbol{y}_{1:T}}(\$) \quad (4)$$

where $G_{\boldsymbol{v}}(y)$ denotes the conditional probability of latent label $y \in \mathcal{Y}$ given the left context $\boldsymbol{v} \in \mathcal{Y}^*$. Each $G_{\mathbf{v}}$ is a distribution over $\mathcal{Y}$, defined recursively as

$$G_{\epsilon} \sim \mathrm{PYP}(d_{\epsilon}, \alpha_{\epsilon}, \mathcal{U}_{\mathcal{Y}}) \quad (5)$$
$$G_{\boldsymbol{v}} \sim \mathrm{PYP}(d_{|\mathbf{v}|}, \alpha_{|\mathbf{v}|}, G_{\sigma(\mathbf{v})})$$

The probability of transitioning to label y_t depends on the assignments of all previous labels $y_1 \cdots y_{t-1}$.

For part-of-speech induction, each label y_t is the part-of-speech associated with the corresponding word x_t. For named-entity recognition, we say that each word token is labeled with a named entity type (LOC, PER, ...),[5] or with *itself* if it is not a named entity but rather a "context word." For example, the word token x_t = `Washington` could have been emitted from the label y_t = LOC, or from y_t = PER, or from y_t = `Washington` itself (in which case $p(x_t \mid y_t) = 1$). This uses a much larger set of labels $\mathcal{Y}$ than in the traditional setup where all context words are emitted from the same latent label type `O`. Of course, most labels are impossible at most positions (e.g., y_t cannot be `Washington` *unless* x_t = `Washington`). This scheme makes our generative model sensitive to specific contexts (which is accomplished in discriminative NER systems by contextual features). For example, the SM for $\boldsymbol{y}$ can learn that `spoke to` PER `yesterday` is a common 4-gram in the label sequence $\boldsymbol{y}$, and thus we are more likely to label `Washington` as a person if $\boldsymbol{x}$ = `...spoke to Washington yesterday...`.

We need one change to make this work, since now $\mathcal{Y}$ must include not only the standard NER labels $\mathcal{Y}' = \{\text{PER}, \text{LOC}, \text{ORG}, \text{GPE}\}$ but also words like `Washington`. Indeed, now $\mathcal{Y} = \mathcal{Y}' \cup \Sigma^*$. But no uniform distribution exists over the infinite set Σ^*, so how should we replace the base distribution $\mathcal{U}_{\mathcal{Y}}$ over labels in equation (5)? Answer: To draw from the new base distribution, sample $y \sim \mathcal{U}_{\mathcal{Y}' \cup \{\text{CONTEXT}\}}$. If y = CONTEXT, however, then "expand" it by resampling $y \sim H_{\text{CONTEXT}}$. Here H_{CONTEXT} is the base distribution over spellings of context words, and is learned just like the other H_y distributions in §2.4.

3 Inference via particle Markov chain Monte Carlo

3.1 Sequential sampler

Taking $\boldsymbol{Y}$ to be a random variable, we are interested in the posterior distribution $p(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{x})$ over label sequences $\boldsymbol{y}$ given the emitted word sequence $\boldsymbol{x}$. Our model does not admit an efficient dynamic programming algorithm, owing to the dependencies introduced among the $\boldsymbol{Y}_t$ when we marginalize over the unknown G and P distributions that govern transitions and emissions, respectively. In contrast to tagging with a hidden Markov model tagging, the distribution of each label Y_t depends on *all* previous labels $\boldsymbol{y}_{1:t-1}$, for two reasons: ① The transition distribution $p(Y_t = y \mid \boldsymbol{y}_{1:t-1})$ has unbounded dependence because of the PYP prior (4). ② The emission distribution $p(x_t \mid Y_t = y)$ depends on the emissions observed from any earlier tokens of y, because of the Chinese restaurant process (1). When ② is the only complication, block Metropolis-Hastings samplers have proven effective (Johnson et al., 2007). However, this approach uses dynamic programming to sample from a proposal distribution efficiently, which ① precludes in our case. Instead, we use sequential Monte Carlo (SMC)—sometimes called *particle filtering*—as a proposal distribution. Particle filtering is typically used in online settings, including word segmentation (Borschinger and Johnson, 2011), to make decisions before all of $\boldsymbol{x}$ has been observed. However, we are interested in the inference (or *smoothing*) problem that conditions on all of $\boldsymbol{x}$ (Dubbin and Blunsom, 2012; Tripuraneni et al., 2015).

SMC employs a *proposal distribution* $q(\boldsymbol{y} \mid \boldsymbol{x})$

[5] In §3.2, we will generalize this labeling scheme to allow multi-word named entities such as `New York`.

whose definition decomposes as follows:

$$q(y_1 \mid x_1) \prod_{t=2}^{T} q(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t}) \tag{6}$$

for $T = |\boldsymbol{x}|$. To sample a sequence of latent labels, first sample an initial label y_1 from q_1, then proceed incrementally by sampling y_t from $q_t(\cdot \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t})$ for $t = 2, \ldots, T$. The final sampled sequence $\boldsymbol{y}$ is called a *particle*, and is given an *unnormalized importance weight* of $\tilde{w} = \tilde{w}_T \cdot p(\$ \mid \boldsymbol{y}_{1:T})$ where $\tilde{w}_T$ was built up via

$$\tilde{w}_t := \tilde{w}_{t-1} \cdot \frac{p(\boldsymbol{y}_{1:t}, \boldsymbol{x}_{1:t})}{p(\boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t-1})\, q(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t})} \tag{7}$$

The SMC procedure consists of generating a system of M weighted particles whose unnormalized importance weights $\tilde{w}^{(m)} : 1 \leq m \leq M$ are normalized into $w^{(m)} := \tilde{w}^{(m)} / \sum_{m=1}^{M} \tilde{w}^{(m)}$. As $M \to \infty$, SMC provides a consistent estimate of the marginal likelihood $p(\boldsymbol{x})$ as $\frac{1}{M} \sum_{m=1}^{M} \tilde{w}^{(m)}$, and samples from the weighted particle system are distributed as samples from the desired posterior $p(\boldsymbol{y} \mid \boldsymbol{x})$ (Doucet and Johansen, 2009).

Particle Gibbs. We employ SMC as a kernel in an MCMC sampler (Andrieu et al., 2010). In particular, we use a block Gibbs sampler in which we iteratively resample the hidden labeling $\boldsymbol{y}$ of a sentence $\boldsymbol{x}$ conditioned on the current labelings for all other sentences in the corpus. In this context, the algorithm is called *conditional* SMC since one particle is always fixed to the previous sampler state for the sentence being resampled, which ensures that the MCMC procedure is ergodic. At a high level, this procedure is analogous to other Gibbs samplers (e.g. for topic models), except that the conditional SMC (CSMC) kernel uses auxiliary variables (particles) in order to generate the new block variable assignments. The procedure is outlined in Algorithm 1. Given a previous latent state assignment $\boldsymbol{y}'_{1:T}$ and observations $\boldsymbol{x}_{1:T}$, the CSMC kernel produces a new latent state assignment via M auxiliary particles where one particle is fixed to the previous assignment. For ergodicity, $M \geq 2$, where larger values of M may improve mixing rate at the expense of increased computation per step.

Proposal distribution. The choice of proposal distribution q is crucial to the performance of SMC methods. In the case of continuous latent variables, it is common to propose y_t from the transition probability $p(Y_t \mid \boldsymbol{y}_{1:t-1})$ because this distribution usually has a simple form that permits efficient sampling. However, it is possible to do better in the case of discrete latent variables. The optimal proposal distribution is the one which minimizes the variance of the importance weights, and is given by

$$\begin{aligned} q(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t}) &:= p(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x}_{1:t}) \qquad (8) \\ &= \frac{p(y_t \mid \boldsymbol{y}_{1:t-1}) p(x_t \mid y_t)}{p(x_t \mid \boldsymbol{y}_{1:t-1})} \end{aligned}$$

where

$$p(x_t \mid \boldsymbol{y}_{1:t-1}) = \sum_{y_t \in \mathcal{Y}} p(y_t \mid \boldsymbol{y}_{1:t-1}) p(x_t \mid y_t) \tag{9}$$

Substituting this expression in equation (7) and simplifying yields the incremental weight update:

$$\tilde{w}_t := \tilde{w}_{t-1} \cdot p(x_t \mid \boldsymbol{y}_{1:t-1}) \tag{10}$$

Resampling. In filtering applications, it is common to use resampling operations to prevent weight degeneracy. We do not find resampling necessary here for three reasons. First, note that we resample hidden label sequences that are only as long as the number of words in a given sentence. Second, we use a proposal which minimizes the variance of the weights. Finally, we use SMC as a kernel embedded in an MCMC sampler; asymptotically, this procedure yields samples from the desired posterior regardless of degeneracy (which only affects the mixing rate). Practically speaking, one can diagnose the need for resampling via the effective sample size (ESS) of the particle system:

$$\text{ESS} := \frac{1}{\sum_{m=1}^{M} (\tilde{w}^{(m)})^2} = \frac{(\sum_{m=1}^{M} w^{(m)})^2}{\sum_{m=1}^{M} (w^{(m)})^2}$$

In our experiments, we find that ESS remains high (a significant fraction of M) even for long sentences, suggesting that resampling is not necessary to enable mixing of the the Gibbs sampler.

Decoding. In order to obtain a single latent variable assignment for evaluation purposes, we simply take the state of the Markov chain after a fixed number of iterations of particle Gibbs. In principle, one could collect many samples during particle Gibbs and use them to perform minimum Bayes risk decoding under a given loss function. However, this approach is somewhat slower and did not appear to improve performance in preliminary experiments

Algorithm 1 Conditional SMC

procedure CSMC($\boldsymbol{x}_{1:T}$, $\boldsymbol{y}'_{1:T}$, M)
  Draw $y_1^{(m)}$ (eqn. 8) for $m \in [1, M-1]$
  Set $y_1^{(M)} = y'_1$
  Set $\tilde{w}_1^{(m)}$ (eqn. 10) for $m \in [1, M]$
  for $t = 2$ to T **do**
    Draw $y_t^{(m)}$ (eqn. 8) for $m \in [1, M-1]$
    Set $y_t^M = y'_t$
    Set $\tilde{w}_t^{(m)}$ (eqn. 10) for $m \in [1, M]$
  Set $\tilde{w}^{(m)} = \tilde{w}_T^{(m)} p(\$ | \boldsymbol{y}_{1:T})$ for $m \in [1, M]$
  Draw index k where $p(k = m) \propto \tilde{w}^{(m)}$
  return $\boldsymbol{y}_{1:T}^{(k)}$

3.2 Segmental sampler

We now present an sampler for settings such as NER where each latent label emits a *segment* consisting of 1 or more words. We make use of the same transition distribution $p(y_t \mid \boldsymbol{y}_{1:t-1})$, which determines the probability of a label in a given context, and an emission distribution $p(x_t \mid y_t)$ (namely P_{y_t}); these are assumed to be drawn from hierarchical Pitman-Yor processes described in §2.5 and §2.1, respectively. To allow the x_t to be a multi-word string, we simply augment the character set with a distinguished space symbol $\_ \in \Sigma$ that separates words within a string. For instance, `New` `York` would be generated as the 9-symbol sequence `New_York$`.

Although the *model* emits `New_York` all at once, we still formulate our *inference procedure* as a particle filter that proposes one tag for each word. Thus, for a given segment label type y, we allow two tag types for its words:

- I-y corresponds to a non-final word in a segment of type y (in effect, a word with a following _ attached).
- E-y corresponds to the final word in a segment of type y.

For instance, $\boldsymbol{x}_{1:2} =$ `New York` would be annotated as a location segment by defining $\boldsymbol{y}_{1:2} =$ I-LOC E-LOC. This says that $\boldsymbol{y}_{1:2}$ has *jointly* emitted $\boldsymbol{x}_{1:2}$, an event with prior probability $P_{\text{LOC}}(\texttt{New\_York})$. Each word that is not part of a named entity is considered to be a single-word segment. For example, if the next word were $x_3 =$ `hosted` then it should be tagged with $y_3 =$ `hosted` as in §2.5, in which case x_3 was emitted with probability 1.

To adapt the sampler described in §3.1 for the segmental case, we need only to define the transition and emission probabilities used in equation (8) and its denominator (9).

For the transition probabilities, we want to model the sequence of segment labels. If y_{t-1} is an I- tag, we take $p(y_t \mid \boldsymbol{y}_{1:t-1}) = 1$, since then y_t merely continues an existing segment. Otherwise y_t starts a new segment, and we take $p(y_t \mid \boldsymbol{y}_{1:t-1}) = 1$ to be defined by the PYP's probability $G_{\boldsymbol{y}_{1:t-1}}(y_t)$ as usual, but where we interpret the subscript $\boldsymbol{y}_{1:t-1}$ to refer to the possibly shorter sequence of segment labels implied by those $t-1$ tags.

For the emission probabilities, if y_t has the form I-y or E-y, then its associated emission probability no longer has the form $p(x_t \mid y_t)$, since the choice of x_t also depends on any words emitted earlier in the segment. Let $s \leq t$ be the starting position of the segment that contains t. If $y_t =$ E-y, then the emission probability is proportional to $P_y(x_s \_ x_{s+1} \_ \ldots \_ x_t)$. If $y_t =$ I-y then the emission probability is proportional to the *prefix probability* $\sum_x P_y(x)$ where x ranges over all strings in Σ^* that have $x_s \_ x_{s+1} \_ \ldots \_ x_t \_$ as a proper prefix. Prefix probabilities in H_y are easy to compute because H_y has the form of a language model, and prefix probabilities in P_y are therefore also easy to compute (using a prefix tree for efficiency).

This concludes the description of the segmental sampler. Note that the particle Gibbs procedure is unchanged.

4 Inducing parts-of-speech with type-level supervision

Automatically inducing parts-of-speech from raw text is a challenging problem (Goldwater et al., 2005). Our focus here is on the easier problem of type-supervised part-of-speech induction, in which (partial) dictionaries are used to guide inference (Garrette and Baldridge, 2012; Li et al., 2012). Conditioned on the unlabeled corpus and dictionary, we use the MCMC procedure described in §3.1 to impute the latent parts-of-speech.

Since dictionaries are freely available for hundreds of languages,[6] we see this as a mild additional requirement in practice over the purely unsupervised setting.

In prior work, dictionaries have been used as constraints on possible parts-of-speech: words appearing in the dictionary take one of their known parts-

[6] https://www.wiktionary.org/

of-speech. In our setting, however, the dictionaries are not constraints but evidence. If `monthly` is listed in (only) the adjective lexicon, this tells us that P_{ADJ} sometimes generates `monthly` and therefore that H_{ADJ} may also tend to generate other words that end with `-ly`. However, for us, $P_{\text{ADV}}(\texttt{monthly}) > 0$ as well, allowing us to still correctly treat `monthly` as a possible adverb if we later encounter it in a training or test corpus.

4.1 Experiments

We follow the experimental procedure described in Li et al. (2012), and use their released code and data to compare to their best model: a second-order maximum entropy Markov model parametrized with log-linear features (SHMM-ME). This model uses hand-crafted features designed to distinguish between different parts-of-speech, and it has special handling for rare words. This approach is surprisingly effective and outperforms alternate approaches such as cross-lingual transfer (Das and Petrov, 2011). However, it also has limitations, since words that do not appear in the dictionary will be unconstrained, and spurious or incorrect lexical entries may lead to propagation of errors.

The lexicons are taken from the Wiktionary project; their size and coverage are documented by (Li et al., 2012). We evaluate our model on multi-lingual data released as part of the CoNLL 2007 and CoNLL-X shared tasks. In particular, we use the same set of languages as Li et al. (2012).[7] For our method, we impute the parts-of-speech by running particle Gibbs for 100 epochs, where one epoch consists of resampling the states for a each sentence in the corpus. The final sampler state is then taken as a 1-best tagging of the unlabeled data.

Results. The results are reported in Table 1. We find that our hierarchical sequence memoizer (HSM) matches or exceeds the performance of the baseline (SHMM-ME) for nearly all the tested languages, particularly for morphologically rich languages such as German where the spelling distributions H_y may capture regularities. It is interesting to note that our model performs worse relative to the baseline for English; one possible explanation is that the baseline uses hand-engineered features whereas ours does not, and these features may have been tuned using English data for validation.

[7] With the exception of Dutch. Unlike the other CoNLL languages, Dutch includes phrases, and the procedure by which these were split into tokens was not fully documented.

Our generative model is supposed to exploit lexicons well. To see what is lost from using a generative model, we also compared with Li et al. (2012) on standard supervised tagging without any lexicons. Even here our generative model is very competive, losing only on English and Swedish.

5 Boostrapping NER with type-level supervision

Name lists and dictionaries are useful for NER particularly when in-domain annotations are scarce. However, with little annotated data, discriminative training may be unable to reliably estimate lexical feature weights and may overfit. In this section, we are interested in evaluating our proposed Bayesian model in the context of low-resource NER.

5.1 Data

Most languages do not have corpora annotated for parts-of-speech, named-entities, syntactic parses, or other linguistic annotations. Therefore, rapidly deploying natural language technologies in a new language may be challenging. In the context of facilitating relief responses in emergencies such as natural disasters, the DARPA LORELEI (Low Resource Languages for Emergent Incidents) program has sponsored the development and release of representative "language packs" for Turkish and Uzbek with more languages planned (Strassel and Tracey, 2016). We use the named-entity annotations as part of these language packs which include persons, locations, organizations, and geo-political entities, in order to explore bootstrapping named-entity recognition from small amounts of data. We consider two types of data: ① in-context annotations, where sentences are fully annotated for named-entities, and ② lexical resources.

The LORELEI language packs lack adequate in-domain lexical resources for our purposes. Therefore, we simulate in-domain lexical resources by holding out portions of the annotated development data and deriving dictionaries and name lists from them. For each label $y \in \{\text{PER}, \text{LOC}, \text{ORG}, \text{GPE}, \text{CONTEXT}\}$, our lexicon for y lists all distinct y-labeled strings that appear in the held-out data. This setup ensures that the labels associated with lexicon entries correspond to the annotation guidelines used in the data we use for evaluation. It avoids possible problems that might arise when leveraging noisy out-of-domain knowledge bases, which we may explore in future.

	Model	Danish	German	Greek	English	Italian	Portuguese	Spanish	Swedish	**Mean**
Wiktionary	SHMM-ME	83.3	85.8	79.2	87.1	86.5	84.5	86.4	86.1	84.9
	HSM	83.7	90.7	81.7	84.0	86.7	85.5	87.6	86.8	85.8
Supervised	SHMM-ME	93.9	97.4	95.1	95.8	93.8	95.5	93.8	95.5	95.1
	HSM	95.2	97.4	97.4	95.2	94.5	96.0	95.6	92.2	95.3

Table 1: Part-of-speech induction results in multiple languages.

5.2 Evaluation

In this section we report supervised NER experiments on two low-resource languages: Turkish and Uzbek. We vary both the amount of supervision as well as the size of the lexical resources. A challenge when evaluating the performance of a model with small amounts of training data is that there may be high-variance in the results. In order to have more confidence in our results, we perform bootstrap resampling experiments in which the training set, evaluation set, and lexical resources are randomized across several replications of the same experiment (for each of the data conditions). We use 10 replications for each of the data conditions reported in Figures 1–2, and report both the mean performance and 95% confidence intervals.

Baseline. We use the Stanford NER system with a standard set of language-independent features (Finkel et al., 2005).[8]. This model is a conditional random field (CRF) with feature templates which include character n-grams as well as word shape features. Crucially, we also incorporate lexical features. The CRF parameters are regularized using an L1 penalty and optimized via Orthant-wise limited-memory quasi-Newton optimization (Andrew and Gao, 2007). For both our proposed method and the discriminative baseline, we use a fixed set of hyperparameters (i.e. we do not use a separate validation set for tuning each data condition). In order to make a fair comparison to the CRF, we use our sampler for forward inference only, without resampling on the test data.

Results. We show learning curves as a function of supervised training corpus size. Figure 1 shows that our generative model strongly beats the baseline in this low-data regime. In particular, when there is little annotated training data, our proposed generative model can compensate by exploiting the lexicon, while the discriminative baseline scores terribly. The performance gap decreases with larger supervised corpora, which is consistent with prior results comparing generative and discriminative training (Ng and Jordan, 2002).

In Figure 2, we show the effect of the lexicon's size: as expected, larger lexicons are better. The generative approach significantly outperforms the discriminative baseline at any lexicon size, although its advantage drops for smaller lexicons or larger training corpora.

In Figure 1 we found that increasing the pseudocount c consistently *decreases* performance, so we used $c = 1$ in our other experiments.[9]

6 Conclusion

This paper has described a generative model for low-resource sequence labeling and segmentation tasks using lexical resources. Experiments in semi-supervised and low-resource settings have demonstrated its applicability to part-of-speech induction and low-resource named-entity recognition. There are many potential avenues for future work. Our model may be useful in the context of active learning where efficient re-estimation and performance in low-data conditions are important. It would also be interesting to explore more expressive parameterizations, such recurrent neural networks for H_y. In the space of neural methods, differentiable memory (Santoro et al., 2016) may be more flexible than the PYP prior, while retaining the ability of the model to cache strings observed in the gazetteer.

Acknowledgments

This work was supported by the JHU Human Language Technology Center of Excellence, DARPA LORELEI, and NSF grant IIS-1423276. Thanks to Jay Feldman for early discussions.

[8] We also experimented with neural models, but found that the CRF outperformed them in low-data conditions.

[9] Why? Even a pseudocount of $c = 1$ is enough to ensure that $P_y(s) \gg H_y(s)$, since the prior probability $H_y(s)$ is rather small for most strings in the lexicon. Indeed, perhaps $c < 1$ would have increased performance, particularly if the lexicon reflects out-of-domain data. This could be arranged, in effect, by using a hierarchical Bayesian model in which the lexicon and corpus emissions are not drawn from the identical distribution P_y but only from similar (coupled) distributions.

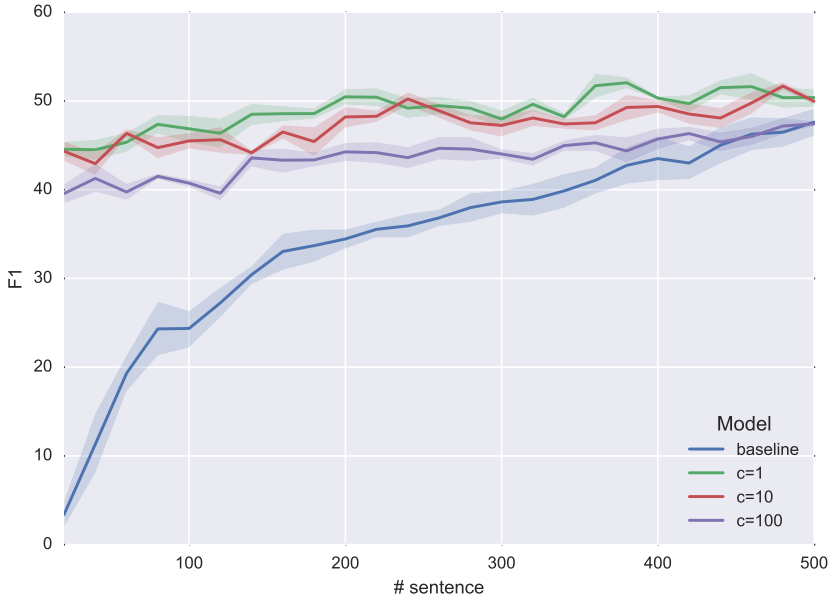


Figure 1: **Absolute** NER performance for Turkish (y-axis) as a function of corpus size (x-axis). The y-axis gives the F1 score on a held-out evaluation set (averaged over 10 bootstrap replicates, with error bars showing 95% confidence intervals). Our generative approach is compared to a baseline discriminative model with lexicon features (lowest curve). 500 held-out sentences were used to create the lexicon for both methods. Note that increasing the pseudocount c for lexicon entries (upper curves) tends to *decrease* performance for the generative model; we therefore take $c = 1$ in all other experiments. This graph shows Turkish; the corresponding Uzbek figure is available as supplementary material.

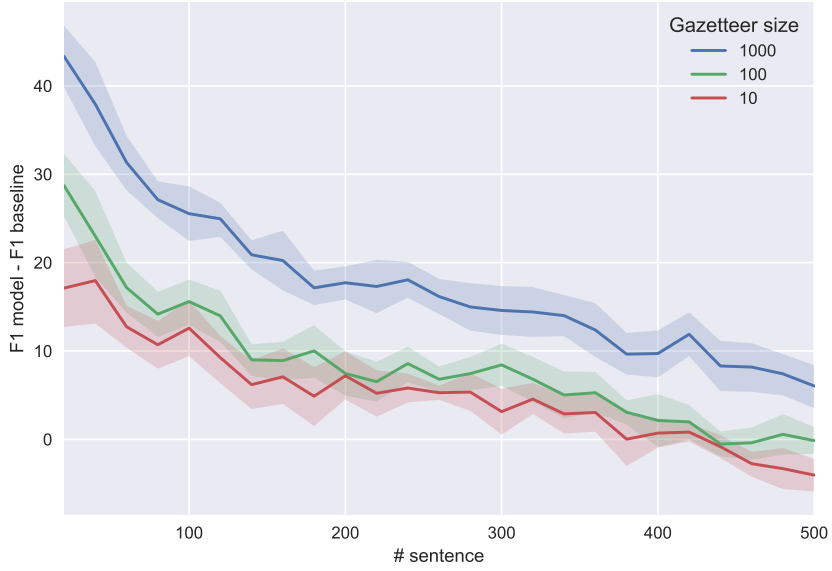


Figure 2: **Relative** NER performance for Turkish (y-axis) as a function of corpus size (x-axis). In this graph, $c = 1$ is constant and the curves instead compare different lexicon sizes derived from 10, 100, and 1000 held-out sentences. The y-axis now gives the *difference* $\text{F1}_{\text{model}} - \text{F1}_{\text{baseline}}$, so positive values indicate *improvement over the baseline* due to the proposed model. Gains are highest for large lexicons and for small corpora. Again, the corresponding Uzbek figure is available as supplementary material.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research* 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*. pages 33–40.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. 2010. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72(3):269–342.

Phil Blunsom and Trevor Cohn. 2010. A hierarchical Pitman-Yor process HMM for unsupervised part-of-speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Benjamin Borschinger and Mark Johnson. 2011. A particle filter algorithm for Bayesian wordsegmentation. In *Proceedings of the Australasian Language Technology Association Workshop 2011*. Canberra, Australia, pages 10–18.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. pages 600–609.

Arnaud Doucet and Adam M. Johansen. 2009. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering* 12:656–704.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a Dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Edinburgh, pages 616–627.

Gregory Dubbin and Phil Blunsom. 2012. Unsupervised Bayesian part of speech inference with particle Gibbs. In *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*. Springer-Verlag, Berlin, Heidelberg, ECML PKDD'12, pages 760–773.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA, ACL '05, pages 363–370.

Dan Garrette and Jason Baldridge. 2012. Type-supervised hidden Markov models for part-of-speech tagging with incomplete tag dictionaries. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 821–831.

Jan Gasthaus and Yee Whye Teh. 2010. Improvements to the sequence memoizer. In *NIPS*. pages 685–693.

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. Prague, Czech Republic, pages 744–751.

Sharon Goldwater, Mark Johnson, and Thomas L. Griffiths. 2005. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems*. pages 459–466.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *HLT-NAACL*. pages 139–146.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *Computing Research Repository* arXiv:1602.02410.

Shen Li, Joao V Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 1389–1398.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. pages 100–108.

Andrew Y. Ng and Michael I. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in Neural Information Processing Systems* 2:841–848.

Jim Pitman. 1996. Some developments of the Blackwell-MacQueen urn scheme. In T. S. Ferguson, L. S. Shapley, and J. B. MacQueen, editors, *Statistics, Probability and Game Theory: Papers in Honor of David Blackwell*, Institute of Mathematical Statistics, volume 30 of *IMS Lecture Notes-Monograph series*, pages 245–267.

Jim Pitman and Marc Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability* pages 855–900.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. 2016. One-shot learning with memory-augmented neural networks. *Computing Research Repository* arXiv:1605.06065.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. pages 18–25.

Andrew Smith and Miles Osborne. 2006. Using gazetteers in discriminative information extraction. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*. pages 133–140.

Stephanie Strassel and Jennifer Tracey. 2016. Lorelei language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2006. Reducing weight undertraining in structured discriminative learning. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Stroudsburg, PA, USA, HLT-NAACL '06, pages 89–95.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. pages 985–992.

Nilesh Tripuraneni, Shixiang Gu, Hong Ge, and Zoubin Ghahramani. 2015. Particle Gibbs for infinite hidden Markov models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 2395–2403.

Frank Wood, Cédric Archambeau, Jan Gasthaus, Lancelot James, and Yee Whye Teh. 2009. A stochastic memoizer for sequence data. In *Proceedings of the 26th Annual International Conference on Machine Learning*. pages 1129–1136.

Semi-Supervised QA with Generative Domain-Adaptive Nets

Zhilin Yang Junjie Hu Ruslan Salakhutdinov William W. Cohen
School of Computer Science
Carnegie Mellon University
{zhiliny,junjieh,rsalakhu,wcohen}@cs.cmu.edu

Abstract

We study the problem of semi-supervised question answering—-utilizing unlabeled text to boost the performance of question answering models. We propose a novel training framework, the *Generative Domain-Adaptive Nets*. In this framework, we train a generative model to generate questions based on the unlabeled text, and combine model-generated questions with human-generated questions for training question answering models. We develop novel domain adaptation algorithms, based on reinforcement learning, to alleviate the discrepancy between the model-generated data distribution and the human-generated data distribution. Experiments show that our proposed framework obtains substantial improvement from unlabeled text.

1 Introduction

Recently, various neural network models were proposed and successfully applied to the tasks of questions answering (QA) and/or reading comprehension (Xiong et al., 2016; Dhingra et al., 2016; Yang et al., 2017). While achieving state-of-the-art performance, these models rely on a large amount of labeled data. However, it is extremely difficult to collect large-scale question answering datasets. Historically, many of the question answering datasets have only thousands of question answering pairs, such as WebQuestions (Berant et al., 2013), MCTest (Richardson et al., 2013), WikiQA (Yang et al., 2015), and TREC-QA (Voorhees and Tice, 2000). Although larger question answering datasets with hundreds of thousands of question-answer pairs have been collected, including SQuAD (Rajpurkar et al., 2016), MSMARCO (Nguyen et al., 2016), and NewsQA (Trischler et al., 2016a), the data collection process is expensive and time-consuming in practice. This hinders real-world applications for domain-specific question answering.

Compared to obtaining labeled question answer pairs, it is trivial to obtain unlabeled text data. In this work, we study the following problem of semi-supervised question answering: is it possible to leverage unlabeled text to boost the performance of question answering models, especially when only a small amount of labeled data is available? The problem is challenging because conventional manifold-based semi-supervised learning algorithms (Zhu and Ghahramani, 2002; Yang et al., 2016a) cannot be straightforwardly applied. Moreover, since the main foci of most question answering tasks are extraction rather than generation, it is also not sensible to use unlabeled text to improve language modeling as in machine translation (Gulcehre et al., 2015).

To better leverage the unlabeled text, we propose a novel neural framework called *Generative Domain-Adaptive Nets* (GDANs). The starting point of our framework is to use linguistic tags to extract possible answer chunks in the unlabeled text, and then train a generative model to generate questions given the answer chunks and their contexts. The model-generated question-answer pairs and the human-generated question-answer pairs can then be combined to train a question answering model, referred to as a *discriminative model* in the following text. However, there is discrepancy between the model-generated data distribution and the human-generated data distribution, which leads to suboptimal discriminative models. To address this issue, we further propose two domain adaptation techniques that treat the model-generated data distribution as a different domain. First, we use an additional *domain tag* to

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1040–1050
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1096

indicate whether a question-answer pair is model-generated or human-generated. We condition the discriminative model on the domain tags so that the discriminative model can learn to factor out domain-specific and domain-invariant representations. Second, we employ a reinforcement learning algorithm to fine-tune the generative model to minimize the loss of the discriminative model in an adversarial way.

In addition, we present a simple and effective baseline method for semi-supervised question answering. Although the baseline method performs worse than our GDAN approach, it is extremely easy to implement and can still lead to substantial improvement when only limited labeled data is available.

We experiment on the SQuAD dataset (Rajpurkar et al., 2016) with various labeling rates and various amounts of unlabeled data. Experimental results show that our GDAN framework consistently improves over both the supervised learning setting and the baseline methods, including adversarial domain adaptation (Ganin and Lempitsky, 2014) and dual learning (Xia et al., 2016). More specifically, the GDAN model improves the F1 score by 9.87 points in F1 over the supervised learning setting when 8K labeled question-answer pairs are used.

Our contribution is four-fold. First, different from most of the previous neural network studies on question answering, we study a critical but challenging problem, semi-supervised question answering. Second, we propose the Generative Domain-Adaptive Nets that employ domain adaptation techniques on generative models with reinforcement learning algorithms. Third, we introduce a simple and effective baseline method. Fourth, we empirically show that our framework leads to substantial improvements.

2 Semi-Supervised Question Answering

Let us first introduce the problem of *semi-supervised question answering*.

Let $L = \{q^{(i)}, a^{(i)}, p^{(i)}\}_{i=1}^{N}$ denote a question answering dataset of N instances, where $q^{(i)}$, $a^{(i)}$, and $p^{(i)}$ are the question, answer, and paragraph of the i-th instance respectively. The goal of question answering is to produce the answer $a^{(i)}$ given the question $q^{(i)}$ along with the paragraph $p^{(i)}$. We will drop the superscript $\cdot^{(i)}$ when the context is unambiguous. In our formulation, following the setting in SQuAD (Rajpurkar et al., 2016), we specifically focus on extractive question answering, where a is always a consecutive chunk of text in p. More formally, let $p = (p_1, p_2, \cdots, p_T)$ be a sequence of word tokens with T being the length, then a can always be represented as $a = (p_j, p_{j+1}, \cdots, p_{k-1}, p_k)$, where j and k are the start and end token indices respectively. The questions can also be represented as a sequence of word tokens $q = (q_1, q_2, \cdots, q_{T'})$ with length T'.

In addition to the labeled dataset L, in the semi-supervised setting, we are also given a set of unlabeled data, denoted as $U = \{a^{(i)}, p^{(i)}\}_{i=1}^{M}$, where M is the number of unlabeled instances. Note that it is usually trivial to have access to an almost infinite number of paragraphs p from sources such as Wikipedia articles and other web pages. And since the answer a is always a consecutive chunk in p, we argue that it is also sensible to extract possible answer chunks from the unlabeled text using linguistic tags. We will discuss the technical details of answer chunk extraction in Section 4.1, and in the formulation of our framework, we assume that the answer chunks a are available.

Given both the labeled data L and the unlabeled data U, the goal of semi-supervised question answering is to learn a question answering model D that captures the probability distribution $\mathbb{P}(a|p, q)$. We refer to this question answering model D as the *discriminative model*, in contrast to the generative model that we will present in Section 3.2.

2.1 A Simple Baseline

We now present a simple baseline for semi-supervised question answering. Given a paragraph $p = (p_1, p_2, \cdots, p_T)$ and the answer $a = (p_j, p_{j+1}, \cdots, p_{k-1}, p_k)$, we extract $(p_{j-W}, p_{j-W+1}, \cdots, p_{j-1}, p_{k+1}, p_{k+2}, p_{k+W})$ from the paragraph and treat it as the question. Here W is the window size and is set at 5 in our experiments so that the lengths of the questions are similar to human-generated questions. The context-based question-answer pairs on U are combined with human-generated pairs on L for training the discriminative model. Intuitively, this method extracts the contexts around the answer chunks to serve as hints for the question answering model. Surprisingly, this simple baseline method leads to substantial improvements when labeled data is limited.

3 Generative Domain-Adaptive Nets

Though the simple method described in Section 2.1 can lead to substantial improvement, we aim to design a learning-based model to move even further. In this section, we will describe the model architecture and the training algorithms for the GDANs. We will use a notation in the context of question answering following Section 2, but one should be able to extend the notion of GDANs to other applications as well.

The GDAN framework consists of two models, *a discriminative model* and a *generative model*. We will first discuss the two models in detail in the context of question answering, and then present an algorithm based on reinforcement learning to combine the two models.

3.1 Discriminative Model

The discriminative model learns the conditional probability of an answer chunk given the paragraph and the question, i.e., $\mathbb{P}(a|p,q)$. We employ a gated-attention (GA) reader (Dhingra et al., 2016) as our base model in this work, but our framework does not make any assumptions about the base models being used. The discriminative model is referred to as D.

The GA model consists of K layers with K being a hyper-parameter. Let $\mathbf{H}_p^k$ be the intermediate paragraph representation at layer k, and $\mathbf{H}_q$ be the question representation. The paragraph representation $\mathbf{H}_p^k$ is a $T \times d$ matrix, and the question representation $\mathbf{H}_q$ is a $T' \times d$ matrix, where d is the dimensionality of the representations. Given the paragraph p, we apply a bidirectional Gated Recurrent Unit (GRU) network (Chung et al., 2014) on top of the embeddings of the sequence $(p_1, p_2, \cdots, p_T)$, and obtain the initial paragraph representation $\mathbf{H}_p^0$. Given the question q, we also apply another bidirectional GRU to obtain the question representation $\mathbf{H}_q$.

The question and paragraph representations are combined with the gated-attention (GA) mechanism (Dhingra et al., 2016). More specifically, for each paragraph token p_i, we compute

$$\alpha_j = \frac{\exp \mathbf{h}_{q,j}^T \mathbf{h}_{p,i}^{k-1}}{\sum_{j'=1}^{T'} \exp \mathbf{h}_{q,j'}^T \mathbf{h}_{p,i}^{k-1}}$$

$$\mathbf{h}_{p,i}^k = \sum_{j=1}^{T'} \alpha_j \mathbf{h}_{q,j} \odot \mathbf{h}_{p,i}^{k-1}$$

where $\mathbf{h}_{p,i}^k$ is the i-th row of $\mathbf{H}_p^k$ and $\mathbf{h}_{q,j}$ is the j-th row of $\mathbf{H}_q$.

Since the answer a is a sequence of consecutive word tokens in the paragraph p, we apply two softmax layers on top of $\mathbf{H}_p^K$ to predict the start and end indices of a, following Yang et al. (2017).

3.1.1 Domain Adaptation with Tags

We will train our discriminative model on both model-generated question-answer pairs and human-generated pairs. However, even a well-trained generative model will produce questions somewhat different from human-generated ones. Learning from both human-generated data and model-generated data can thus lead to a biased model. To alleviate this issue, we propose to view the model-generated data distribution and the human-generated data distribution as two different data domains and explicitly incorporate domain adaptation into the discriminative model.

More specifically, we use a *domain tag* as an additional input to the discriminative model. We use the tag "d_true" to represent the domain of human-generated data (i.e., the *true* data), and "d_gen" for the domain of model-generated data. Following a practice in domain adaptation (Johnson et al., 2016; Chu et al., 2017), we append the domain tag to the end of both the questions and the paragraphs. By introducing the domain tags, we expect the discriminative model to factor out domain-specific and domain-invariant representations. At test time, the tag "d_true" is appended.

3.2 Generative Model

The generative model learns the conditional probability of generating a question given the paragraph and the answer, i.e., $\mathbb{P}(q|p,a)$. We implement the generative model as a sequence-to-sequence model (Sutskever et al., 2014) with a copy mechanism (Gu et al., 2016; Gulcehre et al., 2016).

The generative model consists of an encoder and a decoder. An encoder is a GRU that encodes the input paragraph into a sequence of hidden states $\mathbf{H}$. We inject the answer information by appending an additional zero/one feature to the word embeddings of the paragraph tokens; i.e., if a word token appears in the answer, the feature is set at one, otherwise zero.

The decoder is another GRU with an attention mechanism over the encoder hidden states $\mathbf{H}$. At each time step, the generation probabilities over all

Algorithm 1 Training Generative Domain-Adaptive Nets

Input: labeled data L, unlabeled data U, #iterations T_G and T_D
Initialize G by MLE training on L
Randomly initialize D
while not stopping **do**
  for $t \leftarrow 1$ to T_D **do**
    Update D to maximize $J(L, \text{d\_true}, D) + J(U_G, \text{d\_gen}, D)$ with SGD
  end for
  for $t \leftarrow 1$ to T_G **do**
    Update G to maximize $J(U_G, \text{d\_true}, D)$ with Reinforce and SGD
  end for
end while
return model D

word types are defined with a copy mechanism:

$$\mathbf{p}_{\text{overall}} = g_t \mathbf{p}_{\text{vocab}} + (1 - g_t)\mathbf{p}_{\text{copy}} \quad (1)$$

where g_t is the probability of generating the token from the vocabulary, while $(1 - g_t)$ is the probability of copying a token from the paragraph. The probability g_t is computed based on the current hidden state $\mathbf{h}_t$:

$$g_t = \sigma(\mathbf{w}_g^T \mathbf{h}_t)$$

where σ denotes the logistic function and $\mathbf{w}_g$ is a vector of model parameters. The generation probabilities $\mathbf{p}_{\text{vocab}}$ are defined as a softmax function over the word types in the vocabulary, and the copying probabilities $\mathbf{p}_{\text{copy}}$ are defined as a softmax function over the word types in the paragraph. Both $\mathbf{p}_{\text{vocab}}$ and $\mathbf{p}_{\text{copy}}$ are defined as a function of the current hidden state $\mathbf{h}_t$ and the attention results (Gu et al., 2016).

3.3 Training Algorithm

We first define the objective function of the GDANs, and then present an algorithm to optimize the given objective function. Similar to the Generative Adversarial Nets (GANs) (Goodfellow et al., 2014) and adversarial domain adaptation (Ganin and Lempitsky, 2014), the discriminative model and the generative model have different objectives in our framework. However, rather than formulating the objective as an adversarial game between the two models (Goodfellow et al., 2014; Ganin and Lempitsky, 2014), in our framework, the discriminative model relies on the data generated by the generative model, while the generative model aims to match the model-generated data distribution with the human-generated data distribution using the signals from the discriminative model.

Given a labeled dataset $L = \{p^{(i)}, q^{(i)}, a^{(i)}\}_{i=1}^N$, the objective function of a discriminative model D for a supervised learning setting can be written as $\sum_{p^{(i)},q^{(i)},a^{(i)} \in L} \log \mathbb{P}_D(a^{(i)}|p^{(i)}, q^{(i)})$, where $\mathbb{P}_D$ is a probability distribution defined by the model D. Since we also incorporate domain tags into the model D, we denote the objective function as

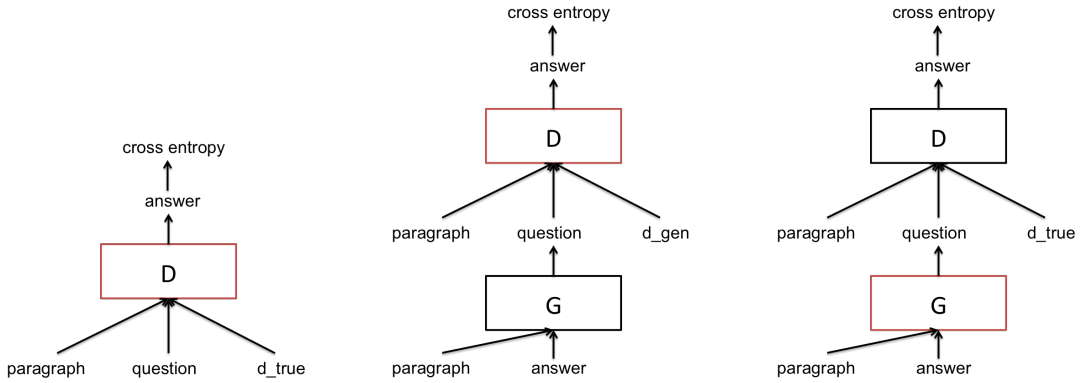
$$J(L, \text{tag}, D) = \frac{1}{|L|} \sum_{p^{(i)},q^{(i)},a^{(i)} \in L} \log \mathbb{P}_{D,\text{tag}}(a^{(i)}|p^{(i)}, q^{(i)})$$

meaning that the domain tag, "tag", is appended to the dataset L. We use $|L| = N$ to denote the number of the instances in the dataset L. The objective function is averaged over all instances such that we can balance labeled and unlabeled data.

Let U_G denote the dataset obtained by generating questions on the unlabeled dataset U with the generative model G. The objective of the discriminative model is then to maximize J for both labeled and unlabeled data under the domain adaptation notions, i.e., $J(L, \text{d\_true}, D) + J(U_G, \text{d\_gen}, D)$.

Now we discuss the objective of the generative model. Similar to the dual learning (Xia et al., 2016) framework, one can define an auto-encoder objective. In this case, the generative model aims to generate questions that can be reconstructed by the discriminative model, i.e., maximizing $J(U_G, \text{d\_gen}, D)$. However, this objective function can lead to degenerate solutions because the questions can be thought of as an overcomplete representation of the answers (Vincent et al., 2010). For example, given p and a, the generative model might learn to generate trivial questions such as copying the answers, which does not contributed to learning a better D.

Instead, we leverage the discriminative model to better match the model-generated data distribution with the human-generated data distribution. We propose to define an adversarial training objective $J(U_G, \text{d\_true}, D)$. We append the tag "d_true" instead of "d_gen" for the model-generated data to "fool" the discriminative model. Intuitively, the goal of G is to generate "useful" questions where the usefulness is measured by the probability that the generated questions can be answered correctly by D.



(a) Training the discriminative model on labeled data.

(b) Training the discriminative model on unlabeled data.

(c) Training the generative model on unlabeled data.

Figure 1: Model architecture and training. Red boxes denote the modules being updated. "d_true" and "d_gen" are two domain tags. D is the discriminative model and G is the generative model. The objectives for the three cases are all to minimize the cross entropy loss of the answer chunks.

The overall objective function now can be written as

$$\begin{aligned} \max_D \quad & J(L, \text{d\_true}, D) + J(U_G, \text{d\_gen}, D) \\ \max_G \quad & J(U_G, \text{d\_true}, D) \end{aligned}$$

With the above objective function in mind, we present a training algorithm in Algorithm 1 to train a GDAN. We first pretrain the generative model on the labeled data L with maximum likelihood estimation (MLE):

$$\max_G \sum_{i=1}^{N} \sum_{t=1}^{T'} \log \mathbb{P}_G(q_t^{(i)} | q_{<t}^{(i)}, p^{(i)}, a^{(i)})$$

where $\mathbb{P}_G$ is the probability defined by Eq. 1.

We then alternatively update D and G based on their objectives. To update D, we sample one batch from the labeled data L and one batch from the unlabeled data U_G, and combine the two batches to perform a gradient update step. Since the output of G is discrete and non-differentiable, we use the Reinforce algorithm (Williams, 1992) to update G. The action space is all possible questions with length T' (possibly with padding) and the reward is the objective function $J(U_G, \text{d\_true}, D)$. Let θ_G be the parameters of G. The gradient can be written as

$$\begin{aligned} & \frac{\partial J(U_G, \text{d\_true}, D)}{\partial \theta_G} \\ = \quad & \mathbb{E}_{\mathbb{P}_G(q|p,a)}(\log \mathbb{P}_{D,\text{d\_true}}(a|p,q) - b) \frac{\partial \log \mathbb{P}_G(q|p,a)}{\partial \theta_G} \end{aligned}$$

where we use an average reward from samples as the baseline b. We approximate the expectation $\mathbb{E}_{\mathbb{P}_G(q|p,a)}$ by sampling one instance at a time from $\mathbb{P}_G(q|p,a)$ and then do an update step. This training algorithm is referred to as reinforcement learning (RL) training in the following sections. The overall architecture and training algorithm are illustrated in Figure 1.

MLE vs RL. The generator G has two training phases–MLE training and RL training, which are different in that: 1) RL training does not require labels, so G can explore a broader data domain of p using unlabeled data, while MLE training requires labels; 2) MLE maximizes $\log P(q|p,a)$, while RL maximizes $\log P_D(a|q,p)$. Since $\log P(q|a,p)$ is the sum of $\log P(q|p)$ and $\log P(a|q,p)$ (plus a constant), maximizing $\log P(a|q,p)$ does not require modeling $\log P(q|p)$ that is irrelevant to QA, which makes optimization easier. Moreover, maximizing $\log P(a|q,p)$ is consistent with the goal of QA.

4 Experiments

4.1 Answer Extraction

As discussed in Section 2, our model assumes that answers are available for unlabeled data. In this section, we introduce how we use linguistic tags and rules to extract answer chunks from unlabeled text.

To extract answers from massive unlabelled Wikipedia articles, we first sample 205,511 Wikipedia articles that are not used in the training, development and test sets in the SQuAD dataset. We extract the paragraphs from each article, and limit the length of each paragraph at the word level to be less than 850. In total, we obtain 950,612

Table 1: Sampled generated questions given the paragraphs and the answers. *P* means paragraphs, *A* means answers, *GQ* means groundtruth questions, and *Q* means questions generated by our models. *MLE* refers to maximum likelihood training, and *RL* refers to reinforcement learning so as to maximize $J(U_G, \text{d\_true}, D)$. We truncate the paragraphs to only show tokens around the answer spans with a window size of 20.

P1: is mediated by ige , which triggers degranulation of mast cells and basophils when cross - linked by antigen . type ii hypersensitivity occurs when antibodies bind to antigens on the patient ' s own cells , marking them for destruction . this **A:** type ii hypersensitivity **GQ:** antibody - dependent hypersensitivity belongs to what class of hypersensitivity ? **Q (MLE):** what was the UNK of the patient ' s own cells ? **Q (RL):** what occurs when antibodies bind to antigens on the patient ' s own cells by antigen when cross
P2: an additional warming of the earth ' s surface . they calculate with confidence that co0 has been responsible for over half the enhanced greenhouse effect . they predict that under a " business as usual " (bau) scenario , **A:** over half **GQ:** how much of the greenhouse effect is due to carbon dioxide ? **Q (MLE):** what is the enhanced greenhouse effect ? **Q (RL):** what the enhanced greenhouse effect that co0 been responsible for
P3:) narrow gauge lines , which are the remnants of five formerly government - owned lines which were built in mountainous areas . **A:** mountainous areas **GQ:** where were the narrow gauge rail lines built in victoria ? **Q (MLE):** what is the government government government - owned lines built ? **Q (RL):** what were the remnants of government - owned lines built in
P4: but not both). in 0000 , bankamericard was renamed and spun off into a separate company known today as visa inc . **A:** visa inc . **GQ:** what present - day company did bankamericard turn into ? **Q (MLE):** what was the separate company bankamericard ? **Q (RL):** what today as bankamericard off into a separate company known today as spun off into a separate company known today
P5: legrande writes that " the formulation of a single all - encompassing definition of the term is extremely difficult , if **A:** legrande **GQ:** who wrote that it is difficult to produce an all inclusive definition of civil disobedience ? **Q (MLE):** what is the term of a single all all all all encompassing definition of a single all **Q (RL):** what writes " the formulation of a single all - encompassing definition of the term all encompassing encompassing encompassing encompassing

paragraphs from unlabelled articles.

Answers in the SQuAD dataset can be categorized into ten types, i.e., "Date", "Other Numeric", "Person", "Location", "Other Entity", "Common Noun Phrase", "Adjective Phrase", "Verb Phrase", "Clause" and "Other" (Rajpurkar et al., 2016). For each paragraph from the unlabeled articles, we utilize Stanford Part-Of-Speech (POS) tagger (Toutanova et al., 2003) to label each word with the corresponding POS tag, and implement a simple constituency parser to extract the noun phrase, verb phrase, adjective and clause based on a small set of constituency grammars. Next, we use Stanford Named Entity Recognizer (NER) (Finkel et al., 2005) to assign each word with one of the seven labels, i.e., "Date", "Money", "Percent", "location", "Organization" and "Time". We then categorize a span of consecutive words with the same NER tags of either "Money" or "Percent" as the answer of the type "Other Numeric". Similarly, we categorize a span of consecutive words with the same NER tags of "Organization" as the answer of the type "Other Entity". Finally, we subsample five answers from all the extracted answers for each paragraph according to the percentage of answer types in the SQuAD dataset. We obtain 4,753,060 answers in total, which is about 50 times larger than the number of answers in the SQuAD dataset.

4.2 Settings and Comparison Methods

The original SQuAD dataset consists of 87,636 training instances and 10,600 development instances. Since the test set is not published, we split 10% of the training set as the test set, and the remaining 90% serves as the actual training set. Instances are split based on articles; i.e., paragraphs in one article always appear in only one set. We tune the hyper-parameters and perform early stopping on the development set using the F1 scores, and the performance is evaluated on the test set using both F1 scores and exact matching (EM) scores (Rajpurkar et al., 2016).

We compare the following methods. **SL** is

the supervised learning setting where we train the model D solely on the labeled data L. **Context** is the simple context-based method described in Section 2.1. **Context + domain** is the "Context" method with domain tags as described in Section 3.1.1. **Gen** is to train a generative model and use the generated questions as additional training data. **Gen + GAN** refers to the domain adaptation method using GANs (Ganin and Lempitsky, 2014); in contrast to the original work, the generative model is updated using Reinforce. **Gen + dual** refers to the dual learning method (Xia et al., 2016). **Gen + domain** is "Gen" with domain tags, while the generative model is trained with MLE and fixed. **Gen + domain + adv** is the approach we propose (Cf. Figure 1 and Algorithm 1), with "adv" meaning adversarial training based on Reinforce. We use our own implementation of "Gen + GAN" and "Gen + dual", since the GAN model (Ganin and Lempitsky, 2014) does not handle discrete features and the dual learning model (Xia et al., 2016) cannot be directly applied to question answering. When implementing these two baselines, we adopt the learning schedule introduced by Ganin and Lempitsky (2014), i.e., gradually increasing the weights of the gradients for the generative model G.

4.3 Results and Analysis

We study the performance of different models with varying labeling rates and unlabeled dataset sizes. Labeling rates are the percentage of training instances that are used to train D. The results are reported in Table 2. Though the unlabeled dataset we collect consists of around 5 million instances, we also sample a subset of around 50,000 instances to evaluate the effects of the size of unlabeled data. The highest labeling rate in Table 2 is 0.9 because 10% of the training instances are used for testing. Since we do early stopping on the development set using the F1 scores, we also report the development F1. We report two metrics, the F1 scores and the exact matching (EM) scores (Rajpurkar et al., 2016), on the test set. All metrics are computed using the official evaluation scripts.

SL v.s. SSL. We observe that semi-supervised learning leads to consistent improvements over supervised learning in all cases. Such improvements are substantial when labeled data is limited. For example, the GDANs improve over supervised learning by 9.87 points in F1 and 7.26 points in EM when the labeling rate is 0.1. With our semi-supervised learning approach, we can use only 0.1 training instances to obtain even better performance than a supervised learning approach with 0.2 training instances, saving more than half of the labeling costs.

Comparison with Baselines. By comparing "Gen + domain + adv" with "Gen + GAN" and "Gen + Dual", it is clear that the GDANs perform substantially better than GANs and dual learning. With labeling rate 0.1, GDANs outperform dual learning and GANs by 2.47 and 4.29 points respectively in terms of F1.

Ablation Study. We also perform an ablation study by examining the effects of "domain" and "adv" when added to "gen". It can be seen that both the domain tags and the adversarial training contribute to the performance of the GDANs when the labeling rate is equal to or less than 0.5. With labeling rate 0.9, adding domain tags still leads to better performance but adversarial training does not seem to improve the performance by much.

Unlabeled Data Size. Moreover, we observe that the performance can be further improved when a larger unlabeled dataset is used, though the gain is relatively less significant compared to changing the model architectures. For example, increasing the unlabeled dataset size from 50K to 5M, the performance of GDANs increases by 0.38 points in F1 and 0.52 points in EM.

Context-Based Method. Surprisingly, the simple context-based method, though performing worse than GDANs, still leads to substantial gains; e.g., 7.00 points in F1 with labeling rate 0.1. Adding domain tags can improve the performance of the context-based method as well.

MLE vs RL. We plot the loss curve of $-J(U_G, \text{d\_gen}, D)$ for both the MLE-trained generator ("Gen + domain") and the RL-trained generator ("Gen + domain + adv") in Figure 2. We observe that the training loss for D on RL-generated questions is lower than MLE-generated questions, which confirms that RL training maximizes $\log P(a|p, q)$.

Samples of Generated Questions. We present some questions generated by our model in Table 1. The generated questions are post-processed by removing repeated subs-sequences. Compared to MLE-generated questions, RL-generated questions are more informative (Cf., P1, P2, and P4), and contain less "UNK" (unknown) tokens (Cf.,

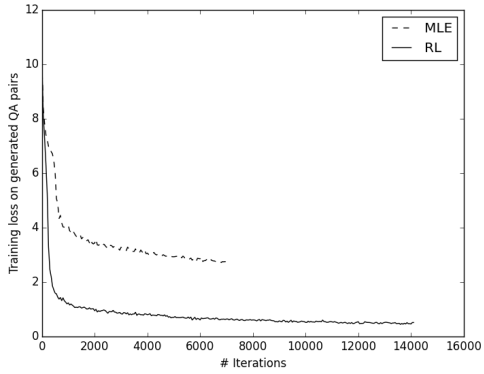


Figure 2: Comparison of discriminator training loss $-J(U_G, \text{d\_gen}, D)$ on generated QA pairs. The lower the better. MLE refers to questions generated by maximum likelihood training, and RL refers to questions generated by reinforcement learning.

P1). Moreover, both semantically and syntactically, RL-generated questions are more accurate (Cf., P3 and P5).

5 Related Work

Semi-Supervised Learning. Semi-supervised learning has been extensively studied in literature (Zhu, 2005). A batch of novel models have been recently proposed for semi-supervised learning based on representation learning techniques, such as generative models (Kingma et al., 2014), ladder networks (Rasmus et al., 2015) and graph embeddings (Yang et al., 2016a). However, most of the semi-supervised learning methods are based on combinations of the supervised loss $p(\mathbf{y}|\mathbf{x})$ and an unsupervised loss $p(\mathbf{x})$. In the context of reading comprehension, directly modeling the likelihood of a paragraph would not possibly improve the supervised task of question answering. Moreover, traditional graph-based semi-supervised learning (Zhu and Ghahramani, 2002) cannot be easily extended to modeling the unlabeled answer chunks.

Domain Adaptation. Domain adaptation has been successfully applied to various tasks, such as classification (Ganin and Lempitsky, 2014) and machine translation (Johnson et al., 2016; Chu et al., 2017). Several techniques on domain adaptation (Glorot et al., 2011) focus on learning distribution invariant features by sharing the intermediate representations for downstream tasks. Another line of research on domain adaptation attempt to match the distance between different domain distributions in a low dimensional space (Long et al., 2015; Baktashmotlagh et al., 2013). There are also methods seeking a domain transition from the source domain to the target domain (Gong et al., 2012; Gopalan et al., 2011; Pan et al., 2011). Our work gets inspiration from a practice in Johnson et al. (2016) and Chu et al. (2017) based on appending domain tags. However, our method is different from the above methods in that we apply domain adaptation techniques to the outputs of a generative model rather than a natural data domain.

Question Answering. Various neural models based on attention mechanisms (Wang and Jiang, 2016; Seo et al., 2016; Xiong et al., 2016; Wang et al., 2016; Dhingra et al., 2016; Kadlec et al., 2016; Trischler et al., 2016b; Sordoni et al., 2016; Cui et al., 2016; Chen et al., 2016) have been proposed to tackle the tasks of question answering and reading comprehension. However, the performance of these neural models largely relies on a large amount of labeled data available for training.

Learning with Multiple Models. GANs (Goodfellow et al., 2014) formulated a adversarial game between a discriminative model and a generative model for generating realistic images. Ganin and Lempitsky (Ganin and Lempitsky, 2014) employed a similar idea to use two models for domain adaptation. Review networks (Yang et al., 2016b) employ a discriminative model as a regularizer for training a generative model. In the context of machine translation, given a language pair, various recent work studied jointly training models to learn the mappings in both directions (Tu et al., 2016; Xia et al., 2016).

6 Conclusions

We study a critical and challenging problem, semi-supervised question answering. We propose a novel neural framework called Generative Domain-Adaptive Nets, which incorporate domain adaptation techniques in combination with generative models for semi-supervised learning. Empirically, we show that our approach leads to substantial improvements over supervised learning models and outperforms several strong baselines including GANs and dual learning. In the future, we plan to apply our approach to more question answering datasets in different domains. It will also be intriguing to generalize GDANs to other applications.

Acknowledgements. This work was funded by the Office of Naval Research grants N000141512791 and N000141310721 and NVIDIA.

Table 2: Performance with various labeling rates, unlabeled data sizes $|U|$, and methods. "Dev" denotes the development set, and "test" denotes the test set. F1 and EM are two metrics.

Labeling rate	$\|U\|$	Method	Dev F1	Test F1	Test EM
0.1	50K	SL	0.4262	0.3815	0.2492
0.1	50K	Context	0.5046	0.4515	0.2966
0.1	50K	Context + domain	0.5139	0.4575	0.3036
0.1	50K	Gen	0.5049	0.4553	0.3018
0.1	50K	Gen + GAN	0.4897	0.4373	0.2885
0.1	50K	Gen + dual	0.5036	0.4555	0.3005
0.1	50K	Gen + domain	0.5234	0.4703	0.3145
0.1	50K	Gen + domain + adv	**0.5313**	**0.4802**	**0.3218**
0.2	50K	SL	0.5134	0.4674	0.3163
0.2	50K	Context	0.5652	0.5132	0.3573
0.2	50K	Context + domain	0.5672	0.5200	0.3581
0.2	50K	Gen	0.5643	0.5159	0.3618
0.2	50K	Gen + GAN	0.5525	0.5037	0.3470
0.2	50K	Gen + dual	0.5720	0.5192	0.3612
0.2	50K	Gen + domain	0.5749	0.5216	0.3658
0.2	50K	Gen + domain + adv	**0.5867**	**0.5394**	**0.3781**
0.5	50K	SL	0.6280	0.5722	0.4187
0.5	50K	Context	0.6300	0.5740	0.4195
0.5	50K	Context + domain	0.6307	0.5791	0.4237
0.5	50K	Gen	0.6237	0.5717	0.4155
0.5	50K	Gen + GAN	0.6110	0.5590	0.4044
0.5	50K	Gen + dual	0.6368	0.5746	0.4163
0.5	50K	Gen + domain	**0.6378**	0.5826	0.4261
0.5	50K	Gen + domain + adv	0.6375	**0.5831**	**0.4267**
0.9	50K	SL	**0.6611**	0.6070	0.4534
0.9	50K	Context	0.6560	0.6028	0.4507
0.9	50K	Context + domain	0.6553	**0.6105**	0.4557
0.9	50K	Gen	0.6464	0.5970	0.4445
0.9	50K	Gen + GAN	0.6396	0.5874	0.4317
0.9	50K	Gen + dual	0.6511	0.5892	0.4340
0.9	50K	Gen + domain	**0.6611**	0.6102	**0.4573**
0.9	50K	Gen + domain + adv	0.6585	0.6043	0.4497
0.1	5M	SL	0.4262	0.3815	0.2492
0.1	5M	Context	0.5140	0.4641	0.3014
0.1	5M	Context + domain	0.5166	0.4599	0.3083
0.1	5M	Gen	0.5099	0.4619	0.3103
0.1	5M	Gen + domain	0.5301	0.4703	0.3227
0.1	5M	Gen + domain + adv	**0.5442**	**0.4840**	**0.3270**
0.9	5M	SL	0.6611	0.6070	0.4534
0.9	5M	Context	0.6605	0.6026	0.4473
0.9	5M	Context + domain	0.6642	0.6066	0.4548
0.9	5M	Gen	0.6647	0.6065	**0.4600**
0.9	5M	Gen + domain	**0.6726**	0.6092	0.4599
0.9	5M	Gen + domain + adv	0.6670	**0.6102**	0.4531

References

Mahsa Baktashmotlagh, Mehrtash T Harandi, Brian C Lovell, and Mathieu Salzmann. 2013. Unsupervised domain adaptation by domain invariant projection. In *ICCV*. pages 769–776.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858* .

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint arXiv:1701.03214* .

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423* .

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* .

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*. Association for Computational Linguistics, pages 363–370.

Yaroslav Ganin and Victor Lempitsky. 2014. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495* .

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*. pages 513–520.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*. IEEE, pages 2066–2073.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*. pages 2672–2680.

Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. 2011. Domain adaptation for object recognition: An unsupervised approach. In *ICCV*. IEEE, pages 999–1006.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* .

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* .

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535* .

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558* .

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547* .

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *NIPS*. pages 3581–3589.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. 2015. Learning transferable features with deep adaptation networks. In *ICML*. pages 97–105.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* .

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks* 22(2):199–210.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. 2015. Semi-supervised learning with ladder networks. In *NIPS*. pages 3546–3554.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*. volume 3.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. pages 3104–3112.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*. Association for Computational Linguistics, pages 173–180.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016a. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830* .

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. 2016b. Natural language comprehension with the epireader. *arXiv preprint arXiv:1606.02270* .

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL*.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR* 11(Dec):3371–3408.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *SIGIR*. ACM, pages 200–207.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* .

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. *arXiv preprint arXiv:1611.00179* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*. Citeseer, pages 2013–2018.

Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2016a. Revisiting semi-supervised learning with graph embeddings. In *ICML*.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. 2017. Words or characters? fine-grained gating for reading comprehension. In *ICLR*.

Zhilin Yang, Ye Yuan, Yuexin Wu, William W Cohen, and Ruslan R Salakhutdinov. 2016b. Review networks for caption generation. In *NIPS*. pages 2361–2369.

Xiaojin Zhu. 2005. Semi-supervised learning literature survey .

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation .

From Language to Programs: Bridging Reinforcement Learning and Maximum Marginal Likelihood

Kelvin Guu
Statistics
Stanford University
kguu@stanford.edu

Panupong Pasupat
Computer Science
Stanford University
ppasupat@stanford.edu

Evan Zheran Liu
Computer Science
Stanford University
evanliu@stanford.edu

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

Abstract

Our goal is to learn a semantic parser that maps natural language utterances into executable programs when only indirect supervision is available: examples are labeled with the correct execution result, but not the program itself. Consequently, we must search the space of programs for those that output the correct result, while not being misled by *spurious programs*: incorrect programs that coincidentally output the correct result. We connect two common learning paradigms, reinforcement learning (RL) and maximum marginal likelihood (MML), and then present a new learning algorithm that combines the strengths of both. The new algorithm guards against spurious programs by combining the systematic search traditionally employed in MML with the randomized exploration of RL, and by updating parameters such that probability is spread more evenly across consistent programs. We apply our learning algorithm to a new neural semantic parser and show significant gains over existing state-of-the-art results on a recent context-dependent semantic parsing task.

1 Introduction

We are interested in learning a semantic parser that maps natural language utterances into executable programs (e.g., logical forms). For example, in Figure 1, a program corresponding to the utterance transforms an initial world state into a new world state. We would like to learn from indirect supervision, where each training example is only labeled with the correct output (e.g. a target world state), but not the program that produced that output (Clarke et al., 2010; Liang et al., 2011; Krishnamurthy and Mitchell, 2012; Artzi and Zettlemoyer, 2013; Liang et al., 2017).

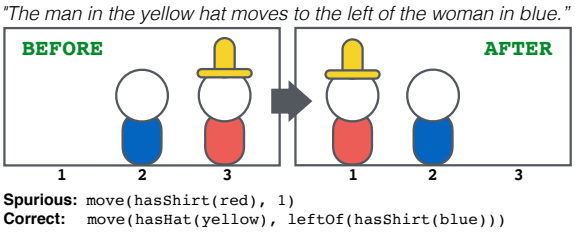


Figure 1: The task is to map natural language utterances to a program that manipulates the world state. The correct program captures the true meaning of the utterances, while spurious programs arrive at the correct output for the wrong reasons. We develop methods to prevent the model from being drawn to spurious programs.

The process of constructing a program can be formulated as a sequential decision-making process, where feedback is only received at the end of the sequence when the completed program is executed. In the natural language processing literature, there are two common approaches for handling this situation: 1) reinforcement learning (RL), particularly the REINFORCE algorithm (Williams, 1992; Sutton et al., 1999), which maximizes the expected reward of a sequence of actions; and 2) maximum marginal likelihood (MML), which treats the sequence of actions as a latent variable, and then maximizes the marginal likelihood of observing the correct program output (Dempster et al., 1977).

While the two approaches have enjoyed success on many tasks, we found them to work poorly out of the box for our task. This is because in addition to the sparsity of correct programs, our task also requires weeding out *spurious programs* (Pasupat and Liang, 2016): incorrect interpretations

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1051–1062
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1097

of the utterances that accidentally produce the correct output, as illustrated in Figure 1.

We show that MML and RL optimize closely related objectives. Furthermore, both MML and RL methods have a mechanism for exploring program space in search of programs that generate the correct output. We explain why this exploration tends to quickly concentrate around short spurious programs, causing the model to sometimes overlook the correct program. To address this problem, we propose RANDOMER, a new learning algorithm with two parts:

First, we propose *randomized beam search*, an exploration strategy which combines the systematic beam search traditionally employed in MML with the randomized off-policy exploration of RL. This increases the chance of finding correct programs even when the beam size is small or the parameters are not pre-trained.

Second, we observe that even with good exploration, the gradients of both the RL and MML objectives may still upweight entrenched spurious programs more strongly than correct programs with low probability under the current model. We propose a *meritocratic parameter update rule*, a modification to the MML gradient update, which more equally upweights all programs that produce the correct output. This makes the model less likely to overfit spurious programs.

We apply RANDOMER to train a new neural semantic parser, which outputs programs in a stack-based programming language. We evaluate our resulting system on SCONE, the context-dependent semantic parsing dataset of Long et al. (2016). Our approach outperforms standard RL and MML methods in a direct comparison, and achieves new state-of-the-art results, improving over Long et al. (2016) in all three domains of SCONE, and by over 30% accuracy on the most challenging one.

2 Task

We consider the semantic parsing task in the SCONE dataset[1] (Long et al., 2016). As illustrated in Figure 1, each example consists of a *world* containing several objects (e.g., people), each with certain properties (e.g., shirt color and hat color). Given the initial world state w_0 and a sequence of M natural language utterances $\mathbf{u} = (u_1, \ldots, u_M)$, the task is to generate a program that manipulates the world state according to the utterances. Each utterance u_m describes a single action that transforms the world state w_{m-1} into a new world state w_m. For training, the system receives weakly supervised examples with input $x = (\mathbf{u}, w_0)$ and the target final world state $y = w_M$.

The dataset includes 3 domains: ALCHEMY, TANGRAMS, and SCENE. The description of each domain can be found in Appendix B. The domains highlight different linguistic phenomena: ALCHEMY features ellipsis (e.g., "throw the rest out", "mix"); TANGRAMS features anaphora on actions (e.g., "repeat step 3", "bring it back"); and SCENE features anaphora on entities (e.g., "he moves back", "...to his left"). Each domain contains roughly 3,700 training and 900 test examples. Each example contains 5 utterances and is labeled with the target world state after each utterance, but not the target program.

Spurious programs. Given a training example $(\mathbf{u}, w_0, w_M)$, our goal is to find the true underlying program $\mathbf{z}^*$ which reflects the meaning of $\mathbf{u}$. The constraint that $\mathbf{z}^*$ must transform w_0 into w_M, i.e. $\mathbf{z}(w_0) = w_M$, is not enough to uniquely identify the true $\mathbf{z}^*$, as there are often many $\mathbf{z}$ satisfying $\mathbf{z}(w_0) = w_M$: in our experiments, we found at least 1600 on average for each example. Almost all do not capture the meaning of $\mathbf{u}$ (see Figure 1). We refer to these incorrect $\mathbf{z}$'s as *spurious programs*. Such programs encourage the model to learn an incorrect mapping from language to program operations: e.g., the spurious program in Figure 1 would cause the model to learn that "man in the yellow hat" maps to `hasShirt(red)`.

Spurious programs in SCONE. In this dataset, utterances often reference objects in different ways (e.g. a person can be referenced by shirt color, hat color, or position). Hence, any target programming language must also support these different reference strategies. As a result, even a single action such as moving a person to a target destination can be achieved by many different programs, each selecting the person and destination in a different way. Across multiple actions, the number of programs grows combinatorially.[2] Only a few programs actually implement the correct reference strategy as defined by the utterance. This problem would be more severe in any more general-purpose language (e.g. Python).

[1] https://nlp.stanford.edu/projects/scone

[2] The number of well-formed programs in SCENE exceeds 10^{15}

3 Model

We formulate program generation as a sequence prediction problem. We represent a program as a sequence of program tokens in postfix notation; for example, `move(hasHat(yellow), leftOf(hasShirt(blue)))` is linearized as `yellow hasHat blue hasShirt leftOf move`. This representation also allows us to incrementally execute programs from left to right using a stack: constants (e.g., `yellow`) are pushed onto the stack, while functions (e.g., `hasHat`) pop appropriate arguments from the stack and push back the computed result (e.g., the list of people with yellow hats). Appendix B lists the full set of program tokens, $\mathcal{Z}$, and how they are executed. Note that each action always ends with an *action token* (e.g., `move`).

Given an input $x = (\mathbf{u}, w_0)$, the model generates program tokens $z_1, z_2, \ldots$ from left to right using a neural encoder-decoder model with attention (Bahdanau et al., 2015). Throughout the generation process, the model maintains an utterance pointer, m, initialized to 1. To generate z_t, the model's encoder first encodes the utterance u_m into a vector e_m. Then, based on e_m and previously generated tokens $z_{1:t-1}$, the model's decoder defines a distribution $p(z_t \mid x, z_{1:t-1})$ over the possible values of $z_t \in \mathcal{Z}$. The next token z_t is sampled from this distribution. If an action token (e.g., `move`) is generated, the model increments the utterance pointer m. The process terminates when all M utterances are processed. The final probability of generating a particular program $\mathbf{z} = (z_1, \ldots, z_T)$ is $p(\mathbf{z} \mid x) = \prod_{t=1}^{T} p(z_t \mid x, z_{1:t-1})$.

Encoder. The utterance u_m under the pointer is encoded using a bidirectional LSTM:

$$h_i^{\mathrm{F}} = \mathrm{LSTM}(h_{i-1}^{\mathrm{F}}, \Phi_{\mathrm{u}}(u_{m,i}))$$
$$h_i^{\mathrm{B}} = \mathrm{LSTM}(h_{i+1}^{\mathrm{B}}, \Phi_{\mathrm{u}}(u_{m,i}))$$
$$h_i = [h_i^{\mathrm{F}}; h_i^{\mathrm{B}}],$$

where $\Phi_{\mathrm{u}}(u_{m,i})$ is the fixed GloVe word embedding (Pennington et al., 2014) of the ith word in u_m. The final utterance embedding is the concatenation $e_m = [h_{|u_m|}^{\mathrm{F}}; h_1^{\mathrm{B}}]$.

Decoder. Unlike Bahdanau et al. (2015), which used a recurrent network for the decoder, we opt for a feed-forward network for simplicity. We use e_m and an embedding $f(z_{1:t-1})$ of the previous execution history (described later) as inputs to compute an attention vector c_t:

$$q_t = \mathrm{ReLU}(W_{\mathrm{q}}[e_m; f(z_{1:t-1})])$$
$$\alpha_i \propto \exp(q_t^\top W_{\mathrm{a}} h_i) \qquad (i = 1, \ldots, |u_m|)$$
$$c_t = \sum_i \alpha_i h_i.$$

Finally, after concatenating q_t with c_t, the distribution over the set $\mathcal{Z}$ of possible program tokens is computed via a softmax:

$$p(z_t \mid x, z_{1:t-1}) \propto \exp(\Phi_{\mathrm{z}}(z_t)^\top W_{\mathrm{s}}[q_t; c_t]),$$

where $\Phi_{\mathrm{z}}(z_t)$ is the embedding for token z_t.

Execution history embedding. We compare two options for $f(z_{1:t-1})$, our embedding of the execution history. A standard approach is to simply take the k most recent tokens $z_{t-k:t-1}$ and concatenate their embeddings. We will refer to this as TOKENS and use $k = 4$ in our experiments.

We also consider a new approach which leverages our ability to incrementally execute programs using a stack. We summarize the execution history by embedding the state of the stack at time $t-1$, achieved by concatenating the embeddings of all values on the stack. (We limit the maximum stack size to 3.) We refer to this as STACK.

4 Reinforcement learning versus maximum marginal likelihood

Having formulated our task as a sequence prediction problem, we must still choose a learning algorithm. We first compare two standard paradigms: reinforcement learning (RL) and maximum marginal likelihood (MML). In the next section, we propose a better alternative.

4.1 Comparing objective functions

Reinforcement learning. From an RL perspective, given a training example (x, y), a policy makes a sequence of decisions $\mathbf{z} = (z_1, \ldots, z_T)$, and then receives a reward at the end of the episode: $R(\mathbf{z}) = 1$ if $\mathbf{z}$ executes to y and 0 otherwise (dependence on x and y has been omitted from the notation).

We focus on *policy gradient* methods, in which a stochastic policy function is trained to maximize the expected reward. In our setup, $p_\theta(\mathbf{z} \mid x)$ is the policy (with parameters θ), and its expected reward on a given example (x, y) is

$$G(x, y) = \sum_{\mathbf{z}} R(\mathbf{z})\, p_\theta(\mathbf{z} \mid x), \qquad (1)$$

where the sum is over all possible programs. The overall RL objective, J_{RL}, is the expected reward across examples:

$$J_{\text{RL}} = \sum_{(x,y)} G(x, y). \tag{2}$$

Maximum marginal likelihood. The MML perspective assumes that y is generated by a partially-observed random process: conditioned on x, a latent program $\mathbf{z}$ is generated, and conditioned on $\mathbf{z}$, the observation y is generated. This implies the marginal likelihood:

$$p_\theta(y \mid x) = \sum_{\mathbf{z}} p(y \mid \mathbf{z})\, p_\theta(\mathbf{z} \mid x). \tag{3}$$

Note that since the execution of $\mathbf{z}$ is deterministic, $p_\theta(y \mid \mathbf{z}) = 1$ if $\mathbf{z}$ executes to y and 0 otherwise. The log marginal likelihood of the data is then

$$J_{\text{MML}} = \log \mathcal{L}_{\text{MML}}, \tag{4}$$

$$\text{where} \quad \mathcal{L}_{\text{MML}} = \prod_{(x,y)} p_\theta(y \mid x). \tag{5}$$

To estimate our model parameters θ, we maximize J_{MML} with respect to θ.

With our choice of reward, the RL expected reward (1) is equal to the MML marginal probability (3). Hence the only difference between the two formulations is that in RL we optimize the *sum* of expected rewards (2), whereas in MML we optimize the *product* (5).[3]

4.2 Comparing gradients

In both policy gradient and MML, the objectives are typically optimized via (stochastic) gradient ascent. The gradients of J_{RL} and J_{MML} are closely related. They both have the form:

$$\begin{aligned} \nabla_\theta J &= \sum_{(x,y)} \mathbb{E}_{\mathbf{z}\sim q}\left[R(\mathbf{z}) \nabla \log p_\theta(\mathbf{z} \mid x)\right] \\ &= \sum_{(x,y)} \sum_{\mathbf{z}} q(\mathbf{z}) R(\mathbf{z}) \nabla \log p_\theta(\mathbf{z} \mid x), \end{aligned} \tag{6}$$

where $q(\mathbf{z})$ equals

$$q_{\text{RL}}(\mathbf{z}) = p_\theta(\mathbf{z} \mid x) \quad \text{for } J_{\text{RL}}, \tag{7}$$

$$\begin{aligned} q_{\text{MML}}(\mathbf{z}) &= \frac{R(\mathbf{z}) p_\theta(\mathbf{z} \mid x)}{\sum_{\tilde{\mathbf{z}}} R(\tilde{\mathbf{z}}) p_\theta(\tilde{\mathbf{z}} \mid x)} \\ &= p_\theta(\mathbf{z} \mid x, R(\mathbf{z}) \neq 0) \quad \text{for } J_{\text{MML}}. \end{aligned} \tag{8}$$

[3] Note that the log of the product in (5) does *not* equal the sum in (2).

Taking a step in the direction of $\nabla \log p_\theta(\mathbf{z} \mid x)$ upweights the probability of $\mathbf{z}$, so we can heuristically think of the gradient as attempting to upweight each reward-earning program $\mathbf{z}$ by a *gradient weight* $q(\mathbf{z})$. In Subsection 5.2, we argue why q_{MML} is better at guarding against spurious programs, and propose an even better alternative.

4.3 Comparing gradient approximation strategies

It is often intractable to compute the gradient (6) because it involves taking an expectation over all possible programs. So in practice, the expectation is approximated.

In the policy gradient literature, *Monte Carlo integration* (MC) is the typical approximation strategy. For example, the popular REINFORCE algorithm (Williams, 1992) uses Monte Carlo sampling to compute an unbiased estimate of the gradient:

$$\Delta_{\text{MC}} = \frac{1}{B} \sum_{\mathbf{z}\in\mathcal{S}} [R(\mathbf{z}) - c] \nabla \log p_\theta(\mathbf{z} \mid x), \tag{9}$$

where $\mathcal{S}$ is a collection of B samples $\mathbf{z}^{(b)} \sim q(\mathbf{z})$, and c is a *baseline* (Williams, 1992) used to reduce the variance of the estimate without altering its expectation.

In the MML literature for latent sequences, the expectation is typically approximated via *numerical integration* (NUM) instead:

$$\Delta_{\text{NUM}} = \sum_{\mathbf{z}\in\mathcal{S}} q(\mathbf{z}) R(\mathbf{z}) \nabla \log p_\theta(\mathbf{z} \mid x). \tag{10}$$

where the programs in $\mathcal{S}$ come from beam search.

Beam search. Beam search generates a set of programs via the following process. At step t of beam search, we maintain a beam $\mathcal{B}_t$ of at most B *search states*. Each state $s \in \mathcal{B}_t$ represents a partially constructed program, $s = (z_1, \ldots, z_t)$ (the first t tokens of the program). For each state s in the beam, we generate all possible *continuations*,

$$\begin{aligned} \texttt{cont}(s) &= \texttt{cont}((z_1, \ldots, z_t)) \\ &= \{(z_1, \ldots, z_t, z_{t+1}) \mid z_{t+1} \in \mathcal{Z}\}. \end{aligned}$$

We then take the union of these continuations, $\texttt{cont}(\mathcal{B}_t) = \bigcup_{s\in\mathcal{B}_t} \texttt{cont}(s)$. The new beam $\mathcal{B}_{t+1}$ is simply the highest scoring B continuations in $\texttt{cont}(\mathcal{B}_t)$, as scored by the policy, $p_\theta(s \mid x)$. Search is halted after a fixed number of iterations

or when there are no continuations possible. $\mathcal{S}$ is then the set of all complete programs discovered during beam search. We will refer to this as *beam search MML* (BS-MML).

In both policy gradient and MML, we think of the procedure used to produce the set of programs $\mathcal{S}$ as an *exploration strategy* which searches for programs that produce reward. One advantage of numerical integration is that it allows us to decouple the *exploration strategy* from the *gradient weights* assigned to each program.

5 Tackling spurious programs

In this section, we illustrate why spurious programs are problematic for the most commonly used methods in RL (REINFORCE) and MML (beam search MML). We describe two key problems and propose a solution to each, based on insights gained from our comparison of RL and MML in Section 4.

5.1 Spurious programs bias exploration

As mentioned in Section 4, REINFORCE and BS-MML both employ an *exploration strategy* to approximate their respective gradients. In both methods, exploration is guided by the current model policy, whereby programs with high probability under the current policy are more likely to be explored. A troubling implication is that programs with low probability under the current policy are likely to be overlooked by exploration.

If the current policy incorrectly assigns low probability to the correct program $\mathbf{z}^*$, it will likely fail to discover $\mathbf{z}^*$ during exploration, and will consequently fail to upweight the probability of $\mathbf{z}^*$. This repeats on every gradient step, keeping the probability of $\mathbf{z}^*$ perpetually low. The same feedback loop can also cause already high-probability spurious programs to gain even more probability. From this, we see that exploration is sensitive to initial conditions: the rich get richer, and the poor get poorer.

Since there are often thousands of spurious programs and only a few correct programs, spurious programs are usually found first. Once spurious programs get a head start, exploration increasingly biases towards them.

As a remedy, one could try initializing parameters such that the model puts a uniform distribution over all possible programs. A seemingly reasonable tactic is to initialize parameters such that the model policy puts near-uniform probability over the decisions at each time step. However, this causes shorter programs to have orders of magnitude higher probability than longer programs, as illustrated in Figure 2 and as we empirically observe. A more sophisticated approach might involve approximating the total number of programs reachable from each point in the program-generating decision tree. However, we instead propose to reduce sensitivity to the initial distribution over programs.

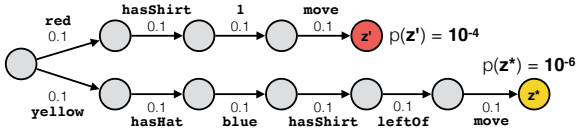


Figure 2: Two possible paths in the tree of all possible programs. One path leads to the spurious program $\mathbf{z}'$ (red) while the longer path leads to the correct program $\mathbf{z}^*$ (gold). Each edge represents a decision and shows the probability of that decision under a uniform policy. The shorter program has two orders of magnitude higher probability.

Solution: randomized beam search

One solution to biased exploration is to simply rely less on the untrustworthy current policy. We can do this by injecting random noise into exploration.

In REINFORCE, a common solution is to sample from an ϵ-greedy variant of the current policy. On the other hand, MML exploration with beam search is deterministic. However, it has a key advantage over REINFORCE-style sampling: even if one program occupies almost all probability under the current policy (a peaky distribution), beam search will still use its remaining beam capacity to explore at least $B-1$ other programs. In contrast, sampling methods will repeatedly visit the mode of the distribution.

To get the best of both worlds, we propose a simple *ϵ-greedy randomized beam search*. Like regular beam search, at iteration t we compute the set of all continuations $\texttt{cont}(\mathcal{B}_t)$ and sort them by their model probability $p_\theta(s \mid x)$. But instead of selecting the B highest-scoring continuations, we choose B continuations one by one without replacement from $\texttt{cont}(\mathcal{B}_t)$. When choosing a continuation from the remaining pool, we either uniformly sample a random continuation with probability ϵ, or pick the highest-scoring continuation in the pool with probability $1-\epsilon$. Empirically, we

find that this performs much better than both classic beam search and ϵ-greedy sampling (Table 3).

5.2 Spurious programs dominate gradients

In both RL and MML, even if exploration is perfect and the gradient is exactly computed, spurious programs can still be problematic.

Even if perfect exploration visits every program, we see from the gradient weights $q(\mathbf{z})$ in (7) and (8) that programs are weighted proportional to their current policy probability. If a spurious program $\mathbf{z}'$ has 100 times higher probability than $\mathbf{z}^*$ as in Figure 2, the gradient will spend roughly 99% of its magnitude upweighting towards $\mathbf{z}'$ and only 1% towards $\mathbf{z}^*$ even though the two programs get the same reward.

This implies that it would take many updates for $\mathbf{z}^*$ to catch up. In fact, $\mathbf{z}^*$ may never catch up, depending on the gradient updates for other training examples. Simply increasing the learning rate is inadequate, as it would cause the model to take overly large steps towards $\mathbf{z}'$, potentially causing optimization to diverge.

Solution: the meritocratic update rule

To solve this problem, we want the upweighting to be more "meritocratic": any program that obtains reward should be upweighted roughly equally.

We first observe that J_{MML} already improves over J_{RL} in this regard. From (6), we see that the gradient weight $q_{\text{MML}}(\mathbf{z})$ is the policy distribution restricted to and renormalized over only reward-earning programs. This renormalization makes the gradient weight uniform *across examples*: even if all reward-earning programs for a particular example have very low model probability, their combined gradient weight $\sum_{\mathbf{z}} q_{\text{MML}}(\mathbf{z})$ is always 1. In our experiments, J_{MML} performs significantly better than J_{RL} (Table 4).

However, while J_{MML} assigns uniform weight across examples, it is still not uniform over the programs *within each example*. Hence we propose a new update rule which goes one step further in pursuing uniform updates. Extending $q_{\text{MML}}(\mathbf{z})$, we define a β-smoothed version:

$$q_\beta(\mathbf{z}) = \frac{q_{\text{MML}}(\mathbf{z})^\beta}{\sum_{\tilde{\mathbf{z}}} q_{\text{MML}}(\tilde{\mathbf{z}})^\beta}. \quad (11)$$

When $\beta = 0$, our weighting is completely uniform across all reward-earning programs within an example while $\beta = 1$ recovers the original MML weighting. Our new update rule is to simply take a modified gradient step where $q = q_\beta$.[4] We will refer to this as the *β-meritocratic* update rule.

5.3 Summary of the proposed approach

We described two problems[5] and their solutions: we reduce exploration bias using *ϵ-greedy randomized beam search* and perform more balanced optimization using the *β-meritocratic parameter update rule*. We call our resulting approach RANDOMER. Table 1 summarizes how RANDOMER combines desirable qualities from both REINFORCE and BS-MML.

6 Experiments

Evaluation. We evaluate our proposed methods on all three domains of the SCONE dataset. Accuracy is defined as the percentage of test examples where the model produces the correct final world state w_M. All test examples have $M = 5$ (5utts), but we also report accuracy after processing the first 3 utterances (3utts). To control for the effects of randomness, we train 5 instances of each model with different random seeds. We report the median accuracy of the instances unless otherwise noted.

Training. Following Long et al. (2016), we decompose each training example into smaller examples. Given an example with 5 utterances, $\mathbf{u} = [u_1, \ldots, u_5]$, we consider all length-1 and length-2 substrings of $\mathbf{u}$: $[u_1], [u_2], \ldots, [u_3, u_4], [u_4, u_5]$ (9 total). We form a new training example from each substring, e.g., $(\mathbf{u}', w_0', w_M')$ where $\mathbf{u}' = [u_4, u_5]$, $w_0' = w_3$ and $w_M' = w_5$.

All models are implemented in TensorFlow (Abadi et al., 2015). Model parameters are randomly initialized (Glorot and Bengio, 2010), with no pre-training. We use the Adam optimizer (Kingma and Ba, 2014) (which is applied to the gradient in (6)), a learning rate of 0.001, a mini-batch size of 8 examples (different from the beam size), and train until accuracy on the validation set converges (on average about 13,000 steps). We

[4] Also, note that if exploration were exhaustive, $\beta = 0$ would be equivalent to supervised learning using the set of all reward-earning programs as targets.

[5] These problems concern the gradient w.r.t. a single example. The full gradient averages over multiple examples, which helps separate correct from spurious. E.g., if multiple examples all mention "yellow hat", we will find a correct program parsing this as `hasHat(yellow)` for each example, whereas the spurious programs we find will follow no consistent pattern. Consequently, spurious gradient contributions may cancel out while correct program gradients will all "vote" in the same direction.

Method	Approximation of $E_q[\cdot]$	Exploration strategy	Gradient weight $q(\mathbf{z})$
REINFORCE	Monte Carlo integration	independent sampling	$p_\theta(\mathbf{z} \mid x)$
BS-MML	numerical integration	beam search	$p_\theta(\mathbf{z} \mid x, R(\mathbf{z}) \neq 0)$
RANDOMER	numerical integration	randomized beam search	$q_\beta(\mathbf{z})$

Table 1: RANDOMER combines qualities of both REINFORCE (RL) and BS-MML. For approximating the expectation over q in the gradient, we use numerical integration as in BS-MML. Our exploration strategy is a hybrid of search (MML) and off-policy sampling (RL). Our gradient weighting is equivalent to MML when $\beta = 1$ and more "meritocratic" than both MML and REINFORCE for lower values of β.

use fixed GloVe vectors (Pennington et al., 2014) to embed the words in each utterance.

Hyperparameters. For all models, we performed a grid search over hyperparameters to maximize accuracy on the validation set. Hyperparameters include the learning rate, the baseline in REINFORCE, ϵ-greediness and β-meritocraticness. For REINFORCE, we also experimented with a regression-estimated baseline (Ranzato et al., 2015), but found it to perform worse than a constant baseline.

6.1 Main results

Comparison to prior work. Table 2 compares RANDOMER to results from Long et al. (2016) as well as two baselines, REINFORCE and BS-MML (using the same neural model but different learning algorithms). Our approach achieves new state-of-the-art results by a significant margin, especially on the SCENE domain, which features the most complex program syntax. We report the results for REINFORCE, BS-MML, and RANDOMER on the seed and hyperparameters that achieve the best validation accuracy.

We note that REINFORCE performs very well on TANGRAMS but worse on ALCHEMY and very poorly on SCENE. This might be because the program syntax for TANGRAMS is simpler than the other two: there is no other way to refer to objects except by index.

We also found that REINFORCE required ϵ-greedy exploration to make any progress. Using ϵ-greedy greatly skews the Monte Carlo approximation of ∇J_{RL}, making it more uniformly weighted over programs in a similar spirit to using β-meritocratic gradient weights q_β. However, q_β increases uniformity over reward-earning programs only, rather than over all programs.

Effect of randomized beam search. Table 3 shows that ϵ-greedy randomized beam search consistently outperforms classic beam search. Even when we increase the beam size of classic beam search to 128, it still does not surpass randomized beam search with a beam of 32, and further increases yield no additional improvement.

	ALCHEMY		TANGRAMS		SCENE	
system	3utts	5utts	3utts	5utts	3utts	5utts
LONG+16	56.8	52.3	64.9	27.6	23.2	14.7
REINFORCE	58.3	44.6	**68.5**	**37.3**	47.8	33.9
BS-MML	58.7	47.3	62.6	32.2	53.5	32.5
RANDOMER	**66.9**	**52.9**	65.8	37.1	**64.8**	**46.2**

Table 2: **Comparison to prior work.** LONG+16 results are directly from Long et al. (2016). Hyperparameters are chosen by best performance on validation set (see Appendix A).

		ALCHEMY		TANGRAMS		SCENE	
random	beam	3utts	5utts	3utts	5utts	3utts	5utts
classic beam search							
None	32	30.3	23.2	0.0	0.0	33.4	20.1
None	128	59.0	46.4	60.9	28.6	24.5	13.9
randomized beam search							
$\epsilon = 0.05$	32	58.7	45.5	61.1	32.5	33.4	23.0
$\epsilon = 0.15$	32	**61.3**	48.3	**65.2**	**34.3**	50.8	33.5
$\epsilon = 0.25$	32	60.5	**48.6**	60.0	27.3	**54.1**	**35.7**

Table 3: **Randomized beam search.** All listed models use gradient weight q_{MML} and TOKENS to represent execution history.

Effect of β-meritocratic updates. Table 4 evaluates the impact of β-meritocratic parameter updates (gradient weight q_β). More uniform upweighting across reward-earning programs leads to higher accuracy and fewer spurious programs, especially in SCENE. However, no single value of β performs best over all domains.

Choosing the right value of β in RANDOMER significantly accelerates training. Figure 3 illustrates that while $\beta = 0$ and $\beta = 1$ ultimately achieve similar accuracy on ALCHEMY, $\beta = 0$ reaches good performance in half the time.

Since lowering β reduces trust in the model policy, $\beta < 1$ helps in early training when the current policy is untrustworthy. However, as it grows more trustworthy, $\beta < 1$ begins to pay a price for ignoring it. Hence, it may be worthwhile to anneal β towards 1 over time.

$q(\mathbf{z})$	ALCHEMY 3utts	ALCHEMY 5utts	TANGRAMS 3utts	TANGRAMS 5utts	SCENE 3utts	SCENE 5utts
q_{RL}	0.2	0.0	0.9	0.6	0.0	0.0
q_{MML} $(q_{\beta=1})$	61.3	48.3	**65.2**	**34.3**	50.8	33.5
$q_{\beta=0.25}$	**64.4**	**48.9**	60.6	29.0	42.4	29.7
$q_{\beta=0}$	63.6	46.3	54.0	23.5	**61.0**	**42.4**

Table 4: **β-meritocratic updates.** All listed models use randomized beam search, $\epsilon = 0.15$ and TOKENS to represent execution history.

	ALCHEMY 3utts	ALCHEMY 5utts	TANGRAMS 3utts	TANGRAMS 5utts	SCENE 3utts	SCENE 5utts
HISTORY	61.3	48.3	**65.2**	**34.3**	50.8	33.5
STACK	**64.2**	**53.2**	63.0	32.4	**59.5**	**43.1**

Table 5: **TOKENS vs STACK embedding.** Both models use $\epsilon = 0.15$ and gradient weight q_{MML}.

Effect of execution history embedding. Table 5 compares our two proposals for embedding the execution history: TOKENS and STACK. STACK performs better in the two domains where an object can be referenced in multiple ways (SCENE and ALCHEMY). STACK directly embeds objects on the stack, invariant to the way in which they were pushed onto the stack, unlike TOKENS. We hypothesize that this invariance increases robustness to spurious behavior: if a program accidentally pushes the right object onto the stack via spurious means, the model can still learn the remaining steps of the program without conditioning on a spurious history.

Fitting vs overfitting the training data. Table 6 reveals that BS-MML and RANDOMER use different strategies to fit the training data. On the depicted training example, BS-MML actually achieves higher expected reward / marginal probability than RANDOMER, but it does so by putting most of its probability on a spurious program—a form of overfitting. In contrast, RANDOMER spreads probability mass over multiple reward-earning programs, including the correct ones.

As a consequence of overfitting, we observed at test time that BS-MML only references people by positional indices instead of by shirt or hat color, whereas RANDOMER successfully learns to use multiple reference strategies.

7 Related work and discussion

Semantic parsing from indirect supervision. Our work is motivated by the classic problem of learning semantic parsers from indirect supervision (Clarke et al., 2010; Liang et al., 2011; Artzi and Zettlemoyer, 2011, 2013; Reddy et al., 2014; Pasupat and Liang, 2015). We are interested in the initial stages of training from scratch, where getting any training signal is difficult due to the combinatorially large search space. We also highlighted the problem of spurious programs which capture reward but give incorrect generalizations.

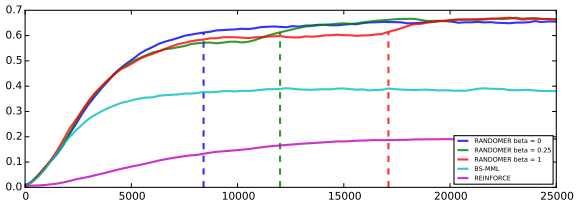


Figure 3: Validation set accuracy (y-axis) across training iterations (x-axis) on ALCHEMY. We compare RANDOMER, BS-MML and REINFORCE. Vertical lines mark the first time each model surpasses 60% accuracy. RANDOMER with $\beta = 0$ reaches this point twice as fast as $\beta = 1$. REINFORCE plateaus for a long time, then begins to climb after 40k iterations (not shown). Training runs are averaged over 5 seeds.

Maximum marginal likelihood with beam search (BS-MML) is traditionally used to learn semantic parsers from indirect supervision.

Reinforcement learning. Concurrently, there has been a recent surge of interest in reinforcement learning, along with the wide application of the classic REINFORCE algorithm (Williams, 1992)—to troubleshooting (Branavan et al., 2009), dialog generation (Li et al., 2016), game playing (Narasimhan et al., 2015), coreference resolution (Clark and Manning, 2016), machine translation (Norouzi et al., 2016), and even semantic parsing (Liang et al., 2017). Indeed, the challenge of training semantic parsers from indirect supervision is perhaps better captured by the notion of sparse rewards in reinforcement learning.

The RL answer would be better exploration, which can take many forms including simple action-dithering such as ϵ-greedy, entropy regularization (Williams and Peng, 1991), Monte Carlo tree search (Coulom, 2006), randomized value functions (Osband et al., 2014, 2016), and methods which prioritize learning environment dynamics (Duff, 2002) or under-explored states (Kearns and Singh, 2002; Bellemare et al., 2016; Nachum et al., 2016). The majority of these methods employ Monte Carlo sampling for exploration. In

Utterance: the man in the purple shirt and red hat moves just to the right of the man in the red shirt and yellow hat

	program	prob
	RANDOMER ($\epsilon = 0.15, \beta = 0$)	
*	`move(hasHat(red), rightOf(hasHat(red)))`	0.122
*	`move(hasShirt(purple), rightOf(hasShirt(red)))`	0.061
o	`move(hasHat(red), rightOf(index(allPeople, 1)))`	0.059
*	`move(hasHat(red), rightOf(hasHat(yellow)))`	0.019
o	`move(index(allPeople, 2), rightOf(hasShirt(red)))`	0.018
x	`move(hasHat(red), 8)`	0.018
	BS-MML	
o	`move(index(allPeople, 2), 2)`	0.887
x	`move(index(allPeople, 2), 6)`	0.041
x	`move(index(allPeople, 2), 5)`	0.020
x	`move(index(allPeople, 2), 8)`	0.016
x	`move(index(allPeople, 2), 7)`	0.009
x	`move(index(allPeople, 2), 3)`	0.008

Table 6: Top-scoring predictions for a *training* example from SCENE (* = correct, o = spurious, x = incorrect). RANDOMER distributes probability mass over numerous reward-earning programs (including the correct ones), while classic beam search MML overfits to one spurious program, giving it very high probability.

contrast, we find randomized beam search to be more suitable in our setting, because it explores low-probability states even when the policy distribution is peaky. Our β-meritocratic update also depends on the fact that beam search returns an *entire set* of reward-earning programs rather than one, since it renormalizes over the reward-earning set. While similar to entropy regularization, β-meritocratic update is more targeted as it only increases uniformity of the gradient among reward-earning programs, rather than across all programs.

Our strategy of using randomized beam search and meritocratic updates lies closer to MML than RL, but this does not imply that RL has nothing to offer in our setting. With the simple connection between RL and MML we established, much of the literature on exploration and variance reduction in RL can be directly applied to MML problems. Of special interest are methods which incorporate a value function such as actor-critic.

Maximum likelihood and RL. It is tempting to group our approach with sequence learning methods which interpolate between supervised learning and reinforcement learning (Ranzato et al., 2015; Venkatraman et al., 2015; Ross et al., 2011; Norouzi et al., 2016; Bengio et al., 2015; Levine, 2014). These methods generally seek to make RL training easier by pre-training or "warm-starting" with fully supervised learning. This requires each training example to be labeled with a reasonably correct output sequence. In our setting, this would amount to labeling each example with the correct program, which is not known. Hence, these methods cannot be directly applied.

Without access to correct output sequences, we cannot directly maximize likelihood, and instead resort to maximizing the *marginal* likelihood (MML). Rather than proposing MML as a form of pre-training, we argue that MML is a superior substitute for the standard RL objective, and that the β-meritocratic update is even better.

Simulated annealing. Our β-meritocratic update employs exponential smoothing, which bears resemblance to the simulated annealing strategy of Och (2003); Smith and Eisner (2006); Shen et al. (2015). However, a key difference is that these methods smooth the objective function whereas we smooth an expectation in the gradient. To underscore the difference, we note that fixing $\beta = 0$ in our method (total smoothing) is quite effective, whereas total smoothing in the simulated annealing methods would correspond to a completely flat objective function, and an uninformative gradient of zero everywhere.

Neural semantic parsing. There has been recent interest in using recurrent neural networks for semantic parsing, both for modeling logical forms (Dong and Lapata, 2016; Jia and Liang, 2016; Liang et al., 2017) and for end-to-end execution (Yin et al., 2015; Neelakantan et al., 2016). We develop a neural model for the context-dependent setting, which is made possible by a new stack-based language similar to Riedel et al. (2016).

Acknowledgments. This work was supported by the NSF Graduate Research Fellowship under No. DGE-114747 and the NSF CAREER Award under No. IIS-1552635.

Reproducibility. Our code is made available at https://github.com/kelvinguu/lang2program. Reproducible experiments are available at https://worksheets.codalab.org/worksheets/0x88c914ee1d4b4a4587a07f36f090f3e5/.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean,

M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. 2015. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Y. Artzi and L. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 421–432.

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

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NIPS)*. pages 1471–1479.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. pages 1171–1179.

S. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. pages 82–90.

K. Clark and C. D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667* .

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

R. Coulom. 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International Conference on Computers and Games*. pages 72–83.

A. P. Dempster, L. N. M., and R. D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B* 39(1):1–38.

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

M. O. Duff. 2002. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. Ph.D. thesis, University of Massachusetts Amherst.

X. Glorot and Y. Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

M. Kearns and S. Singh. 2002. Near-optimal reinforcement learning in polynomial time. *Machine Learning* 49(2):209–232.

D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

J. Krishnamurthy and T. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*. pages 754–765.

S. Levine. 2014. *Motor Skill Learning with Local Trajectory Methods*. Ph.D. thesis, Stanford University.

J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. 2016. Deep reinforcement learning for dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

C. Liang, J. Berant, Q. Le, and K. D. F. N. Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Association for Computational Linguistics (ACL)*.

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

R. Long, P. Pasupat, and P. Liang. 2016. Simpler context-dependent logical forms via model projections. In *Association for Computational Linguistics (ACL)*.

O. Nachum, M. Norouzi, and D. Schuurmans. 2016. Improving policy gradient by exploring under-appreciated rewards. *arXiv preprint arXiv:1611.09321* .

K. Narasimhan, T. Kulkarni, and R. Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941* .

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

M. Norouzi, S. Bengio, N. Jaitly, M. Schuster, Y. Wu, D. Schuurmans, et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*. pages 1723–1731.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Association for Computational Linguistics (ACL)*. pages 160–167.

I. Osband, C. Blundell, A. Pritzel, and B. V. Roy. 2016. Deep exploration via bootstrapped DQN. In *Advances In Neural Information Processing Systems*. pages 4026–4034.

I. Osband, B. V. Roy, and Z. Wen. 2014. Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635* .

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

P. Pasupat and P. Liang. 2016. Inferring logical forms from denotations. In *Association for Computational Linguistics (ACL)*.

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

S. Reddy, M. Lapata, and M. Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics (TACL)* 2(10):377–392.

S. Riedel, M. Bosnjak, and T. Rocktäschel. 2016. Programming with a differentiable forth interpreter. *CoRR, abs/1605.06640* .

S. Ross, G. Gordon, and A. Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Artificial Intelligence and Statistics (AISTATS)*.

S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433* .

D. A. Smith and J. Eisner. 2006. Minimum risk annealing for training log-linear models. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*. pages 787–794.

R. Sutton, D. McAllester, S. Singh, and Y. Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*.

A. Venkatraman, M. Hebert, and J. A. Bagnell. 2015. Improving multi-step prediction of learned time series models. In *Association for the Advancement of Artificial Intelligence (AAAI)*. pages 3024–3030.

R. J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3):229–256.

R. J. Williams and J. Peng. 1991. Function optimization using connectionist reinforcement learning algorithms. *Connection Science* 3(3):241–268.

P. Yin, Z. Lu, H. Li, and B. Kao. 2015. Neural enquirer: Learning to query tables. *arXiv preprint arXiv:1512.00965* .

A Hyperparameters in Table 2

System	ALCHEMY	TANGRAMS	SCENE
REINFORCE	Sample size 32 Baseline 10^{-2} $\epsilon = 0.15$ embed TOKENS	Sample size 32 Baseline 10^{-2} $\epsilon = 0.15$ embed TOKENS	Sample size 32 Baseline 10^{-4} $\epsilon = 0.15$ embed TOKENS
BS-MML	Beam size 128 embed TOKENS	Beam size 128 embed TOKENS	Beam size 128 embed TOKENS
RANDOMER	$\beta = 1$ $\epsilon = 0.05$ embed TOKENS	$\beta = 1$ $\epsilon = 0.15$ embed TOKENS	$\beta = 0$ $\epsilon = 0.15$ embed STACK

B SCONE domains and program tokens

token	type	semantics
Shared across ALCHEMY, TANGRAMS, SCENE		
`1`, `2`, `3`, ... `-1`, `-2`, `-3`, ...	constant	**push:** number
`red`, `yellow`, `green`, `orange`, `purple`, `brown`	constant	**push:** color
`allObjects`	constant	**push:** the list of all objects
`index`	function	**pop:** a list L and a number i **push:** the object $L[i]$ (the index starts from 1; negative indices are allowed)
`prevArg`j ($j = 1, 2$)	function	**pop:** a number i **push:** the j argument from the ith action
`prevAction`	action	**pop:** a number i **perform:** fetch the ith action and execute it using the arguments on the stack
Additional tokens for the ALCHEMY domain An ALCHEMY world contains 7 beakers. Each beaker may contain up to 4 units of colored chemical.		
`1/1`	constant	**push:** fraction (used in the `drain` action)
`hasColor`	function	**pop:** a color c **push:** list of beakers with chemical color c
`drain`	action	**pop:** a beaker b and a number or fraction a **perform:** remove a units of chemical (or all chemical if $a =$ `1/1`) from b
`pour`	action	**pop:** two beakers b_1 and b_2 **perform:** transfer all chemical from b_1 to b_2
`mix`	action	**pop:** a beaker b **perform:** turn the color of the chemical in b to `brown`
Additional tokens for the TANGRAMS domain A TANGRAMS world contains a row of tangram pieces with different shapes. The shapes are anonymized; a tangram can be referred to by an index or a history reference, but not by shape.		
`swap`	action	**pop:** two tangrams t_1 and t_2 **perform:** exchange the positions of t_1 and t_2
`remove`	action	**pop:** a tangram t **perform:** remove t from the stage
`add`	action	**pop:** a number i and a previously removed tangram t **perform:** insert t to position i
Additional tokens for the SCENE domain A SCENE world is a linear stage with 10 positions. Each position may be occupied by a person with a colored shirt and optionally a colored hat. There are usually 1-5 people on the stage.		
`noHat`	constant	**push:** pseudo-color (indicating that the person is not wearing a hat)
`hasShirt`, `hasHat`	function	**pop:** a color c **push:** the list of all people with shirt or hat color c
`hasShirtHat`	function	**pop:** two colors c_1 and c_2 **push:** the list of all people with shirt color c_1 and hat color c_2
`leftOf`, `rightOf`	function	**pop:** a person p **push:** the location index left or right of p
`create`	action	**pop:** a number i and two colors c_1, c_2 **perform:** add a new person at position i with shirt color c_1 and hat color c_2
`move`	action	**pop:** a person p and a number i **perform:** move p to position i
`swapHats`	action	**pop:** two people p_1 and p_2 **perform:** have p_1 and p_2 exchange their hats
`leave`	action	**pop:** a person p **perform:** remove p from the stage

Diversity driven attention model for query-based abstractive summarization

Preksha Nema[†] **Mitesh M. Khapra**[†] **Anirban Laha**[*†] **Balaraman Ravindran**[†]
[†]Indian Institute of Technology Madras, India
[*] IBM Research India
{preksha,miteshk}@cse.iitm.ac.in
anirlaha@in.ibm.com ravi@cse.iitm.ac.in

Abstract

Abstractive summarization aims to generate a shorter version of the document covering all the salient points in a compact and coherent fashion. On the other hand, query-based summarization highlights those points that are relevant in the context of a given query. The encode-attend-decode paradigm has achieved notable success in machine translation, extractive summarization, dialog systems, etc. But it suffers from the drawback of generation of repeated phrases. In this work we propose a model for the query-based summarization task based on the encode-attend-decode paradigm with two key additions (i) a query attention model (in addition to document attention model) which learns to focus on different portions of the query at different time steps (instead of using a static representation for the query) and (ii) a new diversity based attention model which aims to alleviate the problem of repeating phrases in the summary. In order to enable the testing of this model we introduce a new query-based summarization dataset building on debatepedia. Our experiments show that with these two additions the proposed model clearly outperforms vanilla encode-attend-decode models with a gain of 28% (absolute) in ROUGE-L scores.

1 Introduction

Over the past few years neural models based on the encode-attend-decode (Bahdanau et al., 2014) paradigm have shown great success in various natural language generation (NLG) tasks such as machine translation (Bahdanau et al., 2014), abstractive summarization ((Rush et al., 2015),(Nallapati et al., 2016)) dialog (Li et al., 2016), *etc*. One such NLG problem which has not received enough attention in the past is query based abstractive text summarization where the aim is to generate the summary of a document in the context of a query. In general, abstractive summarization, aims to cover all the salient points of a document in a compact and coherent fashion. On the other hand, query focused summarization highlights those points that are relevant in the context of the query. Thus given a document on "the super bowl", the query "How was the half-time show?", would result in a summary that would not cover the actual game itself.

Note that there has been some work on query based extractive summarization in the past where the aim is to simply extract the most salient sentence(s) from a document and treat these as a summary. There is no natural language generation involved. Since, we were interested in abstractive (as opposed to extractive) summarization we created a new dataset based on debatepedia. This dataset contains triplets of the form (query, document, summary). Further, each summary is abstractive and not extractive in the sense that the summary does not necessarily comprise of a sentence which is simply copied from the original document.

Using this dataset as a testbed, we focus on a recurring problem in models based on the encode-attend-decode paradigm. Specifically, it is observed that the summaries produced by such models contain repeated phrases. Table 1 shows a few such examples of summaries gener-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1063–1072
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1098

Document Snippet: The "natural death" alternative to euthanasia is not keeping someone alive via life support until they die on life support. That would, indeed, be unnatural. The natural alternative is, instead, to allow them to die off of life support. **Query**: Is euthanasia better than withdrawing life support (non-treatment)? **Ground Truth Summary**: The alternative to euthanasia is a natural death without life support. **Predicted Summary**: the large to euthanasia is a natural death **life life** use
Document Snippet: Legalizing same-sex marriage would also be a recognition of basic American principles, and would represent the culmination of our nation's commitment to equal rights. It is, some have said, the last major civil-rights milestone yet to be surpassed in our two-century struggle to attain the goals we set for this nation at its formation. **Query:** Is gay marriage a civil right? **Ground Truth Summary**: Gay marriage is a fundamental equal right. **Predicted Summary**: gay marriage is a appropriate **right right**

Table 1: Examples showing repeated words in the output of encoder-decoder models

ated by such a model when trained on this new dataset. This problem has also been reported by (Chen et al., 2016) in the context of summarization and by (Sankaran et al., 2016) in the context of machine translation.

We first provide an intuitive explanation for this problem and then propose a solution for alleviating it. A typical encode-attend-decode model first computes a vectorial representation for the document and the query and then produces a contextual summary one word at a time. Each word is produced by feeding a new context vector to the decoder at each time step by attending to different parts of the document and query. If the decoder produces the same word or phrase repeatedly then it could mean that the context vectors fed to the decoder at these time steps are very similar.

We propose a model which explicitly prevents this by ensuring that successive context vectors are orthogonal to each other. Specifically, we subtract out any component that the current context vector has in the direction of the previous context vector. Notice that, we do not require the current context vector to be orthogonal to all previous context vectors but just its immediate predecessor. This enables the model to attend to words repeatedly if required later in the process. To account for the complete history (or all previous context vectors) we also propose an extension of this idea where we pass the sequence of context vectors through a LSTM (Hochreiter and Schmidhuber, 1997) and ensure that the current state produced by the LSTM is orthogonal to the history. At each time step, the state of the LSTM is then fed to the decoder to produce one word in the summary.

Our contributions can be summarized as follows: (i) We propose a new dataset for query based abstractive summarization and evaluate encode-attend-decode models on this dataset (ii) We study the problem of repeating phrases in NLG in the context of this dataset and propose two solutions for countering this problem. We show that our method outperforms a vanilla encoder-decoder model with a gain of 28% (absolute) in ROUGE-L score (iii) We also demonstrate that our method clearly outperforms a recent state of the art method proposed for handling the problem of repeating phrases with a gain of 7% (absolute) in ROUGE-L scores (iv) We do a qualitative analysis of the results and show that our model indeed produces outputs with fewer repetitions.

2 Related Work

Summarization has been studied in the context of text ((Mani, 2001), (Das and Martins, 2007), (Nenkova and McKeown, 2012)) as well as speech ((Zhu and Penn, 2006), (Zhu et al., 2009)). A vast majority of this work has focused on extractive summarization where the idea is to construct a summary by selecting the most relevant sentences from the document ((Neto et al., 2002), (Erkan and Radev, 2004), (Filippova and Altun, 2013), (Colmenares et al., 2015), (Riedhammer et al., 2010), (Ribeiro et al., 2013)). There has been some work on abstractive summarization in the context of DUC-2003 and DUC-2004 contests (Zajic et al.). We refer the reader to (Das and Martins, 2007) and (Nenkova and McKeown, 2012) for an excellent survey of

the field.

Recent research in abstractive summarization has focused on data driven neural models based on the encode-attend-decode paradigm (Bahdanau et al., 2014). For example, (Rush et al., 2015), report state of the art results on the GigaWord and DUC corpus using such a model. Similarly, the work of Lopyrev (2015) uses neural networks to generate news headline from short news stories. Chopra et al. (2016) extend the work of Rush et al. (2015) and report further improvements on the two datasets. Hu et al. (2015) introduced a dataset for Chinese short text summarization and evaluated a similar RNN encoder-decoder model on it.

One recurring problem in encoder-decoder models for NLG is that they often repeat the same phrase/word multiple times in the summary (at the cost of both coherency and fluency). Sankaran et al. (2016) study this problem in the context of MT and propose a temporal attention model which enforces the attention weights for successive time steps to be different from each other. Similarly, and more relevant to this work, Chen et al. (2016) propose a distraction based attention model which maintains a history of attention vectors and context vectors. It then subtracts this history from the current attention and context vector. When evaluated on our dataset their method performs poorly. This could be because their method is very aggressive in dealing with the history (as explained later in the Experiments section). On the other hand, our method has a better way of handling history (by passing context vectors through an LSTM recurrent network) which gives us the flexibility to forget/retain some portions of the history and at the same time produce diverse context vectors at successive time steps.

We evaluate our method in the context of query based abstractive summarization - a problem which has received almost no attention in the past due to unavailability of datasets. We create a new dataset for this task and show that our method indeed produces better output by reducing the number of repeated phrases produced by encoder decoder models.

Average number of words per		
Document	Summary	Query
66.4	11.16	9.97

Table 2: Average length of documents/queries/summaries in the dataset

3 Dataset

As mentioned earlier, there are no existing datasets for query based abstractive summarization. We create such a dataset from Debatepedia an encyclopedia of pro and con arguments and quotes on critical debate topics. There are 663 debates in the corpus (we have considered only those debates which have at least one query with one document). These 663 debates belong to 53 overlapping categories such as Politics, Law, Crime, Environment, Health, Morality, Religion, etc. A given topic can belong to more than one category. For example, the topic "Eye for an Eye philosophy" belongs to both "Law" as well as "Morality". The average number of queries per debate is 5 and the average number of documents per query is 4. Please refer to the dataset url[1] for more details about number of debates per category.

For example, Figure 1 shows the queries associated with the topic "Algae Biofuel". It also lists the set of documents and an abstractive summary associated with each query. As is obvious from the example, the summary is an abstractive summary and not extracted directly from the document. We crawled 12695 such {query, document, summary} triples from debatepedia (these were all the triples that were available). Table 2 reports the average length of the query, summary and documents in this dataset.

We used 10 fold cross validation for all our experiments. Each fold uses 80% of the documents for training, 10% for validation and 10% for testing.

4 Proposed model

Given a query $\mathbf{q} = q_1, q_2, ..., q_k$ containing k words, a document $\mathbf{d} = d_1, d_2, ..., d_n$ containing n words, the task is to generate a contextual summary $\mathbf{y} = y_1, y_2, ..., y_m$ containing

[1] http://www.cse.iitm.ac.in/~miteshk/datasets/qbas.html

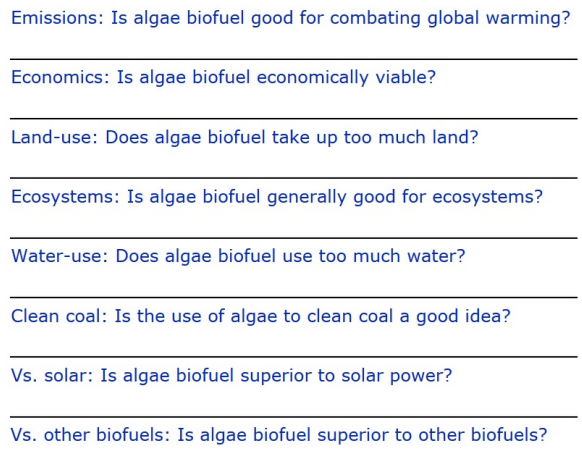


Figure 1: Queries associated with the topic "algae biofuel"

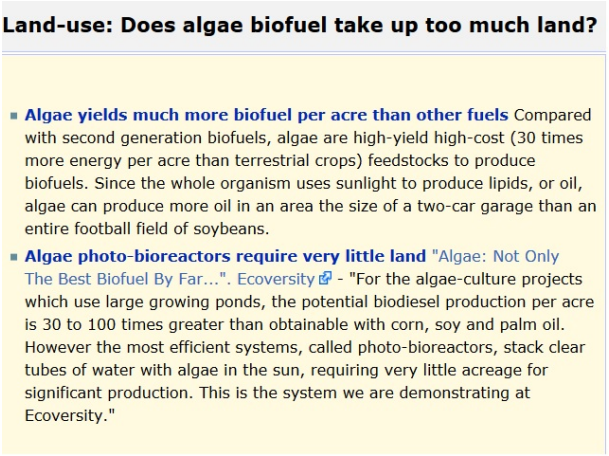


Figure 2: Documents and summaries for a given query

m words. This can be modeled as the problem of finding a $\mathbf{y}^*$ that maximizes the probability $p(\mathbf{y}|\mathbf{q}, \mathbf{d})$ which can be further decomposed as:

$$y^* = \arg\max_y \prod_{t=1}^{m} p(y_t|y_1, ..., y_{t-1}, \mathbf{q}, \mathbf{d}) \quad (1)$$

We now describe a way of modeling $p(y_t|y_1, ..., y_{t-1}, \mathbf{q}, \mathbf{d})$ using the neural encoder-attention-decoder paradigm. The proposed model contains the following components: (i) an encoder RNN for the query (ii) an encoder RNN for the document (iii) attention mechanism for the query (iv) attention mechanism for the document and (v) a decoder RNN. All the RNNs use a GRU cell.

Encoder for the query: We use a recurrent neural network with Gated Recurrent Units (GRU) for encoding the query. It reads the query $\mathbf{q} = q_1, q_2, ..., q_k$ from left to right and computes a hidden representation for each time-step as:

$$h_i^q = \text{GRU}_q(h_{i-1}^q, e(q_i)) \quad (2)$$

where $e(q_i) \in \mathbb{R}^d$ is the d-dimensional embedding of the query word q_i.

Encoder for the document: This is similar to the query encoder and reads the document $\mathbf{d} = d_1, d_2, ..., d_n$ from left to right and computes a hidden representation for each time-step as:

$$h_i^d = \text{GRU}_d(h_{i-1}^d, e(d_i)) \quad (3)$$

where $e(d_i) \in \mathbb{R}^d$ is the d-dimensional embedding of the document word d_i.

Attention mechanism for the query : At each time step, the decoder produces an output word by focusing on different portions of the query (document) with the help of a query (document) attention model. We first describe the query attention model which assigns weights $\alpha_{t,i}^q$ to each word in the query at each decoder timestep using the following equations.

$$a_{t,i}^q = v_q^T \tanh(W_q s_t + U_q h_i^q) \quad (4)$$

$$\alpha_{t,i}^q = \frac{\exp(a_{t,i}^q)}{\sum_{j=1}^{k} \exp(a_{t,j}^q)} \quad (5)$$

where s_t is the current state of the decoder at time step t (we will see an exact formula for this soon). $W_q \in \mathbb{R}^{l_2 \times l_1}$, $U_q \in \mathbb{R}^{l_2 \times l_2}$, $v_q \in \mathbb{R}^{l_2}$, l_1 is the size of the decoder's hidden state, l_2 is both the size of h_i^q and also the size of the final query representation at time step t, which is computed as:

$$q_t = \sum_{i=1}^{k} \alpha_{t,i}^q h_i^q \quad (6)$$

Attention mechanism for the document : We now describe the document attention model which assigns weights to each word in the document using the following equations.

$$a_{t,i}^d = v_d^T \tanh(W_d s_t + U_d h_i^d + Z q_t) \quad (7)$$

$$\alpha_{t,i}^d = \frac{\exp(a_{t,i}^d)}{\sum_{j=1}^{n} \exp(a_{t,j}^d)}$$

where s_t is the current state of the decoder at time step t (we will see an exact formula for this

soon). $W_d \in \mathbb{R}^{l_4 \times l_1}$, $U_d \in \mathbb{R}^{l_4 \times l_4}$, $Z \in \mathbb{R}^{l_4 \times l_2}$, $v_d \in \mathbb{R}^{l_2}$, l_4 is the size of h_i^d and also the size of the final document representation d_t which is passed to the decoder at time step t as:

$$d_t = \sum_{i=1}^{n} \alpha_{t,i}^d h_i^d \tag{8}$$

Note that d_t now encodes the relevant information from the document as well as the query (see Equation (7)) at time step t. We refer to this as the *context vector* for the decoder.

Decoder: The hidden state of the decoder s_t at each time t is again computed using a GRU as follows:

$$s_t = \text{GRU}_{dec}(s_{t-1}, [e(y_{t-1}), d_{t-1}]) \tag{9}$$

where, y_{t-1} gives a distribution over the vocabulary words at timestep $t-1$ and is computed as:

$$y_t = \text{softmax}(W_o f(W_{dec} s_t + V_{dec} d_t)) \tag{10}$$

where $W_o \in \mathbb{R}^{N \times l_1}$, $W_{dec} \in \mathbb{R}^{l_1 \times l_1}$, $V_{dec} \in \mathbb{R}^{l_1 \times l_4}$, N is the vocabulary size, y_t is the final output of the model which defines a probability distribution over the output vocabulary. This is exactly the quantity defined in Equation (1) that we wanted to model $(p(y_t|y_1, ..., y_{t-1}, \mathbf{q}, \mathbf{d}))$. Further, note that, $e(y_{t-1})$ is the d-dimensional embedding of the word which has the highest probability under the distribution y_{t-1}. Also $[e(y_{t-1}), d_{t-1}]$ means a concatenation of the vectors $e(y_{t-1}), d_{t-1}$. We chose f to be the identity function.

The model as described above is an instantiation of the encoder-attention-decoder idea applied to query based abstractive summarization. As mentioned earlier (and demonstrated later through experiments), this model suffers from the problem of repeating the same phrase/word in the output. We now propose a new attention model which we refer to as diversity based attention model to address this problem.

4.1 Diversity based attention model

As hypothesized earlier, if the decoder produces the same phrase/word multiple times then it is possible that the context vectors being fed to the decoder at consecutive time steps are very similar. We propose four models ($\mathbf{D_1}$, $\mathbf{D_2}$, $\mathbf{SD_1}$, $\mathbf{SD_2}$) to directly address this problem.

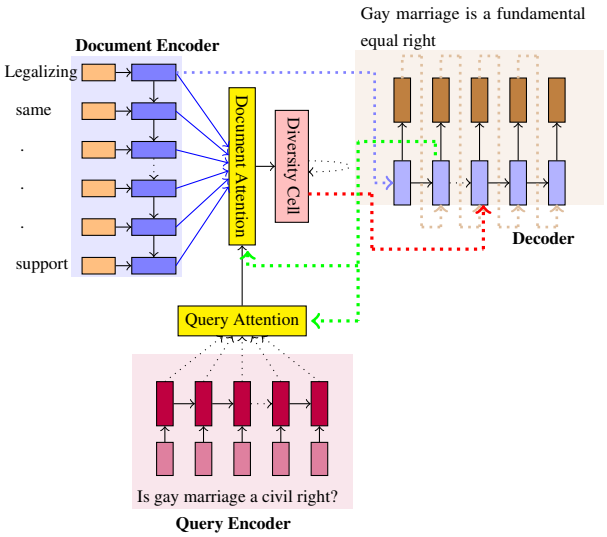


Figure 3: Proposed model for Query based Abstractive Summarization with (i) query encoder (ii) document encoder (iii) query attention model (iv) diversity based document attention model and (v) decoder. The green and red arrows show the connections for timestep 3 of the decoder.

$\mathbf{D_1}$: In this model, after computing d_t as described in Equation (8), we make it orthogonal to the context vector at time $t-1$:

$$d_t' = d_t - \frac{d_t^T d_{t-1}'}{d_{t-1}'^T d_{t-1}'} d_{t-1}' \tag{11}$$

$\mathbf{SD_1}$: The above model imposes a hard orthogonality constraint on the context vector(d_t'). We also propose a relaxed version of the above model which uses a gating parameter. This gating parameter decides what fraction of the previous context vector should be subtracted from the current context vector using the following equations:

$$\gamma_t = W_g d_{t-1} + b_g$$
$$d_t' = d_t - \gamma_t \frac{d_t^T d_{t-1}'}{d_{t-1}'^T d_{t-1}'} d_{t-1}'$$

where $W_g \in \mathbb{R}^{l_4 \times l_4}$, $b_g \in \mathbb{R}^{l_4}$, l_4 is the dimension of d_t as defined in equation (8).

$\mathbf{D_2}$: The above model only ensures that the current context vector is diverse w.r.t the previous context vector. It ignores all history before time step $t-1$. To account for the history, we treat successive context vectors as a sequence and use

a modified LSTM cell to compute the new state at each time step. Specifically, we use the following set of equations to compute a diverse context at time t:

$$
\begin{aligned}
i_t &= \sigma(W_i d_t + U_i h_{t-1} + b_i) \\
f_t &= \sigma(W_f d_t + U_f h_{t-1} + b_f) \\
o_t &= \sigma(W_o d_t + U_o h_{t-1} + b_o) \\
\hat{c}_t &= \tanh(W_c d_t + U_c h_{t-1} + b_c) \\
c_t &= i_t \odot \hat{c}_t + f_t \odot c_{t-1} \\
c_t^{diverse} &= c_t - \frac{{c_t}^T c_{t-1}}{c_{t-1}^T c_{t-1}} c_{t-1} \qquad (12) \\
h_t &= o_t \odot \tanh(c_t^{diverse}) \\
d_t^{'} &= h_t \qquad (13)
\end{aligned}
$$

where $W_i, W_f, W_o, W_c \in \mathbb{R}^{l_5 \times l_4}$, $U_i, U_f, U_o, U_c \in \mathbb{R}^{l_5 \times l_4}$, d_t is the l_4-dimensional output of Equation (8); l_5 is number of hidden units in the LSTM cell. This final $d_t^{'}$ from Equation (13) is then used in Equation (9). Note that Equation (12) ensures that state of the LSTM at time step t is orthogonal to the previous history. Figure 3 shows a pictorial representation of the model with a diversity LSTM cell.

$\mathbf{SD_2}$: This model again uses a relaxed version of the orthogonality constraint used in $\mathbf{D_2}$. Specifically, we define a gating parameter g_t and replace (12) above by (14) as define below:

$$
\begin{aligned}
g_t &= \sigma(W_g d_t + U_g h_{t-1} + b_o) \\
c_t^{diverse} &= c_t - g_t \frac{{c_t}^T c_{t-1}}{c_{t-1}^T c_{t-1}} c_{t-1} \qquad (14)
\end{aligned}
$$

where $W_g \in \mathbb{R}^{l_5 \times l_4}$, $U_g \in \mathbb{R}^{l_5 \times l_4}$

5 Baseline Methods

We compare with two recently proposed baseline diversity methods (Chen et al., 2016) as described below. Note that these methods were proposed in the context of abstractive summarization (not query based abstractive summarization) and we adapt them for the task of query based abstractive summarization. Below we just highlight the key differences from our model in computing the context vector $d_t^{'}$ passed to the decoder.

M1: This model accumulates all the previous context vectors as $\sum_{j=1}^{t-1} d_j^{'}$ and incorporates this history while computing a diverse context vector:

$$
d_t^{'} = \tanh(W_c d_t - U_c \sum_{j=1}^{t-1} d_j^{'}) \qquad (15)
$$

where $W_c, U_c \in \mathbb{R}^{l_4 \times l_4}$ are diagonal matrices. We then use this diversity driven context $d_t^{'}$ in Equation (9) and (10).

M2: In this model, in addition to computing a diverse context as described in Equation (15), the attention weights at each time step are also forced to be diverse from the attention weights at the previous time step.

$$
\alpha_{t,i}^{'} = v_a^T \tanh(W_a s_t^{'} + U_a d_t - b_a \sum_{j=1}^{t-1} \alpha_{j,i}^{'})
$$

where $W_a \in \mathbb{R}^{l_1 \times l_1}$, $U_a \in \mathbb{R}^{l_1 \times l_4}$, $b_a, v_a \in \mathbb{R}^{l_1}$, l_1 is the number of hidden units in the decoder GRU. Once again, they maintain a history of attention weights and compute a diverse attention vector by subtracting the history from the current attention vector.

6 Experimental Setup

We evaluate our models on the dataset described in section 3. Note that there are no prior baselines on query based abstractive summarization so we could only compare with different variations of the encoder decoder models as described above. Further, we compare our diversity based attention models with existing models for diversity by suitably adapting them to this problem as described earlier. Specifically, we compare the performance of the following models:

- **Vanilla e-a-d**: This is the vanilla encoder-attention-decoder model adapted to the problem of abstractive summarization. It contains the following components (i) document encoder (ii) document attention model (iii) decoder. It does not contain an encoder or attention model for the query. This helps us understand the importance of the query.
- $\mathbf{Query}_{enc}$: This model contains the query encoder in addition to the three components used in the vanilla model above. It does not contain any attention model for the query.

- $\mathbf{Query}_{att}$: This model contains the query attention model in addition to all the components in $Query_{enc}$.
- $\mathbf{D}_1$: The diversity attention model as described in Section 4.1.
- $\mathbf{D}_2$: The LSTM based diversity attention model as described in Section 4.1.
- $\mathbf{SD}_1$: The soft diversity attention model as described in Section 4.1
- $\mathbf{SD}_2$: The soft LSTM based diversity attention model as described in Section 4.1
- $\mathbf{B}_1$: Diversity cell in Figure3 is replaced by the basic LSTM cell (i.e. $c_t^{diverse} = c_t$ instead of using Equation (12). This helps us understand whether simply using an LSTM to track the history of context vectors (without imposing a diversity constraint) is sufficient.
- $\mathbf{M}_1$: The baseline model which operates on the context vector as described in Section 5.
- $\mathbf{M}_2$: The baseline model which operates on the attention weights in addition to the context vector as described in Section 5.

We used 80% of the data for training, 10% for validation and 10% for testing. We create 10 such folds and report the average Rouge-1, Rouge-2, Rouge-L scores across the 10 folds. The hyperparameters (batch size and GRU cell sizes) of all the models are tuned on the validation set. We tried the following batch sizes : 32, 64 and the following GRU cell sizes 200, 300, 400. We used Adam (Kingma and Ba, 2014) as the optimization algorithm with the initial learning rate set to 0.0004, $\beta_1 = 0.9$, $\beta_2 = 0.999$. We used pre-trained publicly available Glove word embeddings[2] and fine-tuned them during training. The same word embeddings are used for the query words and the document words.

Table 3 summarizes the results of our experiments.

[2]http://nlp.stanford.edu/projects/glove/

Models	ROUGE-1	ROUGE-2	ROUGE-L
Vanilla e-a-d	13.73	2.06	12.84
$Query_{enc}$	20.87	3.39	19.38
$Query_{att}$	29.28	10.24	28.21
B1	23.18	6.46	22.03
M1	33.06	13.35	32.17
M2	18.42	4.47	17.45
D1	33.85	13.65	32.99
SD1	31.36	11.23	30.5
D2	38.12	16.76	37.31
SD2	**41.26**	**18.75**	**40.43**

Table 3: Performance on various models using full-length ROUGE metrics

7 Discussions

In this section, we discuss the results of the experiments reported in Table 3.

1. Effect of Query: Comparing rows 1 and 2 we observe that adding an encoder for the query and allowing it to influence the outputs of the decoder indeed improves the performance. This is expected as the query contains some keywords which could help in sharpening the focus of the summary.

2. Effect of Query attention model: Comparing rows 2 and 3 we observe that using an attention model to dynamically compute the query representation at each time step improves the results. This suggests that the attention model indeed learns to focus on relevant portions of the query at different time steps.

3. Effect of Diversity models: All the diversity models introduced in the paper (rows 7, 8, 9, 10) give significant improvement over the non-diversity models. In particular, the modified LSTM based diversity model gives the best results. This is indeed very encouraging and Table 4 shows some sample summaries comparing the performance of different models.

4. Comparison with baseline diversity models: The baseline diversity model M1 performs at par with our models D1 and SD1 but not as good as D2 and SD2. However, the model M2 performs very poorly. We believe that simultaneously adding a constraint on the context vectors as well as attention weights (as is indeed the case with M2) is a bit too aggressive and leads to poor performance (although this needs further investigation).

5. Quantitative Analysis: In addition to the qualitative analysis reported in Table 4 we also did a quantitative analysis by counting the num-

Source: Although cannabis does indeed have some harmful effects, it is no more harmful than legal substances like alcohol and tobacco. As a matter of fact, research by the British Medical Association shows that nicotine is far more addictive than cannabis. Furthermore, the consumption of alcohol and the smoking of cigarettes cause more deaths per year than does the use of cannabis (e.g. through lung cancer, stomach ulcers, accidents caused by drunk driving etc.). The legalization of cannabis will remove an anomaly in the law whereby substances that are more dangerous than cannabis are legal whilst the possession and use of cannabis remains unlawful. **Query:** is marijuana harmless enough to be considered a medicine **G:** marijuana is no more harmful than tobacco and alcohol $Query_{attn}$: marijuana is no the **drug drug** for **tobacco and tobacco** **D1:** marijuana is no more harmful than tobacco and tobacco **SD1:** marijuana is more for evidence than tobacco and health **D2:** marijuana is no more harmful than tobacco and use **SD2:** marijuana is no more harmful than tobacco and alcohol
Source: Fuel cell critics point out that hydrogen is flammable, but so is gasoline. Unlike gasoline, which can pool up and burn for a long time, hydrogen dissipates rapidly. Gas tanks tend to be easily punctured, thin-walled containers, while the latest hydrogen tanks are made from Kevlar. Also, gaseous hydrogen isn't the only method of storage under consideration–BMW is looking at liquid storage while other researchers are looking at chemical compound storage, such as boron pellets. **Query:** safety are hydrogen fuel cell vehicles safe **G:** hydrogen in cars is less dangerous than gasoline $Query_{attn}$: hydrogen is **hydrogen hydrogen hydrogen** fuel energy **D1:**hydrogen in cars is less natural than gasoline **SD1:** hydrogen in cars is reduce risk than fuel **D2:** hydrogen in waste is less effective than gasoline **SD2:**hydrogen in cars is less dangerous than gasoline
Source: The basis of all animal rights should be the Golden Rule: we should treat them as we would wish them to treat us, were any other species in our dominant position. **Query:** do animals have rights that makes eating them inappropriate **G:** animals should be treated as we would want to be treated $Query_{att}$: animals should be **treated** as we would protect to be **treated** **D1:** animals should be **treated** as we most individual to be **treated** **SD1:** animals should be **treated** as we would physically to be treated **D2:** animals should be **treated** as we would illegal to be **treated** **SD2:** animals should be **treated** as those would want to be **treated**

Table 4: Summaries generated by different models. In general, we observed that the baseline models which do not use a diversity based attention model tend to produce more repetitions. Notice that the last example shows that our model is not very aggressive in dealing with the history and is able to produce valid repetitions (treated ... treated) when needed

ber of sentences containing repeated words generated by different models. Specifically for the 1268 test instances we counted the number of sentences containing repeated words as generated by different modes. Table 5 summarizes this analysis.

Model	Number
$\mathbf{Query}_{attn}$	498
$\mathbf{SD}_1$	352
$\mathbf{SD}_2$	344
$\mathbf{D}_1$	191
$\mathbf{D}_2$	179

Table 5: Average number of sentences with repeating words across 10 folds

8 Conclusion

In this work we proposed a query-based summarization method. The unique feature of

the model is a novel diversification mechanism based on successive orthogonalization. This gives us the flexibility to: (i) provide diverse context vectors at successive time steps and (ii) pay attention to words repeatedly if need be later in the summary (as opposed to existing models which aggressively delete the history). We also introduced a new data set and empirically verified we perform significantly better (gain of 28% (absolute) in ROUGE-L score) than applying a plain encode-attend-decode mechanism to this problem. We observe that adding an attention mechanism on the query string gives significant improvements. We also compare with a state of the art diversity model and outperform it by a good margin (gain of 7% (absolute) in ROUGE-L score). The diversification model proposed is general enough to apply to other NLG tasks with suitable modifications and we are currently working on extending this to dialog systems and general summarization.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*. pages 2754–2760.

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *Proceedings of NAACL-HLT16* pages 93–98.

Carlos A Colmenares, Marina Litvak, Amin Mantrach, and Fabrizio Silvestri. 2015. Heads: Headline generation as sequence prediction using an abstract feature-rich space. In *HLT-NAACL*. pages 133–142.

Dipanjan Das and André FT Martins. 2007. A survey on automatic text summarization. *Literature Survey for the Language and Statistics II course at CMU* 4:192–195.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* 22:457–479.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *EMNLP*. Citeseer, pages 1481–1491.

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

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. *arXiv preprint arXiv:1506.05865* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155* .

Konstantin Lopyrev. 2015. Generating news headlines with recurrent neural networks. *arXiv preprint arXiv:1512.01712* .

Inderjeet Mani. 2001. *Automatic summarization*, volume 3. John Benjamins Publishing.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In *Mining text data*, Springer, pages 43–76.

Joel Larocca Neto, Alex A Freitas, and Celso AA Kaestner. 2002. Automatic text summarization using a machine learning approach. In *Brazilian Symposium on Artificial Intelligence*. Springer, pages 205–215.

Ricardo Ribeiro, Luís Marujo, David Martins de Matos, Joao P Neto, Anatole Gershman, and Jaime Carbonell. 2013. Self reinforcement for important passage retrieval. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 845–848.

Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tür. 2010. Long story short–global unsupervised models for keyphrase based meeting summarization. *Speech Communication* 52(10):801–815.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Baskaran Sankaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal attention model for neural machine translation. *arXiv preprint arXiv:1608.02927* .

David Zajic, Bonnie Dorr, and Richard Schwartz. ???? Bbn/umd at duc-2004: Topiary.

Xiaodan Zhu and Gerald Penn. 2006. Comparing the roles of textual, acoustic and spoken-language features on spontaneous-conversation summarization. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. Association for Computational Linguistics, pages 197–200.

Xiaodan Zhu, Gerald Penn, and Frank Rudzicz. 2009. Summarizing multiple spoken documents: finding evidence from untranscribed audio. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, pages 549–557.

Get To The Point: Summarization with Pointer-Generator Networks

Abigail See
Stanford University
abisee@stanford.edu

Peter J. Liu
Google Brain
peterjliu@google.com

Christopher D. Manning
Stanford University
manning@stanford.edu

Abstract

Neural sequence-to-sequence models have provided a viable new approach for *abstractive* text summarization (meaning they are not restricted to simply selecting and rearranging passages from the original text). However, these models have two shortcomings: they are liable to reproduce factual details inaccurately, and they tend to repeat themselves. In this work we propose a novel architecture that augments the standard sequence-to-sequence attentional model in two orthogonal ways. First, we use a hybrid pointer-generator network that can copy words from the source text via *pointing*, which aids accurate reproduction of information, while retaining the ability to produce novel words through the *generator*. Second, we use *coverage* to keep track of what has been summarized, which discourages repetition. We apply our model to the *CNN / Daily Mail* summarization task, outperforming the current abstractive state-of-the-art by at least 2 ROUGE points.

Original Text (truncated): lagos, nigeria (cnn) a day after winning nigeria's presidency, ***muhammadu buhari*** told cnn's christiane amanpour that **he plans to aggressively fight corruption that has long plagued nigeria** and go after the root of the nation's unrest. *buhari* said he'll "rapidly give attention" to curbing violence in the northeast part of nigeria, where the terrorist group boko haram operates. by cooperating with neighboring nations chad, cameroon and niger, **he said his administration is confident it will be able to thwart criminals** and others contributing to nigeria's instability. for the first time in nigeria's history, the opposition defeated the ruling party in democratic elections. *buhari* defeated incumbent goodluck jonathan by about 2 million votes, according to nigeria's independent national electoral commission. **the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.**

Baseline Seq2Seq + Attention: **UNK UNK** says his administration is confident it will be able to **destabilize nigeria's economy**. **UNK** says his administration is confident it will be able to thwart criminals and other **nigerians**. **he says the country has long nigeria and nigeria's economy.**

Pointer-Gen: *muhammadu buhari* says he plans to aggressively fight corruption **in the northeast part of nigeria**. he says he'll "rapidly give attention" to curbing violence **in the northeast part of nigeria**. he says his administration is confident it will be able to thwart criminals.

Pointer-Gen + Coverage: *muhammadu buhari* says he plans to aggressively fight corruption that has long plagued nigeria. he says his administration is confident it will be able to thwart criminals. the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.

Figure 1: Comparison of output of 3 abstractive summarization models on a news article. The baseline model makes **factual errors**, a **nonsensical sentence** and struggles with OOV words *muhammadu buhari*. The pointer-generator model is accurate but **repeats itself**. Coverage eliminates repetition. The final summary is composed from **several fragments**.

1 Introduction

Summarization is the task of condensing a piece of text to a shorter version that contains the main information from the original. There are two broad approaches to summarization: *extractive* and *abstractive*. *Extractive methods* assemble summaries exclusively from passages (usually whole sentences) taken directly from the source text, while *abstractive methods* may generate novel words and phrases not featured in the source text – as a human-written abstract usually does. The extractive approach is easier, because copying large chunks of text from the source document ensures baseline levels of grammaticality and accuracy. On the other hand, sophisticated abilities that are crucial to high-quality summarization, such as paraphrasing, generalization, or the incorporation of real-world knowledge, are possible only in an abstractive framework (see Figure 5).

Due to the difficulty of abstractive summarization, the great majority of past work has been extractive (Kupiec et al., 1995; Paice, 1990; Saggion and Poibeau, 2013). However, the recent success of *sequence-to-sequence* models (Sutskever

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1073–1083
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1099

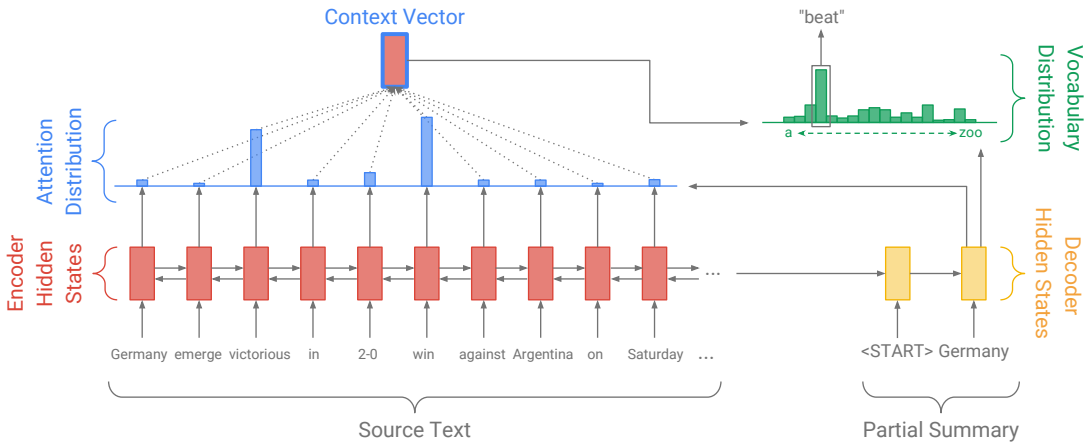


Figure 2: Baseline sequence-to-sequence model with attention. The model may attend to relevant words in the source text to generate novel words, e.g., to produce the novel word *beat* in the abstractive summary *Germany* ***beat*** *Argentina 2-0* the model may attend to the words *victorious* and *win* in the source text.

et al., 2014), in which recurrent neural networks (RNNs) both read and freely generate text, has made abstractive summarization viable (Chopra et al., 2016; Nallapati et al., 2016; Rush et al., 2015; Zeng et al., 2016). Though these systems are promising, they exhibit undesirable behavior such as inaccurately reproducing factual details, an inability to deal with out-of-vocabulary (OOV) words, and repeating themselves (see Figure 1).

In this paper we present an architecture that addresses these three issues in the context of multi-sentence summaries. While most recent abstractive work has focused on headline generation tasks (reducing one or two sentences to a single headline), we believe that longer-text summarization is both more challenging (requiring higher levels of abstraction while avoiding repetition) and ultimately more useful. Therefore we apply our model to the recently-introduced *CNN/ Daily Mail* dataset (Hermann et al., 2015; Nallapati et al., 2016), which contains news articles (39 sentences on average) paired with multi-sentence summaries, and show that we outperform the state-of-the-art abstractive system by at least 2 ROUGE points.

Our hybrid *pointer-generator* network facilitates copying words from the source text via *pointing* (Vinyals et al., 2015), which improves accuracy and handling of OOV words, while retaining the ability to *generate* new words. The network, which can be viewed as a balance between extractive and abstractive approaches, is similar to Gu et al.'s (2016) CopyNet and Miao and Blunsom's (2016) Forced-Attention Sentence Compression, that were applied to short-text summarization. We propose a novel variant of the *coverage vector* (Tu et al., 2016) from Neural Machine Translation, which we use to track and control coverage of the source document. We show that coverage is remarkably effective for eliminating repetition.

2 Our Models

In this section we describe (1) our baseline sequence-to-sequence model, (2) our pointer-generator model, and (3) our coverage mechanism that can be added to either of the first two models. The code for our models is available online.[1]

2.1 Sequence-to-sequence attentional model

Our baseline model is similar to that of Nallapati et al. (2016), and is depicted in Figure 2. The tokens of the article w_i are fed one-by-one into the encoder (a single-layer bidirectional LSTM), producing a sequence of *encoder hidden states* h_i. On each step t, the decoder (a single-layer unidirectional LSTM) receives the word embedding of the previous word (while training, this is the previous word of the reference summary; at test time it is the previous word emitted by the decoder), and has *decoder state* s_t. The *attention distribution* a^t is calculated as in Bahdanau et al. (2015):

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}}) \quad (1)$$

$$a^t = \text{softmax}(e^t) \quad (2)$$

where v, W_h, W_s and b_{attn} are learnable parameters. The attention distribution can be viewed as

[1] www.github.com/abisee/pointer-generator

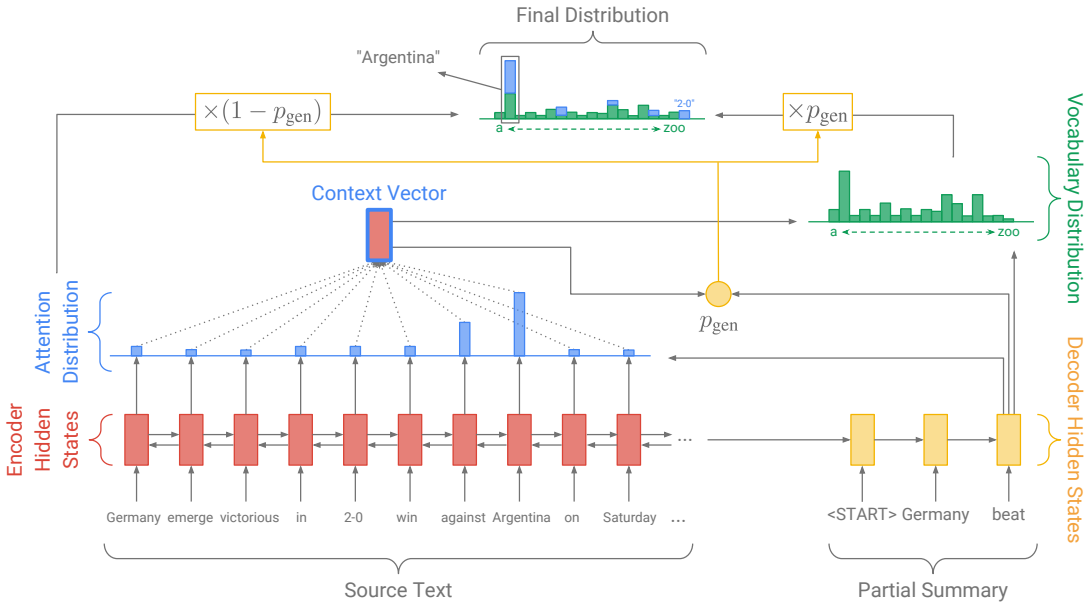


Figure 3: Pointer-generator model. For each decoder timestep a generation probability $p_{\text{gen}} \in [0,1]$ is calculated, which weights the probability of *generating* words from the vocabulary, versus *copying* words from the source text. The vocabulary distribution and the attention distribution are weighted and summed to obtain the final distribution, from which we make our prediction. Note that out-of-vocabulary article words such as *2-0* are included in the final distribution. Best viewed in color.

a probability distribution over the source words, that tells the decoder where to look to produce the next word. Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the *context vector* h_t^*:

$$h_t^* = \sum_i a_i^t h_i \tag{3}$$

The context vector, which can be seen as a fixed-size representation of what has been read from the source for this step, is concatenated with the decoder state s_t and fed through two linear layers to produce the vocabulary distribution P_{vocab}:

$$P_{\text{vocab}} = \text{softmax}(V'(V[s_t, h_t^*] + b) + b') \tag{4}$$

where V, V', b and b' are learnable parameters. P_{vocab} is a probability distribution over all words in the vocabulary, and provides us with our final distribution from which to predict words w:

$$P(w) = P_{\text{vocab}}(w) \tag{5}$$

During training, the loss for timestep t is the negative log likelihood of the target word w_t^* for that timestep:

$$\text{loss}_t = -\log P(w_t^*) \tag{6}$$

and the overall loss for the whole sequence is:

$$\text{loss} = \frac{1}{T}\sum_{t=0}^{T} \text{loss}_t \tag{7}$$

2.2 Pointer-generator network

Our pointer-generator network is a hybrid between our baseline and a pointer network (Vinyals et al., 2015), as it allows both copying words via pointing, and generating words from a fixed vocabulary. In the pointer-generator model (depicted in Figure 3) the attention distribution a^t and context vector h_t^* are calculated as in section 2.1. In addition, the *generation probability* $p_{\text{gen}} \in [0,1]$ for timestep t is calculated from the context vector h_t^*, the decoder state s_t and the decoder input x_t:

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}}) \tag{8}$$

where vectors w_{h^*}, w_s, w_x and scalar b_{ptr} are learnable parameters and σ is the sigmoid function. Next, p_{gen} is used as a soft switch to choose between *generating* a word from the vocabulary by sampling from P_{vocab}, or *copying* a word from the input sequence by sampling from the attention distribution a^t. For each document let the *extended vocabulary* denote the union of the vocabulary, and all words appearing in the source document. We obtain the following probability distribution over the extended vocabulary:

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i=w} a_i^t \tag{9}$$

Note that if w is an out-of-vocabulary (OOV) word, then $P_{\text{vocab}}(w)$ is zero; similarly if w does

not appear in the source document, then $\sum_{i:w_i=w} a_i^t$ is zero. The ability to produce OOV words is one of the primary advantages of pointer-generator models; by contrast models such as our baseline are restricted to their pre-set vocabulary.

The loss function is as described in equations (6) and (7), but with respect to our modified probability distribution $P(w)$ given in equation (9).

2.3 Coverage mechanism

Repetition is a common problem for sequence-to-sequence models (Tu et al., 2016; Mi et al., 2016; Sankaran et al., 2016; Suzuki and Nagata, 2016), and is especially pronounced when generating multi-sentence text (see Figure 1). We adapt the *coverage model* of Tu et al. (2016) to solve the problem. In our coverage model, we maintain a *coverage vector* c^t, which is the sum of attention distributions over all previous decoder timesteps:

$$c^t = \sum_{t'=0}^{t-1} a^{t'} \tag{10}$$

Intuitively, c^t is a (unnormalized) distribution over the source document words that represents the degree of coverage that those words have received from the attention mechanism so far. Note that c^0 is a zero vector, because on the first timestep, none of the source document has been covered.

The coverage vector is used as extra input to the attention mechanism, changing equation (1) to:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + w_c c_i^t + b_{\text{attn}}) \tag{11}$$

where w_c is a learnable parameter vector of same length as v. This ensures that the attention mechanism's current decision (choosing where to attend next) is informed by a reminder of its previous decisions (summarized in c^t). This should make it easier for the attention mechanism to avoid repeatedly attending to the same locations, and thus avoid generating repetitive text.

We find it necessary (see section 5) to additionally define a *coverage loss* to penalize repeatedly attending to the same locations:

$$\text{covloss}_t = \sum_i \min(a_i^t, c_i^t) \tag{12}$$

Note that the coverage loss is bounded; in particular $\text{covloss}_t \leq \sum_i a_i^t = 1$. Equation (12) differs from the coverage loss used in Machine Translation. In MT, we assume that there should be a roughly one-to-one translation ratio; accordingly the final coverage vector is penalized if it is more or less than 1. Our loss function is more flexible: because summarization should not require uniform coverage, we only penalize the overlap between each attention distribution and the coverage so far – preventing repeated attention. Finally, the coverage loss, reweighted by some hyperparameter λ, is added to the primary loss function to yield a new composite loss function:

$$\text{loss}_t = -\log P(w_t^*) + \lambda \sum_i \min(a_i^t, c_i^t) \tag{13}$$

3 Related Work

Neural abstractive summarization. Rush et al. (2015) were the first to apply modern neural networks to abstractive text summarization, achieving state-of-the-art performance on DUC-2004 and Gigaword, two sentence-level summarization datasets. Their approach, which is centered on the attention mechanism, has been augmented with recurrent decoders (Chopra et al., 2016), Abstract Meaning Representations (Takase et al., 2016), hierarchical networks (Nallapati et al., 2016), variational autoencoders (Miao and Blunsom, 2016), and direct optimization of the performance metric (Ranzato et al., 2016), further improving performance on those datasets.

However, large-scale datasets for summarization of *longer* text are rare. Nallapati et al. (2016) adapted the DeepMind question-answering dataset (Hermann et al., 2015) for summarization, resulting in the *CNN/Daily Mail* dataset, and provided the first abstractive baselines. The same authors then published a neural *extractive* approach (Nallapati et al., 2017), which uses hierarchical RNNs to select sentences, and found that it significantly outperformed their abstractive result with respect to the ROUGE metric. To our knowledge, these are the only two published results on the full dataset.

Prior to modern neural methods, abstractive summarization received less attention than extractive summarization, but Jing (2000) explored cutting unimportant parts of sentences to create summaries, and Cheung and Penn (2014) explore sentence fusion using dependency trees.

Pointer-generator networks. The pointer network (Vinyals et al., 2015) is a sequence-to-sequence model that uses the soft attention distribution of Bahdanau et al. (2015) to produce an output sequence consisting of elements from

the input sequence. The pointer network has been used to create hybrid approaches for NMT (Gulcehre et al., 2016), language modeling (Merity et al., 2016), and summarization (Gu et al., 2016; Gulcehre et al., 2016; Miao and Blunsom, 2016; Nallapati et al., 2016; Zeng et al., 2016).

Our approach is close to the Forced-Attention Sentence Compression model of Miao and Blunsom (2016) and the CopyNet model of Gu et al. (2016), with some small differences: (i) We calculate an explicit switch probability p_{gen}, whereas Gu et al. induce competition through a shared softmax function. (ii) We recycle the attention distribution to serve as the copy distribution, but Gu et al. use two separate distributions. (iii) When a word appears multiple times in the source text, we sum probability mass from all corresponding parts of the attention distribution, whereas Miao and Blunsom do not. Our reasoning is that (i) calculating an explicit p_{gen} usefully enables us to raise or lower the probability of all generated words or all copy words at once, rather than individually, (ii) the two distributions serve such similar purposes that we find our simpler approach suffices, and (iii) we observe that the pointer mechanism often copies a word while attending to multiple occurrences of it in the source text.

Our approach is considerably different from that of Gulcehre et al. (2016) and Nallapati et al. (2016). Those works train their pointer components to activate only for out-of-vocabulary words or named entities (whereas we allow our model to freely learn when to use the pointer), and they do not mix the probabilities from the copy distribution and the vocabulary distribution. We believe the mixture approach described here is better for abstractive summarization – in section 6 we show that the copy mechanism is vital for accurately reproducing rare but in-vocabulary words, and in section 7.2 we observe that the mixture model enables the language model and copy mechanism to work together to perform abstractive copying.

Coverage. Originating from Statistical Machine Translation (Koehn, 2009), coverage was adapted for NMT by Tu et al. (2016) and Mi et al. (2016), who both use a GRU to update the coverage vector each step. We find that a simpler approach – summing the attention distributions to obtain the coverage vector – suffices. In this respect our approach is similar to Xu et al. (2015), who apply a coverage-like method to image captioning, and Chen et al. (2016), who also incorporate a coverage mechanism (which they call 'distraction') as described in equation (11) into neural summarization of longer text.

Temporal attention is a related technique that has been applied to NMT (Sankaran et al., 2016) and summarization (Nallapati et al., 2016). In this approach, each attention distribution is divided by the sum of the previous, which effectively dampens repeated attention. We tried this method but found it too destructive, distorting the signal from the attention mechanism and reducing performance. We hypothesize that an early intervention method such as coverage is preferable to a post hoc method such as temporal attention – it is better to *inform* the attention mechanism to help it make better decisions, than to *override* its decisions altogether. This theory is supported by the large boost that coverage gives our ROUGE scores (see Table 1), compared to the smaller boost given by temporal attention for the same task (Nallapati et al., 2016).

4 Dataset

We use the *CNN/Daily Mail* dataset (Hermann et al., 2015; Nallapati et al., 2016), which contains online news articles (781 tokens on average) paired with multi-sentence summaries (3.75 sentences or 56 tokens on average). We used scripts supplied by Nallapati et al. (2016) to obtain the same version of the the data, which has 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs. Both the dataset's published results (Nallapati et al., 2016, 2017) use the *anonymized* version of the data, which has been pre-processed to replace each named entity, e.g., *The United Nations*, with its own unique identifier for the example pair, e.g., `@entity5`. By contrast, we operate directly on the original text (or *non-anonymized* version of the data),[2] which we believe is the favorable problem to solve because it requires no pre-processing.

5 Experiments

For all experiments, our model has 256-dimensional hidden states and 128-dimensional word embeddings. For the pointer-generator models, we use a vocabulary of 50k words for both source and target – note that due to the pointer network's ability to handle OOV words, we can use

[2] at `www.github.com/abisee/pointer-generator`

	ROUGE			METEOR	
	1	2	L	exact match	+ stem/syn/para
abstractive model (Nallapati et al., 2016)*	35.46	13.30	32.65	-	-
seq-to-seq + attn baseline (150k vocab)	30.49	11.17	28.08	11.65	12.86
seq-to-seq + attn baseline (50k vocab)	31.33	11.81	28.83	12.03	13.20
pointer-generator	36.44	15.66	33.42	15.35	16.65
pointer-generator + coverage	**39.53**	**17.28**	**36.38**	17.32	18.72
lead-3 baseline (ours)	40.34	17.70	36.57	20.48	22.21
lead-3 baseline (Nallapati et al., 2017)*	39.2	15.7	35.5	-	-
extractive model (Nallapati et al., 2017)*	39.6	16.2	35.3	-	-

Table 1: ROUGE F_1 and METEOR scores on the test set. Models and baselines in the top half are abstractive, while those in the bottom half are extractive. Those marked with * were trained and evaluated on the anonymized dataset, and so are not strictly comparable to our results on the original text. All our ROUGE scores have a 95% confidence interval of at most ± 0.25 as reported by the official ROUGE script. The METEOR improvement from the 50k baseline to the pointer-generator model, and from the pointer-generator to the pointer-generator+coverage model, were both found to be statistically significant using an approximate randomization test with $p < 0.01$.

a smaller vocabulary size than Nallapati et al.'s (2016) 150k source and 60k target vocabularies. For the baseline model, we also try a larger vocabulary size of 150k.

Note that the pointer and the coverage mechanism introduce very few additional parameters to the network: for the models with vocabulary size 50k, the baseline model has 21,499,600 parameters, the pointer-generator adds 1153 extra parameters (w_{h^*}, w_s, w_x and b_{ptr} in equation 8), and coverage adds 512 extra parameters (w_c in equation 11).

Unlike Nallapati et al. (2016), we do not pretrain the word embeddings – they are learned from scratch during training. We train using Adagrad (Duchi et al., 2011) with learning rate 0.15 and an initial accumulator value of 0.1. (This was found to work best of Stochastic Gradient Descent, Adadelta, Momentum, Adam and RMSProp). We use gradient clipping with a maximum gradient norm of 2, but do not use any form of regularization. We use loss on the validation set to implement early stopping.

During training and at test time we truncate the article to 400 tokens and limit the length of the summary to 100 tokens for training and 120 tokens at test time.[3] This is done to expedite training and testing, but we also found that truncating the article can *raise* the performance of the model (see section 7.1 for more details). For training, we found it efficient to start with highly-truncated sequences, then raise the maximum length once converged. We train on a single Tesla K40m GPU with a batch size of 16. At test time our summaries are produced using beam search with beam size 4.

We trained both our baseline models for about 600,000 iterations (33 epochs) – this is similar to the 35 epochs required by Nallapati et al.'s (2016) best model. Training took 4 days and 14 hours for the 50k vocabulary model, and 8 days 21 hours for the 150k vocabulary model. We found the pointer-generator model quicker to train, requiring less than 230,000 training iterations (12.8 epochs); a total of 3 days and 4 hours. In particular, the pointer-generator model makes much quicker progress in the early phases of training. To obtain our final coverage model, we added the coverage mechanism with coverage loss weighted to $\lambda = 1$ (as described in equation 13), and trained for a further 3000 iterations (about 2 hours). In this time the coverage loss converged to about 0.2, down from an initial value of about 0.5. We also tried a more aggressive value of $\lambda = 2$; this reduced coverage loss but increased the primary loss function, thus we did not use it.

We tried training the coverage model without the loss function, hoping that the attention mechanism may learn by itself not to attend repeatedly to the same locations, but we found this to be ineffective, with no discernible reduction in repetition. We also tried training with coverage from the first

[3]The upper limit of 120 is mostly invisible: the beam search algorithm is self-stopping and almost never reaches the 120th step.

iteration rather than as a separate training phase, but found that in the early phase of training, the coverage objective interfered with the main objective, reducing overall performance.

6 Results

6.1 Preliminaries

Our results are given in Table 1. We evaluate our models with the standard ROUGE metric (Lin, 2004b), reporting the F_1 scores for ROUGE-1, ROUGE-2 and ROUGE-L (which respectively measure the word-overlap, bigram-overlap, and longest common sequence between the reference summary and the summary to be evaluated). We obtain our ROUGE scores using the `pyrouge` package.[4] We also evaluate with the METEOR metric (Denkowski and Lavie, 2014), both in exact match mode (rewarding only exact matches between words) and full mode (which additionally rewards matching stems, synonyms and paraphrases).[5]

In addition to our own models, we also report the lead-3 baseline (which uses the first three sentences of the article as a summary), and compare to the only existing abstractive (Nallapati et al., 2016) and extractive (Nallapati et al., 2017) models on the full dataset. The output of our models is available online.[6]

Given that we generate plain-text summaries but Nallapati et al. (2016; 2017) generate anonymized summaries (see Section 4), our ROUGE scores are not strictly comparable. There is evidence to suggest that the original-text dataset may result in higher ROUGE scores in general than the anonymized dataset – the lead-3 baseline is higher on the former than the latter. One possible explanation is that multi-word named entities lead to a higher rate of n-gram overlap. Unfortunately, ROUGE is the only available means of comparison with Nallapati et al.'s work. Nevertheless, given that the disparity in the lead-3 scores is (+1.1 ROUGE-1, +2.0 ROUGE-2, +1.1 ROUGE-L) points respectively, and our best model scores exceed Nallapati et al. (2016) by (+4.07 ROUGE-1, +3.98 ROUGE-2, +3.73 ROUGE-L) points, we may estimate that we outperform the only previous abstractive system by at least 2 ROUGE points all-round.

[4] `pypi.python.org/pypi/pyrouge/0.1.3`
[5] `www.cs.cmu.edu/~alavie/METEOR`
[6] `www.github.com/abisee/pointer-generator`

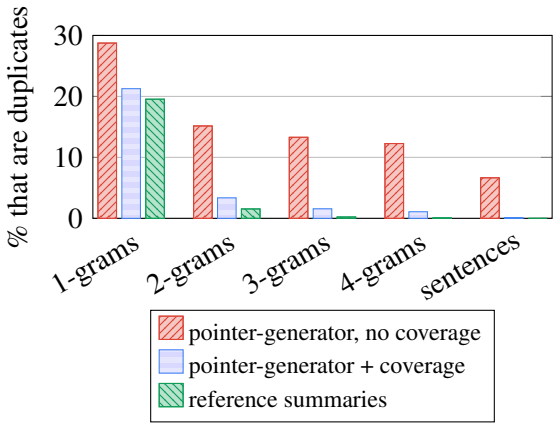


Figure 4: Coverage eliminates undesirable repetition. Summaries from our **non-coverage model** contain many duplicated n-grams while our **coverage model** produces a similar number as the **reference summaries**.

6.2 Observations

We find that both our baseline models perform poorly with respect to ROUGE and METEOR, and in fact the larger vocabulary size (150k) does not seem to help. Even the better-performing baseline (with 50k vocabulary) produces summaries with several common problems. Factual details are frequently reproduced incorrectly, often replacing an uncommon (but in-vocabulary) word with a more-common alternative. For example in Figure 1, the baseline model appears to struggle with the rare word *thwart*, producing *destabilize* instead, which leads to the fabricated phrase *destabilize nigeria's economy*. Even more catastrophically, the summaries sometimes devolve into repetitive nonsense, such as the third sentence produced by the baseline model in Figure 1. In addition, the baseline model can't reproduce out-of-vocabulary words (such as *muhammadu buhari* in Figure 1). Further examples of all these problems are provided in the supplementary material.

Our pointer-generator model achieves much better ROUGE and METEOR scores than the baseline, despite many fewer training epochs. The difference in the summaries is also marked: out-of-vocabulary words are handled easily, factual details are almost always copied correctly, and there are no fabrications (see Figure 1). However, repetition is still very common.

Our pointer-generator model with coverage improves the ROUGE and METEOR scores further, convincingly surpassing the best abstractive model

Article: smugglers lure arab and african migrants by offering discounts to get onto overcrowded ships if people bring more potential passengers, a cnn investigation has revealed. (...)
Summary: cnn investigation **uncovers** the **business inside** a **human smuggling ring**.

Article: eyewitness video showing white north charleston police officer michael slager shooting to death an unarmed black man has exposed discrepancies in the reports of the first officers on the scene. (...)
Summary: more **questions than answers emerge** in **controversial s.c.** police shooting.

Figure 5: Examples of highly abstractive reference summaries (**bold** denotes novel words).

of Nallapati et al. (2016) by several ROUGE points. Despite the brevity of the coverage training phase (about 1% of the total training time), the repetition problem is almost completely eliminated, which can be seen both qualitatively (Figure 1) and quantitatively (Figure 4). However, our best model does not quite surpass the ROUGE scores of the lead-3 baseline, nor the current best extractive model (Nallapati et al., 2017). We discuss this issue in section 7.1.

7 Discussion

7.1 Comparison with extractive systems

It is clear from Table 1 that extractive systems tend to achieve higher ROUGE scores than abstractive, and that the extractive lead-3 baseline is extremely strong (even the best extractive system beats it by only a small margin). We offer two possible explanations for these observations.

Firstly, news articles tend to be structured with the most important information at the start; this partially explains the strength of the lead-3 baseline. Indeed, we found that using only the first 400 tokens (about 20 sentences) of the article yielded significantly higher ROUGE scores than using the first 800 tokens.

Secondly, the nature of the task and the ROUGE metric make extractive approaches and the lead-3 baseline difficult to beat. The choice of content for the reference summaries is quite subjective – sometimes the sentences form a self-contained summary; other times they simply showcase a few interesting details from the article. Given that the articles contain 39 sentences on average, there are many equally valid ways to choose 3 or 4 highlights in this style. Abstraction introduces even more options (choice of phrasing), further decreasing the likelihood of matching the reference summary. For example, *smugglers profit from desperate migrants* is a valid alternative abstractive summary for the first example in Figure 5, but it scores 0 ROUGE with respect to the reference summary. This inflexibility of ROUGE is exacerbated by only having one reference summary, which has been shown to lower ROUGE's reliability compared to multiple reference summaries (Lin, 2004a).

Due to the subjectivity of the task and thus the diversity of valid summaries, it seems that ROUGE rewards safe strategies such as selecting the first-appearing content, or preserving original phrasing. While the reference summaries *do* sometimes deviate from these techniques, those deviations are unpredictable enough that the safer strategy obtains higher ROUGE scores on average. This may explain why extractive systems tend to obtain higher ROUGE scores than abstractive, and even extractive systems do not significantly exceed the lead-3 baseline.

To explore this issue further, we evaluated our systems with the METEOR metric, which rewards not only exact word matches, but also matching stems, synonyms and paraphrases (from a predefined list). We observe that all our models receive over 1 METEOR point boost by the inclusion of stem, synonym and paraphrase matching, indicating that they may be performing some abstraction. However, we again observe that the lead-3 baseline is not surpassed by our models. It may be that news article style makes the lead-3 baseline very strong with respect to any metric. We believe that investigating this issue further is an important direction for future work.

7.2 How abstractive is our model?

We have shown that our pointer mechanism makes our abstractive system more reliable, copying factual details correctly more often. But does the ease of copying make our system any less *abstractive*?

Figure 6 shows that our final model's summaries contain a much lower rate of novel n-grams (i.e., those that don't appear in the article) than the reference summaries, indicating a lower degree of abstraction. Note that the baseline model produces novel n-grams more frequently – however, this statistic includes all the incorrectly copied words, *UNK* tokens and fabrications alongside the good instances of abstraction.

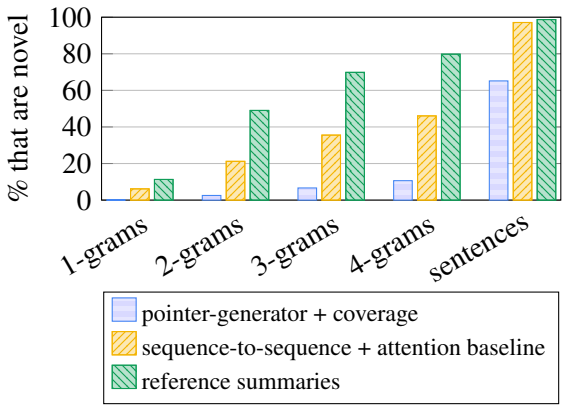


Figure 6: Although our **best model** is abstractive, it does not produce novel n-grams (i.e., n-grams that don't appear in the source text) as often as the **reference summaries**. The **baseline model** produces more novel n-grams, but many of these are erroneous (see section 7.2).

Article: andy murray (...) is into the semi-finals of the miami open , but not before getting a scare from 21 year-old austrian dominic thiem, who pushed him to 4-4 in the second set before going down 3-6 6-4, 6-1 in an hour and three quarters. (...)
Summary: andy murray **defeated** dominic thiem 3-6 6-4, 6-1 in an hour and three quarters.

Article: (...) wayne rooney smashes home during manchester united 's 3-1 win over aston villa on saturday. (...)
Summary: manchester united **beat** aston villa 3-1 at old trafford on saturday.

Figure 7: Examples of abstractive summaries produced by our model (**bold** denotes novel words).

In particular, Figure 6 shows that our final model copies whole article sentences 35% of the time; by comparison the reference summaries do so only 1.3% of the time. This is a main area for improvement, as we would like our model to move beyond simple sentence extraction. However, we observe that the other 65% encompasses a range of abstractive techniques. Article sentences are truncated to form grammatically-correct shorter versions, and new sentences are composed by stitching together fragments. Unnecessary interjections, clauses and parenthesized phrases are sometimes omitted from copied passages. Some of these abilities are demonstrated in Figure 1, and the supplementary material contains more examples.

Figure 7 shows two examples of more impressive abstraction – both with similar structure. The dataset contains many sports stories whose summaries follow the *X beat Y* ⟨*score*⟩ *on* ⟨*day*⟩ template, which may explain why our model is most confidently abstractive on these examples. In general however, our model does not routinely produce summaries like those in Figure 7, and is not close to producing summaries like in Figure 5.

The value of the generation probability p_{gen} also gives a measure of the abstractiveness of our model. During training, p_{gen} starts with a value of about 0.30 then increases, converging to about 0.53 by the end of training. This indicates that the model first learns to mostly copy, then learns to generate about half the time. However at test time, p_{gen} is heavily skewed towards copying, with a mean value of 0.17. The disparity is likely due to the fact that during training, the model receives word-by-word supervision in the form of the reference summary, but at test time it does not. Nonetheless, the generator module is useful even when the model is copying. We find that p_{gen} is highest at times of uncertainty such as the beginning of sentences, the join between stitched-together fragments, and when producing periods that truncate a copied sentence. Our mixture model allows the network to copy while simultaneously consulting the language model – enabling operations like stitching and truncation to be performed with grammaticality. In any case, encouraging the pointer-generator model to write more abstractively, while retaining the accuracy advantages of the pointer module, is an exciting direction for future work.

8 Conclusion

In this work we presented a hybrid pointer-generator architecture with coverage, and showed that it reduces inaccuracies and repetition. We applied our model to a new and challenging long-text dataset, and significantly outperformed the abstractive state-of-the-art result. Our model exhibits many abstractive abilities, but attaining higher levels of abstraction remains an open research question.

9 Acknowledgment

We thank the ACL reviewers for their helpful comments. This work was begun while the first author was an intern at Google Brain and continued at Stanford. Stanford University gratefully acknowledges the support of the DARPA DEFT Program AFRL contract no. FA8750-13-2-0040. Any opinions in this material are those of the authors alone.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *International Joint Conference on Artificial Intelligence*.

Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *Empirical Methods in Natural Language Processing*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *North American Chapter of the Association for Computational Linguistics*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *EACL 2014 Workshop on Statistical Machine Translation*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Association for Computational Linguistics*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Association for Computational Linguistics*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Neural Information Processing Systems*.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Applied natural language processing*.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *International ACM SIGIR conference on Research and development in information retrieval*.

Chin-Yew Lin. 2004a. Looking for a few good metrics: Automatic summarization evaluation-how many samples are enough? In *NACSIS/NII Test Collection for Information Retrieval (NTCIR) Workshop*.

Chin-Yew Lin. 2004b. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: ACL workshop*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. In *NIPS 2016 Workshop on Multi-class and Multi-label Learning in Extremely Large Label Spaces*.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Empirical Methods in Natural Language Processing*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Empirical Methods in Natural Language Processing*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Association for the Advancement of Artificial Intelligence*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çaglar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Computational Natural Language Learning*.

Chris D Paice. 1990. Constructing literature abstracts by computer: techniques and prospects. *Information Processing & Management* 26(1):171–186.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Empirical Methods in Natural Language Processing*.

Horacio Saggion and Thierry Poibeau. 2013. Automatic text summarization: Past, present and future. In *Multi-source, Multilingual Information Extraction and Summarization*, Springer, pages 3–21.

Baskaran Sankaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal attention model for neural machine translation. *arXiv preprint arXiv:1608.02927* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems*.

Jun Suzuki and Masaaki Nagata. 2016. RNN-based encoder-decoder approach with word frequency estimation. *arXiv preprint arXiv:1701.00138* .

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Empirical Methods in Natural Language Processing*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Association for Computational Linguistics*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Neural Information Processing Systems*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382* .

Supervised Learning of Automatic Pyramid for Optimization-Based Multi-Document Summarization

Maxime Peyrard and **Judith Eckle-Kohler**
Research Training Group AIPHES and UKP Lab
Computer Science Department, Technische Universität Darmstadt
www.aiphes.tu-darmstadt.de, www.ukp.tu-darmstadt.de

Abstract

We present a new supervised framework that learns to estimate automatic Pyramid scores and uses them for optimization-based extractive multi-document summarization. For learning automatic Pyramid scores, we developed a method for automatic training data generation which is based on a genetic algorithm using automatic Pyramid as the fitness function. Our experimental evaluation shows that our new framework significantly outperforms strong baselines regarding automatic Pyramid, and that there is much room for improvement in comparison with the upper-bound for automatic Pyramid.

1 Introduction

We consider extractive text summarization, the task of condensing a textual source, e.g., a set of source documents in multi-document summarization (MDS), into a short summary text. The quality of an automatic system summary is traditionally evaluated by comparing it against one or more reference summaries written by humans. This comparison is performed by means of an evaluation metric measuring indicators of summary quality and combining them into an aggregated score.

Many state-of-the-art summarization systems cast extractive summarization as an optimization problem and maximize an objective function in order to create good, i.e., high-scoring summaries. To this end, optimization-based systems commonly use an objective function which encodes exactly those quality indicators which are measured by the particular evaluation metric being used. Some systems even employ an approximation of the evaluation metric as objective function.

Consider as an example the ROUGE metric which has become a de-facto standard for summary evaluation (Lin, 2004). ROUGE computes the n-gram overlap between a system summary and a pool of reference summaries. There are several previous approaches which have used an approximation of ROUGE as the optimization objective (e.g., Sipos et al. (2012); Peyrard and Eckle-Kohler (2016a)).

However, ROUGE has been widely criticized for being too simplistic and not suitable for capturing important quality aspects we are interested in. In particular, ROUGE does not capture sentences which are semantically equivalent but expressed with different words (Nenkova et al., 2007).

Ideally, we would like to evaluate our summaries based on human judgments. A well-known example of such a human evaluation method is the so-called Pyramid method (Nenkova et al., 2007): it evaluates the particular quality aspect of content selection and is based on a manual comparison of Summary Content Units (SCUs) in reference summaries against SCUs in system summaries. While the resulting Pyramid score is much more meaningful and informative than ROUGE, it is very expensive to obtain, and – worse – not reproducible.

These issues have been addressed by a line of research aimed at automating the Pyramid evaluation (Harnly et al., 2005; Passonneau et al., 2013). Recently, Yang et al. (2016) developed a freely available off-the-shelf system for automatic Pyramid scoring called PEAK, which uses open Information Extraction (open IE) propositions as SCUs and relies on proposition comparison. Automatic Pyramid (AP) scores are reproducible, and unlike ROUGE, they are based on semantically motivated content units (SCUs) rather than word n-grams. Moreover, they correlate better with human judgments than ROUGE (Yang et al., 2016).

Given these recent advances in the automatic

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1084–1094
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1100

evaluation of summaries regarding content selection, we believe that research in optimization-based summarization should move away from ROUGE towards AP as a more meaningful evaluation metric to approximate and to optimize.

In our work, we are the first to explore this new direction and to systematically investigate the use of AP in optimization-based extractive summarization. We make the following contributions:

- We compute an upper-bound for AP with a Genetic Algorithm (GA), and compare it to the ROUGE upper-bound.
- We develop a new extractive MDS system specifically optimizing for an approximation of AP. Our system uses a supervised learning setup to learn an approximation of AP from automatically generated training data. We constrain the learned approximation of AP to be linear so that we can extract summaries efficiently via Integer Linear Programming (ILP). Our experimental evaluation shows that our approach significantly outperforms strong baselines on the AP metric.

The code both for the new upper-bound and for our ILP is available at `github.com/UKPLab/acl2017-optimize_pyramid`.

2 Background

In this section, we summarize the Pyramid method and the PEAK system, the automated version of Pyramid we consider in this work.

Pyramid The Pyramid method (Nenkova et al., 2007) is a manual evaluation method which determines to what extent a system summary covers the content expressed in a set of reference summaries. The comparison of system summary content to reference summary content is performed on the basis of SCUs which correspond to semantically motivated, subsentential units, such as phrases or clauses.

The Pyramid method consists of two steps: the creation of a *Pyramid set* from reference summaries, and second, *Pyramid scoring* of system summaries based on the Pyramid set. In the first step, humans annotate phrasal content units in the reference summaries and group them into clusters of semantically equivalent phrases. The resulting clusters are called SCUs and the annotators assign an *SCU label* to each cluster, which is a sentence describing the cluster content in their own words. The final set of SCUs forms the Pyramid set. Each SCU has a weight corresponding to the number of reference summaries in which the SCU appears. Since each SCU must not appear more than once in each reference summary, the maximal weight of an SCU is the total number of reference summaries. In the second step, humans annotate phrasal content units in a system summary and align them to the corresponding SCUs in the Pyramid set. The Pyramid score of a system summary is then calculated as the sum of the SCU weights for all Pyramid set SCUs being aligned to annotated system summary phrases.

PEAK The AP system PEAK by Yang et al. (2016) uses *clauses* as the content expressing units and represents them as propositions in the open IE paradigm. An open IE proposition is a triple of subject, predicate and object phrases. PEAK uses the state-of-the-art system clausIE (Del Corro and Gemulla, 2013) for proposition extraction.

While PEAK includes the automatic creation of Pyramid sets from reference summaries, as well as automatic Pyramid scoring of system summaries, in this work, we use PEAK for automatic scoring only. As for the Pyramid sets, we can assume that these have already been created, either via PEAK or by humans (e.g., using the TAC 2009 data[1]).

Since automatic scoring with PEAK requires that the Pyramid sets consist of representative open IE propositions which constitute the automated counterparts of the SCUs, we first need to represent the manually constructed SCUs as open IE propositions, too. To this end, we use clausIE to extract an open IE proposition from each SCU *label* – a sentence describing the cluster content. As a result, each pyramid set is represented as a list of propositions $\{p_j\}$ with a weight taken from the underlying SCU.

For scoring, PEAK processes a system summary with clausIE, converting it from a list of sentences to a list of propositions $\{s_i\}$. A bipartite graph G is constructed, where the two sets of nodes are the summary propositions $\{s_i\}$ and the pyramid propositions $\{p_j\}$. An edge is drawn between s_i and p_j if the similarity is above a given threshold. PEAK computes the similarity with the ADW system (*Align, Disambiguate and Walk*), a system for computing text similarity based on WordNet, which reaches state-of-the-

[1] `http://tac.nist.gov/2009/Summarization`

art performance but is slow (Pilehvar et al., 2013). Since each system summary unit can be aligned to at most one SCU, the alignment of the summary propositions $\{s_i\}$ and the pyramid propositions $\{p_j\}$ is equivalent to finding a maximum weight matching, which PEAK solves using the Munkres-Kuhn bipartite graph algorithm. From the matched pyramid propositions $\{p_j\}$ the final pyramid score is computed.

3 Approach

3.1 Upper-bound for Automatic Pyramid

We start by computing upper-bound summaries according to AP in order to gain a better understanding of the metric.

Notations Let $D = \{s_i\}$ be a document collection considered as a set of sentences. A summary S is simply a subset of D. We use p^{pyr} to denote the set of propositions in the Pyramid sets extracted from the SCU labels using clausIE.

The upper-bound is the set of sentences S^* with the best AP score.

Method The task is to extract the set of sentences which contains the propositions matching most of the highest-weighted SCUs, thus resulting in the best matching of propositions, i.e., the highest AP score possible. Formally, we have to solve the following optimization problem:

$$S^* = \underset{S}{\operatorname{argmax}}\, AutoPyr(S) \qquad (1)$$

Unfortunately, it cannot be solved directly via ILP because of the Munkres-Kuhn bipartite graph algorithm within AP. While Munkres-Kuhn is an ILP, we solve a different problem. In our problem, Munkres-Kuhn would act as constraint because we are looking for the best matching among all valid matchings. Munkres-Kuhn only yields the valid matching for one particular set of sentences. One global ILP can be written down by enumerating all possible matchings in the constraints but it will have a completely unrealistic runtime.

Instead, we have to rely on search-based algorithms and compute summaries close to the upper-bound. We search for such an approximate solution by employing a meta-heuristic solver introduced recently for extractive MDS by Peyrard and Eckle-Kohler (2016a). Specifically, we use the tool published with their paper.[2] Their meta-heuristic solver implements a Genetic Algorithm (**GA**) to create and iteratively optimize summaries over time.

In this implementation, the individuals of the population are the candidate solutions which are valid extractive summaries. Valid means that the summary meets the length constraint. Each summary is represented by a binary vector indicating for each sentence in the source document whether it is included in the summary or not. The size of the population is a hyper-parameter that we set to 100. Two evolutionary operators are applied: the mutation and the reproduction. The mutation happens to several randomly chosen summaries by randomly removing one of its sentences and adding a new one that does not violate the length constraint. The reproduction is performed by randomly extracting a valid summary from the union of sentences of randomly selected parent summaries. Both operators are controlled by hyper-parameters which we set to their default values.

In our scenario, *the fitness function is the AP metric*, which takes a summary S as input and outputs its AP score. S is converted into a list of propositions p^S by looking-up the propositions of each sentence in S from a pre-computed hash-map. For all sentences in the document collection D, the hash-map stores the corresponding propositions. Then the Munkres-Kuhn algorithm is applied to p^S and p^{pyr} in order to find matching propositions, and finally the scores of their corresponding SCUs are used to evaluate the fitness of the summary.

The runtime might become an issue, because the similarity computation between propositions via ADW is slow. However, all the necessary information is present in the similarity matrix A defined by:

$$A_{ij} = ADW(p_i^D, p_j^{pyr}) \qquad (2)$$

Here A_{ij} is the semantic similarity between the proposition p_i^D from the source document i and the proposition p_j^P from the Pyramid set j. A has dimensions $m \times n$ if m is the number of propositions in the document collection and n the number of propositions in the Pyramid set. We keep the runtime low by pre-computing the similarity matrix A.

With a population of 100 summaries in the GA, the algorithm converges in less than a minute to high scoring summaries, which we can expect to be close to the real upper-bound.

[2] https://github.com/UKPLab/coling2016-genetic-swarm-MDS

3.2 Supervised Setup to Learn an Approximation of AP

We denote the true AP scoring function by π^*. π^* scores summaries by matching the summary propositions to the Pyramid propositions in P_{pyr} as described before. In this work, we aim to learn a function π, which approximates π^* without having access to P_{pyr}, but only to the document collection D.

Formally, it means that over all document collections $\mathcal{D}$ and all summaries $\mathcal{S}$, we look for π which minimizes the following loss:

$$\mathcal{L}(\pi) = \sum_{D \in \mathcal{D}} \sum_{S \in \mathcal{S}} \|\pi(D, S) - \pi^*(P_{pyr}, S)\|^2 \quad (3)$$

This states that the learned π minimizes the squared distance from π^* over the available training data.

Model Note that we simply denote $\pi(D, S)$ by $\pi(S)$ as it is not ambiguous which document collection is used when S is a summary of D.

In order to be able to use an exact and efficient solver like ILP, we constrain π to be a linear function. Therefore, we look for π of the following form:

$$\pi(S) = \sum_{s \in S} f_\theta(s) - \sum_{i>j} g_\gamma(s_i \cap s_j) \quad (4)$$

Two functions are jointly learned: f_θ is a function scoring individual sentences, and g_γ is a function scoring the intersection of sentences. $\theta \cup \gamma$ is the set of learned paramaters.

We can interpret this learning scenario as jointly learning the sentence importance and the redundancy to get π as close as possible to the true AP π^*. f_θ represents the notion of importance learned in the context of AP, while g_γ contains notions of coherence and redundancy by scoring sentence intersections. This scenario is intuitive and inspired by previous work on summarization (McDonald, 2007).

Now, we explain how to learn these two functions while enforcing π to be linear. Suppose each sentence is represented by a feature set ϕ and each sentence intersections is represented by $\phi_\cap$, then the set of features for a summary S is:

$$\Phi(S) = \{\bigcup_{s \in S} \phi(s) \cup \bigcup_{i>j} \phi_\cap(s_i \cap s_j)\} \quad (5)$$

It is clear that the number of features is variable and depends on the number m of sentences in S. In order to deal with a variable number of sentences as input, one could use recurrent neural networks, but at the cost of loosing linearity.

Instead, to keep the linearity and to cope with variable sized inputs, we employ linear models for both f_θ and g_γ:

$$\pi(S) = \sum_{s \in S} \theta \cdot \phi(s) - \sum_{i>j} \gamma \cdot \phi_\cap(s_i \cap s_j) \quad (6)$$

By leveraging the properties of linear models we end-up with the following formulation:

$$\pi(S) = \theta \cdot \sum_{s \in S} \phi(s) - \gamma \cdot \sum_{i \geq j} \phi_\cap(s_i \cap s_j) \quad (7)$$

Because of the linear models, we can sum features over sentences and over sentence intersections to obtain a fixed size feature set:

$$\Phi^\Sigma(S) = \{\phi^\Sigma(S) \cup \phi^\Sigma_\cap(S)\} \quad (8)$$

where we introduced the following notations:

$$\phi^\Sigma(S) = \sum_{s \in S} \phi(s)$$
$$\phi^\Sigma_\cap(S) = \sum_{i>j} \phi(s_i \cap s_j)$$

Suppose ϕ is composed of k features and $\phi_\cap$ of n features. Then $\phi^\Sigma(S)$ is a vector of dimension k, and similarly $\phi^\Sigma_\cap(S)$ is of dimension n. Finally, Φ_Σ is a fixed size feature set of dimension $k + n$.

The function π as defined in equation 6 is still linear with respect to sentence and sentence intersection features, which is convenient for the subsequent summary extraction stage.

Features While any feature set for sentences ϕ and for sentence intersections $\phi_\cap$ could be used, we focused on simple ones in this work.

For a sentence s, $\phi(s)$ consists of the following features:

- **Sentence length** in number of words.
- **Sentence position** as an integer number starting from 0.
- **Word overlap with title:** Jaccard similarity between the unigrams in the title t and a sentence s:

$$Jaccard(s, t) = \frac{|t \cap s|}{|t \cup s|} \quad (9)$$

- **Sum of frequency** of unigrams and bigrams in the sentence.

- **Sum of TF*IDF** of unigrams and bigrams in the sentence. The idf of unigrams and bigrams is trained on a background corpus of DBpedia articles.[3]

- **Centrality** of the sentence computed via PageRank: A similarity matrix is built between sentences in the document collection based on their TF*IDF vector similarity. Then a power method is applied on the similarity matrix to get PageRank scores of individual sentences. It is similar to the classic LexRank algorithm (Erkan and Radev, 2004).

- **Propositions centrality:** We also use the centrality feature for propositions. Each sentence is scored by the sum of the centrality of its propositions. As PEAK is based on propositions, we expect proposition-level features to provide a useful signal.

Finally, $\phi_\cap(s_i \cap s_j)$ consists of the unigram, bigram and trigram **overlap** between the two sentences s_i and s_j.

Training The model is trained with a standard linear least squares regression using pairs of $(\Phi(S), \pi^*(S))$ as training examples. Because our approach relies on an automatic metric, an arbitrarily large number of summaries and their corresponding scores can be generated. In contrast, getting manual Pyramid annotations for a large number of summaries would be expensive and time-consuming.

As training examples we take the population of scored summaries created by the same GA we use for computing upper-bound summaries. It is important to note that this GA is also a perfect generator of training instances: the summaries in its population are already scored because the fitness function is the AP metric. Indeed, for each topic, an arbitrarily large amount of scored summaries can be generated by adjusting the size of the population. Moreover, the summaries in the population are very diverse and have a wide range of scores, from almost upper-bound to completely random.

Optimization-based Summary Extraction Since the function π is constrained to be linear, we can extract the best scoring summary by solving an ILP.

Let x be a binary vector indicating whether sentence i is in the summary or not. Similarly, let α be a binary matrix indicating whether both sentence i and j are in the summary. Finally, let K be the length constraint. With these notations, the best summary is extracted by solving the follwogin ILP:

$$\begin{aligned}\underset{S}{\operatorname{argmax}} \sum_{s_i \in S} x_i * \theta \cdot \phi(s_i) - \sum_{i \geq j} \alpha_{i,j} * \gamma \cdot \phi_\cap(s_i \cap s_j) \\ \sum_{i=1}^{m} x_i * len(s_i) \leq K \\ \forall (i,j), \alpha_{i,j} - x_i \leq 0 \\ \forall (i,j), \alpha_{i,j} - x_j \leq 0 \\ \forall (i,j), x_i + x_j - \alpha_{i,j} \leq 1\end{aligned}$$

Which is the ILP directly corresponding to maximizing π as defined by equation 6. Note that $\cdot$ is the dot product while $*$ is the scalar multiplication in $\mathbb{R}$.

4 Experiments

4.1 Setup

Dataset We perform our experiments on a multi-document summarization dataset from the Text Analysis Conference (TAC) shared task in 2009, TAC-2009.[4] TAC-2009 contains 44 topics, each consisting of 10 news articles to be summarized in a maximum of 100 words. In our experiments, we use only the so-called initial summaries (A summaries), but not the update summaries. For each topic, there are 4 human reference summaries and a manually created Pyramid set. As described in section 2, we pre-processed these Pyramid sets with clausIE in order to make them compatible with PEAK.

Metrics We primarily evaluate our system via automatic Pyramid scoring from PEAK, after pre-processing the summaries with clausIE. PEAK has a parameter t which is the minimal similarity value required for matching a summary proposition and a Pyramid proposition. We use two different values: $t = 0.6$ (AP-60) and $t = 0.7$ (AP-70).

For completeness, we also report the ROUGE scores identified by Owczarzak et al. (2012a) as strongly correlating with human evaluation methods: ROUGE-1 (R-1) and ROUGE-2 (R-2) recall with stemming and stopwords not removed.

Finally, we perform significance testing with t-test to compare differences between two means.[5]

[3] http://wiki.dbpedia.org/nif-abstract-datasets

[4] http://tac.nist.gov/2009/Summarization/

[5] The symbol * indicates that the difference compared to

4.2 Automatic Evalution

Upper-bound Comparison We compute the set of upper-bound summaries for both ROUGE-2 (R-UB) and for AP (AP-UB).[6] Both sets of upper-bound summaries are evaluated with ROUGE and AP, and the results are reported in Table 1.

	R-1	R-2	AP-60	AP-70
R-UB	**0.4722***	**0.2062***	0.5088	0.3074
AP-UB	0.3598	0.1057	**0.5789***	**0.3790***

Table 1: Upper bound comparison between ROUGE and Automatic Pyramid (AP).

Interestingly, we observe significant differences between the two upper-bounds. While it is obvious that each set of upper-bound summaries reaches the best score on the metric it maximizes, the same summary set scores much worse when evaluated with the other metric. This observation empirically confirms that the two metrics measure different properties of system summaries.

Moreover, the upper-bound for AP gives us information about the room for improvement that summarization systems have with respect to AP. This is relevant in the next paragraph, where we compare systems in an end-to-end evaluation.

End-to-end Evaluation We evaluate the quality of the summaries extracted by the summarizer $\pi - ILP$ in a standard end-to-end evaluation scenario. $\pi - ILP$ is the system composed of the learned function π and the ILP defined in the previous section.

Learning π Using our GA data generation method, we produce 100 scored summaries for each of the 44 topics in TAC2009 while computing the upper-bound. We use the threshold value of 0.65 as a compromise between AP-60 and AP-70. The data generated have scores ranging from 0. to 0.4627 with an average of 0.1615. The data is well distributed because the standard deviation is 0.1449. A highly diverse set of summaries is produced, because on average two summaries in the training set only have 1.5% sentences in common, and most of the sentences of the source documents are contained in at least one summary.

The model is then trained in a leave-one-out cross-validation setup. The parameters θ and γ are trained on all topics but one. The trained model is used to extract a high-scoring summary on the remaining topic by solving the ILP defined above.

	R-1	R-2	AP-60	AP-70
TF*IDF	0.3251	0.0626	0.2857	0.1053
LexRank	0.3539	0.0900	0.3969	0.1854
ICSI	**0.3670**	**0.1030**	0.3520	0.1568
JS-Gen	0.3381	0.0868	0.3745	0.1463
π-ILP	0.3498	0.0867	**0.4402***	**0.2109***

Table 2: End-to-end evaluation of our approach on TAC-2009.

Our framework is compared to the following baselines:

TF*IDF weighting A simple heuristic introduced by Luhn (1958) where each sentence receives a score from the TF*IDF of its terms. The best sentences are greedily extracted until the length constraint is met. We use the implementation available in the sumy package.[7]

LexRank (Erkan and Radev, 2004) is a popular graph-based approach. A similarity graph $G(V, E)$ is constructed where V is the set of sentences and an edge e_{ij} is drawn between sentences v_i and v_j if and only if the cosine similarity between them is above a given threshold. Sentences are scored according to their PageRank score in G. It is also available in the sumy package.

ICSI (Gillick and Favre, 2009) is a recent system that has been identified as one of the state-of-the-art systems by Hong et al. (2014). It is an ILP framework that extracts a summary by solving a maximum coverage problem considering the most frequent bigrams in the source documents. We use the Python implementation released by Boudin et al. (2015).

JS-Gen (Peyrard and Eckle-Kohler, 2016a) is a recent approach which uses a GA to minimize the Jensen-Shannon (JS) divergence between the extracted summary and the source documents. JS divergence measures the difference between probability distributions of words in the source documents and in the summary.

Results We report the performance of $\pi - ILP$ in comparison to the baselines in Table 2.

The results confirm an expected behavior. Our supervised framework which aims at approximating and maximizing AP, easily and significantly outperforms all the other baselines when evaluated

the previous best baseline is significant with $p \leq 0.05$.

[6] We use the parameter $t = 0.6$ during the upper-bound computation of AP-UB.

[7] https://github.com/miso-belica/sumy

with AP for both values of the threshold. While the system is not designed with ROUGE in mind, it still performs reasonably well in the ROUGE evaluation, even though it does not outperform previous works.

In general, the two metrics ROUGE and AP do not produce the same rankings of systems. This is another piece of empirical evidence that they measure different properties of summaries.

When we compare the system performances to the upper-bound scores reported in Table 1, we see that there is still a large room for improvements. We take a closer look at this performance gap in the next paragraph where we evaluate the learning component of our approach.

Evaluation of Learned π In this paragraph, we evaluate the learning of π as an approximation of π^*. We do so by measuring the correlation between π and the true AP π^*.

We report three correlation metrics to evaluate and compare the ranking of summaries induced by π and π^*: Pearson's r, Spearman's ρ and NDCG. Pearson's r is a value correlation metric which depicts linear relationship between the scores produced by two ranking lists.

Spearman's ρ is a rank correlation metric which compares the ordering of systems induced by the two ranking lists.

NDCG is a metric from information retrieval which compares ranked lists and puts a special emphasis on the top elements by applying logarithm decay weighting for elements further down in the list. Intuitively, it describes how well the π function is able to recognize the best scoring summaries. In our case, it is particularly desirable to have a high NDCG score, because the optimizer extracts summaries with high π scores; we want to confirm that top scoring summaries are also among top scoring summaries according to the true π^*.

For comparison, we report how well our baselines correlate with π^*. For this, we consider the scoring function for summaries which is part of all our baselines, and which they explicitly or implicitly optimize: **TF*IDF** greedily maximizes f_{TF*IDF}, the sum of the frequency of the words in the summary. **ICSI** maximizes the sum of the document frequency of bigrams (f_{ICSI}). **LexRank** maximizes $f_{LexRank}$, the sum of the PageRank of sentences in the summary, and f_{JS} is the JS divergence between the summary and the source documents optimized by **JS-Gen**.

	Pearson's r	Spearman's ρ	NDCG
f_{TF*IDF}	0.1246	0.0765	0.8869
$f_{LexRank}$	0.1733	0.0879	0.8774
f_{ICSI}	0.3742	0.3295	0.8520
f_{JS}	0.4074	0.3833	0.8803
π	**0.4929***	**0.4667***	**0.9429***

Table 3: Performance of the supervised learning of π on TAC-2009 in a leave-one-out cross-validation.

For our supervised learning of π, the training procedure is the same as described in the previous section. The correlation scores are averaged over topics and reported in Table 3.

We observe that π is able to approximate AP significantly better than any baseline for all metrics. This explains why optimizing π with ILP outperforms the baseline systems in the end-to-end evaluation (Table 2).

The learned π achieves a high NDCG, indicating that optimizing π produces summaries very likely to have high π^* scores. This means that π is capable of accurately identifying high-scoring summaries, which again explains the strong performance of $\pi - ILP$. The fact that the overall correlations are lower for every system shows that it is difficult to predict π for poor and average quality summaries.

It is interesting to observe that features such as unigram and bigram frequency, which are known to be strong features to approximate ROUGE, are less useful to approximate the more complex AP.

Feature Weights The advantage of linear models is their interpretability. One can investigate the contribution of each feature by looking at its corresponding weight learned during training. The sign of the weight indicates whether the feature correlates positively or negatively with the results, and its amplitude determines the importance of this feature in the final estimation.

We observe that the most useful feature is the proposition centrality, which confirms our expectation that proposition-based features are useful for approximating PEAK. The bigram coverage has also a high weight explaining the strong performance of ICSI. The least useful feature is the sentence position, even if it still contains some useful signal.

Interestingly, the analysis of features from the

	Pearson's r	Spearman's ρ	NDCG
$ROUGE-1$	0.3292	0.3187	0.7195
$ROUGE-2$	0.3292	0.2936	0.7259

Table 4: Correlation between ROUGE-1 and ROUGE-2 with AP on the automatically generated training data for TAC-2009.

sentence intersection reveals a slightly positive correlation for the unigram and bigram overlap, but a negative correlation for trigram overlap. Our interpretation is that the model learns that good summaries tend to have repeated unigrams and bigrams to ensure some coherence, while the repeated trigrams are more indicative of undesired redundancy.

Agreement between ROUGE and AP In the previous paragraphs, we already saw that different metrics produce different rankings of systems. We want to investigate this further and understand to what extent ROUGE and AP disagree. To that end, we use the summaries automatically generated by the genetic algorithm during the upper-bound computation. Remember that for each topic of TAC-2009 it produces 100 summaries with a wide range of AP scores. We then score these summaries with both ROUGE-1 and ROUGE-2 and compare how ROUGE metrics correlate with AP. In order to get a meaningful picture, we use the same three correlation metrics as above: Pearson's, Spearman's ρ and NDCG. The results are presented in Table 4.

We observe a low correlation between ROUGE metrics and AP in terms of both rank correlation (Spearman's ρ) and value correlation (Pearson's r). Even though the NDCG numbers are better, the correlation is also relatively low given that higher numbers are usually expected for NDCG (also observed in Table 3).

This analysis confirms the initial claim that ROUGE and AP behave quite differently and measure different aspects of summary quality. Therefore, we believe systems developed and trained for AP are worth studying because they necessarily capture different aspects of summarization.

5 Related Work

We discuss (i) related work in extractive summarization where an approximation of an automatic evaluation metric was optimized, and (ii) work related to AP specifically.

As ROUGE is the metric predominantly used for evaluation of extractive summarization, there are several previous optimization-based approaches which included an approximation of ROUGE in the objective function to maximize. For example, Takamura and Okumura (2010) and Sipos et al. (2012) performed structured output learning (using pairs of summaries and their ROUGE scores available in benchmark datasets as training examples) and thereby learned to maximize the ROUGE scores of the system summaries. Peyrard and Eckle-Kohler (2016b) on the other hand, learned an approximation of ROUGE scores for individual sentences in a supervised setup, and subsequently employed these estimated sentence scores in an ILP formulation to extract summaries.

There is also recent work on considering fully automatic evaluation metrics (not relying on human reference summaries), such as the JS divergence as optimization objective. Peyrard and Eckle-Kohler (2016a) used metaheuristics to minimize JS divergence in a multi-document summarization approach and showed that the resulting extractive summaries also scored competitively using ROUGE.

Regarding AP, there is not much prior work apart from the papers where the different variants of AP have been presented (Harnly et al., 2005; Passonneau et al., 2013; Yang et al., 2016). Especially, there is no prior work in optimization-based extractive summarization which has developed an approximation of AP and used it in an objective function.

However, AP as an evaluation metric is becoming ever more important in the context of abstractive summarization, a research topic which has been gaining momentum in the last few years. For example Li (2015) and Bing et al. (2015) use an earlier version of AP based on distributional semantics (Passonneau et al., 2013) to evaluate abstractive multi-document summarization.

6 Discussion and Future Work

We presented a supervised framework that learns automatic Pyramid scores and uses them for optimization-based summary extraction. Using the TAC-2009 multi-document summarization dataset, we performed an upper-bound analysis for AP, and we evaluated the summaries extracted with our framework in an end-to-end evaluation

using automatic evaluation metrics. We observed that the summaries extracted with our framework achieve significantly better AP scores than several strong baselines, but compared to the upper-bound for AP, there is still a large room for improvement.

We show that AP and ROUGE catch different aspects of summary quality, but further work would be needed in order to substantiate the claim that AP is indeed better than ROUGE. One way of doing so would be to perform a human evaluation of high-scoring summaries according to ROUGE and AP. In general, ROUGE-1 and ROUGE-2 were considered as the baselines for validating the performance of AP because these variants strongly correlate with human evaluation methods (Owczarzak et al., 2012a,b). However, the comparison could be repeated with ROUGE-3, ROUGE-4 and ROUGE-BE, which have been found to predict manual Pyramid better than ROUGE-1 and ROUGE-2 (Rankel et al., 2013).

More generally, we see two main directions for future research: (i) the more specific question on how to improve the approximation of AP and (ii) the general need for more research on AP.

There are several possible ways how to improve the approximation of AP. First, more semantically-oriented features could be developed, e.g., features based on propositions rather than sentences or n-grams, or word embedding features encoding a large amount of distributional semantic knowledge (Mikolov et al., 2013). Second, the linearity constraint we used for efficiency reasons could be relaxed. Modeling AP as a non-linear function will presumably enhance the approximation. For the extraction of summaries based on a non-linear function, greedy algorithms or search-based strategies could be used, e.g., the GA we used in this work for the upper-bound computation.

We see a general need for more research on AP, because the way AP measures the quality aspect of content selection is not only more meaningful than ROUGE, but also applicable to the growing field of abstractive summarization.

An important direction would be the improvement of AP itself, both in terms of methods used to compute AP, and in terms of tools: while the current off-the-shelf system PEAK is a promising start, it is very slow and therefore difficult to apply in practice.

In this context, we would like to stress that our GA-based method to create training data for learning a model of AP can easily be adapted to any automatic scoring metric, and specifically to other or future AP variants.

Finally, we hope to encourage the community to move away from ROUGE and instead consider AP as the main summary evaluation metric. This would be especially interesting for optimization-based approaches, since the quality of the summaries created by such approaches depends on the quality of the underlying scoring metric.

7 Conclusion

We presented the first work on AP in optimization-based extractive summarization. We computed an upper-bound for AP and developed a supervised framework which learns an approximation of AP based on automatically generated training instances. We could access a large number of high-quality training data by using the population of a genetic algorithm. Our end-to-end evaluation showed that of our framework significantly outperforms strong baselines on the AP metric, but also revealed a large room for improvement in comparison to the upper-bound, which motivates future work on developing systems with better performance on the semantically motivated AP metric.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1, and via the German-Israeli Project Cooperation (DIP, grant No. GU 798/17-1).

References

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. 2015. Abstractive Multi-Document Summarization via Phrase Selection and Merging. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1587–1597.

Florian Boudin, Hugo Mougard, and Benoit Favre. 2015. Concept-based Summarization using Integer Linear Programming: From Concept Pruning to Multiple Optimal Solutions. In *Proceedings of*

the 2015 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Lisbon, Portugal, pages 1914–1918.

Luciano Del Corro and Rainer Gemulla. 2013. ClausIE: Clause-based Open Information Extraction. In *Proceedings of the 22Nd International Conference on World Wide Web*. ACM, Rio de Janeiro, Brazil, pages 355–366.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *Journal of Artificial Intelligence Research* pages 457–479.

Dan Gillick and Benoit Favre. 2009. A Scalable Global Model for Summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*. Association for Computational Linguistics, Boulder, Colorado, pages 10–18.

Aaron Harnly, Rebecca Passonneau, and Owen Rambow. 2005. Automation of Summary Evaluation by the Pyramid Method. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*. Borovets, Bulgaria, pages 226–232.

Kai Hong, John Conroy, benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A Repository of State of the Art and Competitive Baseline Summaries for Generic News Summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland, pages 1608–1616.

Wei Li. 2015. Abstractive Multi-document Summarization with Semantic Information Extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1908–1913.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Association for Computational Linguistics, Barcelona, Spain, pages 74–81.

Hans Peter Luhn. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research Development* 2:159–165.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the 29th European Conference on IR Research*. Springer-Verlag, Rome, Italy, ECIR'07, pages 557–564.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The Pyramid Method: Incorporating Human Content Selection Variation in Summarization Evaluation. *ACM Transactions on Speech and Language Processing (TSLP)* 4(2).

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012a. An Assessment of the Accuracy of Automatic Evaluation in Summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*. Association for Computational Linguistics, Montréal, Canada, pages 1–9.

Karolina Owczarzak, Peter A. Rankel, Hoa Trang Dang, and John M. Conroy. 2012b. Assessing the Effect of Inconsistent Assessors on Summarization Evaluation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Jeju Island, Korea, pages 359–362.

Rebecca Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated Pyramid Scoring of Summaries using Distributional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sofia, Bulgaria, pages 143–147.

Maxime Peyrard and Judith Eckle-Kohler. 2016a. A General Optimization Framework for Multi-Document Summarization Using Genetic Algorithms and Swarm Intelligence. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 247 – 257.

Maxime Peyrard and Judith Eckle-Kohler. 2016b. Optimizing an Approximation of ROUGE - a Problem-Reduction Approach to Extractive Multi-Document Summarization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1825–1836.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, Disambiguate and Walk: A Unified Approach for Measuring Semantic Similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1341–1351.

Peter A. Rankel, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2013. A Decade of Automatic Content Evaluation of News Summaries: Reassessing the State of the Art. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sofia, Bulgaria, pages 131–136.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Large-margin Learning of Submodular Summarization Models. In *Proceedings*

of the 13th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, Avignon, France, pages 224–233.

Hiroya Takamura and Manabu Okumura. 2010. Learning to Generate Summary as Structured Output. In *Proceedings of the 19th ACM international Conference on Information and Knowledge Management*. Association for Computing Machinery, Toronto , ON, Canada, pages 1437–1440.

Qian Yang, Rebecca Passonneau, and Gerard de Melo. 2016. PEAK: Pyramid Evaluation via Automated Knowledge Extraction. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*. AAAI Press, Phoenix, AZ, USA.

Selective Encoding for Abstractive Sentence Summarization

Qingyu Zhou[†*] **Nan Yang**[‡] **Furu Wei**[‡] **Ming Zhou**[‡]
[†]Harbin Institute of Technology, Harbin, China
[‡]Microsoft Research, Beijing, China
qyzhou@hit.edu.cn {nanya,fuwei,mingzhou}@microsoft.com

Abstract

We propose a selective encoding model to extend the sequence-to-sequence framework for abstractive sentence summarization. It consists of a sentence encoder, a selective gate network, and an attention equipped decoder. The sentence encoder and decoder are built with recurrent neural networks. The selective gate network constructs a second level sentence representation by controlling the information flow from encoder to decoder. The second level representation is tailored for sentence summarization task, which leads to better performance. We evaluate our model on the English Gigaword, DUC 2004 and MSR abstractive sentence summarization datasets. The experimental results show that the proposed selective encoding model outperforms the state-of-the-art baseline models.

1 Introduction

Sentence summarization aims to shorten a given sentence and produce a brief summary of it. This is different from document level summarization task since it is hard to apply existing techniques in extractive methods, such as extracting sentence level features and ranking sentences. Early works propose using rule-based methods (Zajic et al., 2007), syntactic tree pruning methods (Knight and Marcu, 2002), statistical machine translation techniques (Banko et al., 2000) and so on for this task. We focus on abstractive sentence summarization task in this paper.

Recently, neural network models have been applied in this task. Rush et al. (2015) use auto-constructed sentence-headline pairs to train a neural network summarization model. They use a Convolutional Neural Network (CNN) encoder and feed-forward neural network language model decoder for this task. Chopra et al. (2016) extend their work by replacing the decoder with Recurrent Neural Network (RNN). Nallapati et al. (2016) follow this line and change the encoder to RNN to make it a full RNN based sequence-to-sequence model (Sutskever et al., 2014).

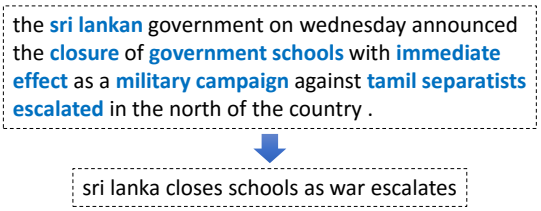


Figure 1: An abstractive sentence summarization system may produce the output summary by distilling the salient information from the highlight to generate a fluent sentence. We model the distilling process with selective encoding.

All the above works fall into the encoding-decoding paradigm, which first encodes the input sentence to an abstract representation and then decodes the intended output sentence based on the encoded information. As an extension of the encoding-decoding framework, attention-based approach (Bahdanau et al., 2015) has been broadly used: the encoder produces a list of vectors for all tokens in the input, and the decoder uses an attention mechanism to dynamically extract encoded information and align with the output tokens. This approach achieves huge success in tasks like machine translation, where alignment between all parts of the input and output are required. However, in abstractive sentence summarization, there is no explicit alignment relationship between the input sentence and the summary ex-

[*]Contribution during internship at Microsoft Research.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1095–1104
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1101

cept for the extracted common words. The challenge here is not to infer the alignment, but to select the highlights while filtering out secondary information in the input. A desired work-flow for abstractive sentence summarization is encoding, selection, and decoding. After selecting the important information from an encoded sentence, the decoder produces the output summary using the selected information. For example, in Figure 1, given the input sentence, the summarization system first selects the important information, and then rephrases or paraphrases to produce a well-organized summary. Although this is implicitly modeled in the encoding-decoding framework, we argue that abstractive sentence summarization shall benefit from explicitly modeling this selection process.

In this paper we propose **S**elective **E**ncoding for **A**bstractive **S**entence **S**ummarization (**SEASS**). We treat the sentence summarization as a three-phase task: encoding, selection, and decoding. It consists of a sentence encoder, a selective gate network, and a summary decoder. First, the sentence encoder reads the input words through an RNN unit to construct the first level sentence representation. Then the selective gate network selects the encoded information to construct the second level sentence representation. The selective mechanism controls the information flow from encoder to decoder by applying a gate network according to the sentence information, which helps improve encoding effectiveness and release the burden of the decoder. Finally, the attention-equipped decoder generates the summary using the second level sentence representation. We conduct experiments on English Gigaword, DUC 2004 and Microsoft Research Abstractive Text Compression test sets. Our SEASS model achieves 17.54 ROUGE-2 F1, 9.56 ROUGE-2 recall and 10.63 ROUGE-2 F1 on these test sets respectively, which improves performance compared to the state-of-the-art methods.

2 Related Work

Abstractive sentence summarization, also known as sentence compression and similar to headline generation, is used to help compress or fuse the selected sentences in extractive document summarization systems since they may inadvertently include unnecessary information. The sentence summarization task has been long connected to the headline generation task. There are some previous methods to solve this task, such as the linguistic rule-based method (Dorr et al., 2003). As for the statistical machine learning based methods, Banko et al. (2000) apply statistical machine translation techniques by modeling headline generation as a translation task and use 8000 article-headline pairs to train the system.

Rush et al. (2015) propose leveraging news data in Annotated English Gigaword (Napoles et al., 2012) corpus to construct large scale parallel data for sentence summarization task. They propose an ABS model, which consists of an attentive Convolutional Neural Network encoder and an neural network language model (Bengio et al., 2003) decoder. On this Gigaword test set and DUC 2004 test set, the ABS model produces the state-of-the-art results. Chopra et al. (2016) extend this work, which keeps the CNN encoder but replaces the decoder with recurrent neural networks. Their experiments showes that the CNN encoder with RNN decoder model performs better than Rush et al. (2015). Nallapati et al. (2016) further change the encoder to an RNN encoder, which leads to a full RNN sequence-to-sequence model. Besides, they enrich the encoder with lexical and statistic features which play important roles in traditional feature based summarization systems, such as NER and POS tags, to improve performance. Experiments on the Gigaword and DUC 2004 test sets show that the above models achieve state-of-the-art results.

Gu et al. (2016) and Gulcehre et al. (2016) come up similar ideas that summarization task can benefit from copying words from input sentences. Gu et al. (2016) propose CopyNet to model the copying action in response generation, which also applies for summarization task. Gulcehre et al. (2016) propose a switch gate to control whether to copy from source or generate from decoder vocabulary. Zeng et al. (2016) also propose using copy mechanism and add a scalar weight on the gate of GRU/LSTM for this task. Cheng and Lapata (2016) use an RNN based encoder-decoder for extractive summarization of documents.

Yu et al. (2016) propose a segment to segment neural transduction model for sequence-to-sequence framework. The model introduces a latent segmentation which determines correspondences between tokens of the input sequence and the output sequence. Experiments on this task show that the proposed transduction model per-

forms comparable to the ABS model. Shen et al. (2016) propose to apply Minimum Risk Training (MRT) in neural machine translation to directly optimize the evaluation metrics. Ayana et al. (2016) apply MRT on abstractive sentence summarization task and the results show that optimizing for ROUGE improves the test performance.

3 Problem Formulation

For sentence summarization, given an input sentence $x = (x_1, x_2, \ldots, x_n)$, where n is the sentence length, $x_i \in \mathcal{V}_s$ and $\mathcal{V}_s$ is the source vocabulary, the system summarizes x by producing $y = (y_1, y_2, \ldots, y_l)$, where $l \leq n$ is the summary length , $y_i \in \mathcal{V}_t$ and $\mathcal{V}_t$ is the target vocabulary.

If $|y| \subseteq |x|$, which means all words in summary y must appear in given input, we denote this as *extractive* sentence summarization. If $|y| \nsubseteq |x|$, which means not all words in summary come from input sentence, we denote this as *abstractive* sentence summarization. Table 1 provides an example. We focus on *abstracive* sentence summarization task in this paper.

Input:	South Korean President Kim Young-Sam left here Wednesday on a week - long state visit to Russia and Uzbekistan for talks on North Korea 's nuclear confrontation and ways to strengthen bilateral ties .
Output:	Kim leaves for Russia for talks on NKorea nuclear standoff

Table 1: An abstractive sentence summarization example.

4 Model

As shown in Figure 2, our model consists of a sentence encoder using the Gated Recurrent Unit (GRU) (Cho et al., 2014), a selective gate network and an attention-equipped GRU decoder. First, the bidirectional GRU encoder reads the input words $x = (x_1, x_2, \ldots, x_n)$ and builds its representation $(h_1, h_2, \ldots, h_n)$. Then the selective gate selects and filters the word representations according to the sentence meaning representation to produce a tailored sentence word representation for abstractive sentence summarization task. Lastly, the GRU decoder produces the output summary with attention to the tailored representation. In the following sections, we introduce the sentence encoder, the selective mechanism, and the summary decoder respectively.

4.1 Sentence Encoder

The role of the sentence encoder is to read the input sentence and construct the basic sentence representation. Here we employ a bidirectional GRU (BiGRU) as the recurrent unit, where GRU is defined as:

$$z_i = \sigma(\mathbf{W}_z[x_i, h_{i-1}]) \quad (1)$$

$$r_i = \sigma(\mathbf{W}_r[x_i, h_{i-1}]) \quad (2)$$

$$\widetilde{h}_i = \tanh(\mathbf{W}_h[x_i, r_i \odot h_{i-1}]) \quad (3)$$

$$h_i = (1 - z_i) \odot h_{i-1} + z_i \odot \widetilde{h}_i \quad (4)$$

where $\mathbf{W}_z$, $\mathbf{W}_r$ and $\mathbf{W}_h$ are weight matrices.

The BiGRU consists of a forward GRU and a backward GRU. The forward GRU reads the input sentence word embeddings from left to right and gets a sequence of hidden states, $(\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_n)$. The backward GRU reads the input sentence embeddings reversely, from right to left, and results in another sequence of hidden states, $(\overleftarrow{h}_1, \overleftarrow{h}_2, \ldots, \overleftarrow{h}_n)$:

$$\vec{h}_i = \text{GRU}(x_i, \vec{h}_{i-1}) \quad (5)$$

$$\overleftarrow{h}_i = \text{GRU}(x_i, \overleftarrow{h}_{i+1}) \quad (6)$$

The initial states of the BiGRU are set to zero vectors, i.e., $\vec{h}_1 = 0$ and $\overleftarrow{h}_n = 0$. After reading the sentence, the forward and backward hidden states are concatenated, i.e., $h_i = [\vec{h}_i; \overleftarrow{h}_i]$, to get the basic sentence representation.

4.2 Selective Mechanism

In the sequence-to-sequence machine translation (MT) model, the encoder and decoder are responsible for mapping input sentence information to a list of vectors and decoding the sentence representation vectors to generate an output sentence (Bahdanau et al., 2015). Some previous works apply this framework to summarization generation tasks (Nallapati et al., 2016; Gu et al., 2016; Gulcehre et al., 2016). However, abstractive sentence summarization is different from MT in two ways. First, there is no explicit alignment relationship between the input sentence and the output summary except for the common words. Second, summarization task needs to keep the highlights and remove the unnecessary information, while MT needs to keep all information literally.

Herein, we propose a selective mechanism to model the selection process for abstractive sentence summarization. The selective mechanism

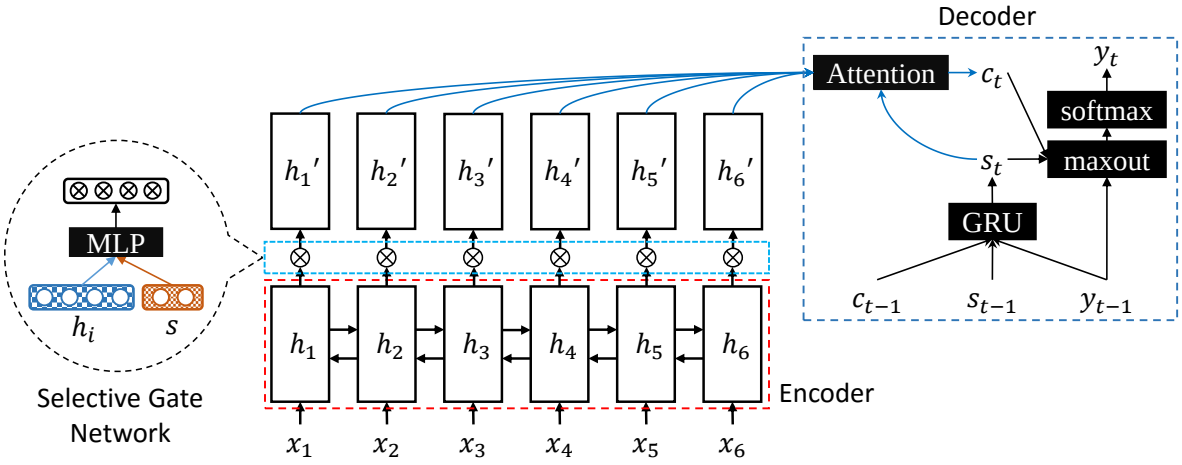


Figure 2: Overview of the **S**elective **E**ncoding for **A**bstractive **S**entence **S**ummarization (**SEASS**).

extends the sequence-to-sequence model by constructing a tailored representation for abstractive sentence summarization task. Concretely, the selective gate network in our model takes two vector inputs, the sentence word vector h_i and the sentence representation vector s. The sentence word vector h_i is the output of the BiGRU encoder and represents the meaning and context information of word x_i. The sentence vector s is used to represent the meaning of the sentence. For each word x_i, the selective gate network generates a gate vector $sGate_i$ using h_i and s, then the tailored representation is constructed, i.e., h'_i.

In detail, we concatenate the last forward hidden state $\vec{h}_n$ and backward hidden state $\overleftarrow{h}_1$ as the sentence representation s:

$$s = \begin{bmatrix} \overleftarrow{h}_1 \\ \vec{h}_n \end{bmatrix} \tag{7}$$

For each time step i, the selective gate takes the sentence representation s and BiGRU hidden h_i as inputs to compute the gate vector $sGate_i$:

$$sGate_i = \sigma(\mathbf{W}_s h_i + \mathbf{U}_s s + b) \tag{8}$$

$$h'_i = h_i \odot sGate_i \tag{9}$$

where $\mathbf{W}_s$ and $\mathbf{U}_s$ are weight matrices, b is the bias vector, σ denotes sigmoid activation function, and $\odot$ is element-wise multiplication. After the selective gate network, we obtain another sequence of vectors $(h'_1, h'_2, \ldots, h'_n)$. This new sequence is then used as the input sentence representation for the decoder to generate the summary.

4.3 Summary Decoder

On top of the sentence encoder and the selective gate network, we use GRU with attention as the decoder to produce the output summary.

At each decoding time step t, the GRU reads the previous word embedding w_{t-1} and previous context vector c_{t-1} as inputs to compute the new hidden state s_t. To initialize the GRU hidden state, we use a linear layer with the last backward encoder hidden state $\overleftarrow{h}_1$ as input:

$$s_t = \text{GRU}(w_{t-1}, c_{t-1}, s_{t-1}) \tag{10}$$

$$s_0 = \tanh(\mathbf{W}_d \overleftarrow{h}_1 + b) \tag{11}$$

where $\mathbf{W}_d$ is the weight matrix and b is the bias vector.

The context vector c_t for current time step t is computed through the concatenate attention mechanism (Luong et al., 2015), which matches the current decoder state s_t with each encoder hidden state h'_i to get an importance score. The importance scores are then normalized to get the current context vector by weighted sum:

$$e_{t,i} = v_a^\top \tanh(\mathbf{W}_a s_{t-1} + \mathbf{U}_a h'_i) \tag{12}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{i=1}^{n} \exp(e_{t,i})} \tag{13}$$

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h'_i \tag{14}$$

We then combine the previous word embedding w_{t-1}, the current context vector c_t, and the decoder state s_t to construct the readout state r_t. The readout state is then passed through a maxout hidden layer (Goodfellow et al., 2013) to predict the

next word with a softmax layer over the decoder vocabulary.

$$r_t = \mathbf{W}_r w_{t-1} + \mathbf{U}_r c_t + \mathbf{V}_r s_t \quad (15)$$

$$m_t = [\max\{r_{t,2j-1}, r_{t,2j}\}]^\top_{j=1,\dots,d} \quad (16)$$

$$p(y_t|y_1, \dots, y_{t-1}) = \text{softmax}(\mathbf{W}_o m_t) \quad (17)$$

where $\mathbf{W}_a$, $\mathbf{U}_a$, $\mathbf{W}_r$, $\mathbf{U}_r$, $\mathbf{V}_r$ and $\mathbf{W}_o$ are weight matrices. Readout state r_t is a $2d$-dimensional vector, and the maxout layer (Equation 16) picks the max value for every two numbers in r_t and produces a d-dimensional vector m_t.

4.4 Objective Function

Our goal is to maximize the output summary probability given the input sentence. Therefore, we optimize the negative log-likelihood loss function:

$$J(\theta) = -\frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \log p(y|x) \quad (18)$$

where $\mathcal{D}$ denotes a set of parallel sentence-summary pairs and θ is the model parameter. We use Stochastic Gradient Descent (SGD) with mini-batch to learn the model parameter θ.

5 Experiments

In this section we introduce the dataset we use, the evaluation metric, the implementation details, the baselines we compare to, and the performance of our system.

5.1 Dataset

Training Set For our training set, we use a parallel corpus which is constructed from the Annotated English Gigaword dataset (Napoles et al., 2012) as mentioned in Rush et al. (2015). The parallel corpus is produced by pairing the first sentence and the headline in the news article with some heuristic rules. We use the script[1] released by Rush et al. (2015) to pre-process and extract the training and development datasets. The script performs various basic text normalization, including PTB tokenization, lower-casing, replacing all digit characters with #, and replacing word types seen less than 5 times with $\langle unk \rangle$. The extracted corpus contains about 3.8M sentence-summary pairs for the training set and 189K examples for the development set.

For our test set, we use the English Gigaword, DUC 2004, and Microsoft Research Abstractive Text Compression test sets.

[1] https://github.com/facebook/NAMAS

English Gigaword Test Set We randomly sample 8000 pairs from the extracted development set as our development set since it is relatively large. For the test set, we use the same randomly held-out test set of 2000 sentence-summary pairs as Rush et al. (2015).[2]

We also find that except for the empty titles, this test set has some invalid lines like the input sentence containing only one word. Therefore, we further sample 2000 pairs as our internal test set and release it for future works[3].

DUC 2004 Test Set We employ DUC 2004 data for tasks 1 & 2 (Over et al., 2007) in our experiments as one of the test sets since it is too small to train a neural network model on. The dataset pairs each document with 4 different human-written reference summaries which are capped at 75 bytes. It has 500 input sentences with each sentence paired with 4 summaries.

MSR-ATC Test Set Toutanova et al. (2016) release a new dataset for sentence summarization task by crowdsourcing. This dataset contains approximately 6,000 source text sentences with multiple manually-created summaries (about 26,000 sentence-summary pairs in total). Toutanova et al. (2016) provide a standard split of the data into training, development, and test sets, with 4,936, 448 and 785 input sentences respectively. Since the training set is too small, we only use the test set as one of our test sets. We denote this dataset as MSR-ATC (Microsoft Research Abstractive Text Compression) test set in the following.

Table 2 summarizes the statistic information of the three datasets we used.

5.2 Evaluation Metric

We employ ROUGE (Lin, 2004) as our evaluation metric. ROUGE measures the quality of summary by computing overlapping lexical units, such as unigram, bigram, trigram, and longest common subsequence (LCS). It becomes the standard evaluation metric for DUC shared tasks and popular for summarization evaluation. Following previous work, we use ROUGE-1 (unigram), ROUGE-2 (bi-

[2] Thanks to Rush et al. (2015), we acquired the test set they used. Following Chopra et al. (2016), we remove pairs with empty titles resulting in slightly different accuracy compared to Rush et al. (2015) for their systems. The cleaned test set contains 1951 sentence-summary pairs.

[3] Our development and test sets can be found at https://res.qyzhou.me

Data Set	Giga	DUC†	MSR†
#(sent)	3.99M	500	785
#(sentWord)	125M	17.8K	29K
#(summWord)	33M	20.9K	85.9K
#(ref)	1	4	3-5
AvgInputLen	31.35	35.56	36.97
AvgSummLen	8.23	10.43	25.5

Table 2: Data statistics for the English Gigaword, DUC 2004 and MSR-ATC datasets. #(x) denotes the number of x, e.g., #(ref) is the number of reference summaries of an input sentence. AvgInputLen is the average input sentence length and AvgSummLen is the average summary length. †DUC 2004 and MSR-ATC datasets are for test purpose only.

gram) and ROUGE-L (LCS) as the evaluation metrics in the reported experimental results.

5.3 Implementation Details

Model Parameters The input and output vocabularies are collected from the training data, which have 119,504 and 68,883 word types respectively. We set the word embedding size to 300 and all GRU hidden state sizes to 512. We use dropout (Srivastava et al., 2014) with probability $p = 0.5$.

Model Training We initialize model parameters randomly using a Gaussian distribution with Xavier scheme (Glorot and Bengio, 2010). We use Adam (Kingma and Ba, 2015) as our optimizing algorithm. For the hyperparameters of Adam optimizer, we set the learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 10^{-8}$. During training, we test the model performance (ROUGE-2 F1) on development set for every 2,000 batches. We halve the Adam learning rate α if the ROUGE-2 F1 score drops for twelve consecutive tests on development set. We also apply gradient clipping (Pascanu et al., 2013) with range $[-5, 5]$ during training. To both speed up the training and converge quickly, we use mini-batch size 64 by grid search.

Beam Search We use beam search to generate multiple summary candidates to get better results. To avoid favoring shorter outputs, we average the ranking score along the beam path by dividing it by the number of generated words. To both decode fast and get better results, we set the beam size to 12 in our experiments.

5.4 Baseline

We compare SEASS model with the following state-of-the-art baselines:

ABS Rush et al. (2015) use an attentive CNN encoder and NNLM decoder to do the sentence summarization task. We trained this baseline model with the released code[1] and evaluate it with our internal English Gigaword test set and MSR-ATC test set.

ABS+ Based on ABS model, Rush et al. (2015) further tune their model using DUC 2003 dataset, which leads to improvements on DUC 2004 test set.

CAs2s As an extension of the ABS model, Chopra et al. (2016) use a convolutional attention-based encoder and RNN decoder, which outperforms the ABS model.

Feats2s Nallapati et al. (2016) use a full RNN sequence-to-sequence encoder-decoder model and add some features to enhance the encoder, such as POS tag, NER, and so on.

Luong-NMT Neural machine translation model of Luong et al. (2015) with two-layer LSTMs for the encoder-decoder with 500 hidden units in each layer implemented in (Chopra et al., 2016).

s2s+att We also implement a sequence-to-sequence model with attention as our baseline and denote it as "s2s+att".

5.5 Results

We report ROUGE F1, ROUGE recall and ROUGE F1 for English Gigaword, DUC 2004 and MSR-ATC test sets respectively. We use the official ROUGE script (version 1.5.5) [4] to evaluate the summarization quality in our experiments. For English Gigaword[5] and MSR-ATC[6] test sets, the outputs have different lengths so we evaluate the system with F1 metric. As for the DUC 2004 test set[7], the task requires the system to produce a fixed length summary (75 bytes), therefore we employ ROUGE recall as the evaluation metric. To satisfy the length requirement, we decode the output summary to a roughly expected length following Rush et al. (2015).

[4] http://www.berouge.com/

[5] The ROUGE evaluation option is the same as Rush et al. (2015), -m -n 2 -w 1.2

[6] The ROUGE evaluation option is, -m -n 2 -w 1.2

[7] The ROUGE evaluation option is, -m -b 75 -n 2 -w 1.2

English Gigaword We acquire the test set from Rush et al. (2015) so we can make fair comparisons to the baselines.

Models	RG-1	RG-2	RG-L
ABS (beam)‡	29.55-	11.32-	26.42-
ABS+ (beam)‡	29.76-	11.88-	26.96-
Feats2s (beam)‡	32.67-	15.59-	30.64-
CAs2s (greedy)‡	33.10-	14.45-	30.25-
CAs2s (beam)‡	33.78-	15.97-	31.15-
Luong-NMT (beam)‡	33.10-	14.45-	30.71-
s2s+att (greedy)	33.18-	14.79-	30.80-
s2s+att (beam)	34.04-	15.95-	31.68-
SEASS (greedy)	35.48	16.50	32.93
SEASS (beam)	**36.15**	**17.54**	**33.63**

Table 3: Full length ROUGE F1 evaluation results on the English Gigaword test set used by Rush et al. (2015). RG in the Table denotes ROUGE. Results with ‡ mark are taken from the corresponding papers. The superscript - indicates that our SEASS model with beam search performs significantly better than it as given by the 95% confidence interval in the official ROUGE script.

Models	RG-1	RG-2	RG-L
ABS (beam)	37.41-	15.87-	34.70-
s2s+att (greedy)	42.41-	20.76-	39.84-
s2s+att (beam)	43.76-	22.28-	41.14-
SEASS (greedy)	45.27	22.88	42.20
SEASS (beam)	**46.86**	**24.58**	**43.53**

Table 4: Full length ROUGE F1 evaluation on our internal English Gigaword test data. The superscript - indicates that our SEASS model performs significantly better than it as given by the 95% confidence interval in the official ROUGE script.

In Table 3, we report the ROUGE F1 score of our model and the baseline methods. Our SEASS model with beam search outperforms all baseline models by a large margin. Even for greedy search, our model still performs better than other methods which used beam search. For the popular ROUGE-2 metric, our SEASS model achieves 17.54 F1 score and performs better than the previous works. Compared to the ABS model, our model has a 6.22 ROUGE-2 F1 relative gain. Compared to the highest CAs2s baseline, our model achieves 1.57 ROUGE-2 F1 improvement and passes the significant test according to the official ROUGE script.

Table 4 summarizes our results on our internal test set using ROUGE F1 evaluation metrics. The performance on our internal test set is comparable to our development set, which achieves 24.58 ROUGE-2 F1 and outperforms the baselines.

DUC 2004 We evaluate our model using the ROUGE recall score since the reference summaries of the DUC 2004 test set are capped at 75 bytes. Therefore, we decode the summary to a fixed length 18 to ensure that the generated summary satisfies the minimum length requirement. As summarized in Table 5, our SEASS outperforms all the baseline methods and achieves 29.21, 9.56 and 25.51 for ROUGE 1, 2 and L recall. Compared to the ABS+ model which is tuned using DUC 2003 data, our model performs significantly better by 1.07 ROUGE-2 recall score and is trained only with English Gigaword sentence-summary data without being tuned using DUC data.

Models	RG-1	RG-2	RG-L
ABS (beam)‡	26.55-	7.06-	22.05-
ABS+ (beam)‡	28.18-	8.49-	23.81-
Feats2s (beam)‡	28.35-	9.46	24.59-
CAs2s (greedy)‡	29.13	7.62-	23.92-
CAs2s (beam)‡	28.97	8.26-	24.06-
Luong-NMT (beam)‡	28.55	8.79-	24.43-
s2s+att (greedy)	27.03-	7.89-	23.80-
s2s+att (beam)	28.13	9.25	24.76
SEASS (greedy)	28.68	8.55	25.04
SEASS (beam)	**29.21**	**9.56**	**25.51**

Table 5: ROUGE recall evaluation results on DUC 2004 test set. All these models are tested using beam search. Results with ‡ mark are taken from the corresponding papers. The superscript - indicates that our SEASS model performs significantly better than it as given by the 95% confidence interval in the official ROUGE script.

MSR-ATC We report the full length ROUGE F1 score on the MSR-ATC test set in Table 6. To the best of our knowledge, this is the first work that reports ROUGE metric scores on the MSR-ATC dataset. Note that we only compare our model with ABS since the others are not publicly available. Our SEASS achieves 10.63 ROUGE-2 F1 and outperforms the s2s+att baseline by 1.02 points.

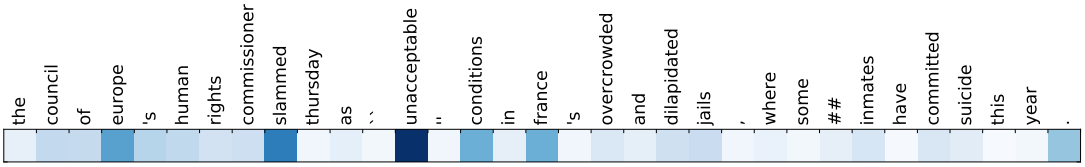


Figure 3: First derivative heat map of the output with respect to the selective gate. The important words are selected in the input sentence, such as "europe", "slammed" and "unacceptable". The output summary of our system is "council of europe slams french prison conditions" and the true summary is "council of europe again slams french prison conditions".

Models	RG-1	RG-2	RG-L
ABS (beam)	20.27$^-$	5.26$^-$	17.10$^-$
s2s+att (greedy)	15.15$^-$	4.48$^-$	13.62$^-$
s2s+att (beam)	22.65$^-$	9.61$^-$	21.39$^-$
SEASS (greedy)	19.77	6.44	17.36
SEASS (beam)	**25.75**	**10.63**	**22.90**

Table 6: Full length ROUGE F1 evaluation on MSR-ATC test set. Beam search are used in both the baselines and our method. The superscript $^-$ indicates that our SEASS model performs significantly better than it as given by the 95% confidence interval in the official ROUGE script.

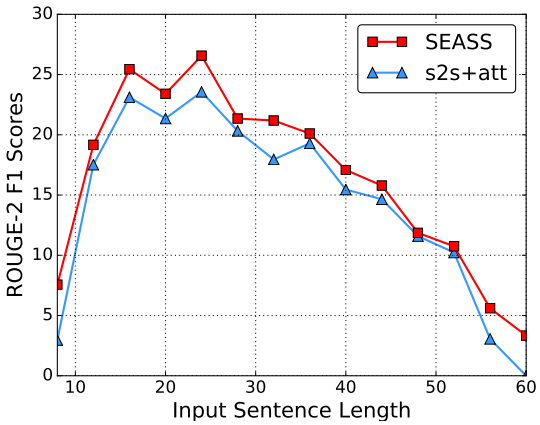


Figure 4: ROUGE-2 F1 score on different groups of input sentences in terms of their length for s2s+att baseline and our SEASS model on English Gigaword test sets.

6 Discussion

In this section, we first compare the performance of SEASS with the s2s+att baseline model to illustrate that the proposed method succeeds in selecting information and building tailored representation for abstractive sentence summarization. We then analyze selective encoding by visualizing the heat map.

Effectiveness of Selective Encoding We further test the SEASS model with different sentence lengths on English Gigaword test sets, which are merged from the Rush et al. (2015) test set and our internal test set. The length of sentences in the test sets ranges from 10 to 80. We group the sentences with an interval of 4 and get 18 different groups and we draw the first 14 groups. We find that the performance curve of our SEASS model always appears to be on the top of that of s2s+att with a certain margin. For the groups of 16, 20, 24, 32, 56 and 60, the SEASS model obtains big improvements compared to the s2s+att model. Overall, these improvements on all groups indicate that the selective encoding method benefits the abstractive sentence summarization task.

Saliency Heat Map of Selective Gate Since the output of the selective gate network is a high dimensional vector, it is hard to visualize all the gate values. We use the method in Li et al. (2016) to visualize the contribution of the selective gate to the final output, which can be approximated by the first derivative. Given sentence words x with associated output summary y, the trained model associates the pair (x, y) with a score $S_y(x)$. The goal is to decide which gate g associated with a specific word makes the most significant contribution to $S_y(x)$. We approximate the $S_y(g)$ by computing the first-order Taylor expansion since the score $S_y(x)$ is a highly non-linear function in the deep neural network models:

$$S_y(g) \approx w(g)^T g + b \tag{19}$$

where $w(g)$ is first the derivative of S_y with respect to the gate g:

$$w(g) = \frac{\partial(S_y)}{\partial g}|_g \tag{20}$$

We then draw the Euclidean norm of the first derivative of the output y with respect to the selective gate g associated with each input words.

Figure 3 shows an example of the first derivative heat map, in which most of the important words are selected by the selective gate such as "europe", "slammed", "unacceptable", "conditions", and "france". We can observe that the selective gate determines the importance of each word before decoder, which releases the burden of it by providing tailored sentence encoding.

7 Conclusion

This paper proposes a selective encoding model which extends the sequence-to-sequence model for abstractive sentence summarization task. The selective mechanism mimics one of the human summarizers' behaviors, selecting important information before writing down the summary. With the proposed selective mechanism, we build an end-to-end neural network summarization model which consists of three phases: encoding, selection, and decoding. Experimental results show that the selective encoding model greatly improves the performance with respect to the state-of-the-art methods on English Gigaword, DUC 2004 and MSR-ATC test sets.

Acknowledgments

We thank Chuanqi Tan, Junwei Bao, Shuangzhi Wu and the anonymous reviewers for their helpful comments. We also thank Alexander M. Rush for providing the dataset for comparison and helpful discussions.

References

Ayana, Shiqi Shen, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with minimum risk training. *CoRR* abs/1604.01904.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of 3rd International Conference for Learning Representations*. San Diego.

Michele Banko, Vibhu O Mittal, and Michael J Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 318–325.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *journal of machine learning research* 3(Feb):1137–1155.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 484–494.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 93–98.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*. Association for Computational Linguistics, pages 1–8.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*. volume 9, pages 249–256.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. 2013. Maxout networks. *ICML (3)* 28:1319–1327.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1631–1640.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 140–149.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of 3rd International Conference for Learning Representations*. San Diego.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence* 139(1):91–107.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 681–691.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, volume 8.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421.

Ramesh Nallapati, Bowen Zhou, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, Stroudsburg, PA, USA, AKBC-WEKEX '12, pages 95–100.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management* 43(6):1506–1520.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28:1310–1318.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 379–389.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1683–1692.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 340–350.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1307–1316.

David Zajic, Bonnie J Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing & Management* 43(6):1549–1570.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382* .

PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents

Corina Florescu and **Cornelia Caragea**
Computer Science and Engineering
University of North Texas, USA
CorinaFlorescu@my.unt.edu, ccaragea@unt.edu

Abstract

The large and growing amounts of online scholarly data present both challenges and opportunities to enhance knowledge discovery. One such challenge is to automatically extract a small set of keyphrases from a document that can accurately describe the document's content and can facilitate fast information processing. In this paper, we propose PositionRank, an unsupervised model for keyphrase extraction from scholarly documents that incorporates information from all positions of a word's occurrences into a biased PageRank. Our model obtains remarkable improvements in performance over PageRank models that do not take into account word positions as well as over strong baselines for this task. Specifically, on several datasets of research papers, PositionRank achieves improvements as high as 29.09%.

1 Introduction

The current Scholarly Web contains many millions of scientific documents. For example, Google Scholar is estimated to have more than 100 million documents. On one hand, these rapidly-growing scholarly document collections offer benefits for knowledge discovery, and on the other hand, finding useful information has become very challenging. Keyphrases associated with a document typically provide a high-level topic description of the document and can allow for efficient information processing. In addition, keyphrases are shown to be rich sources of information in many natural language processing and information retrieval tasks such as scientific paper summarization, classification, recommendation, clustering, and search (Abu-Jbara and Radev, 2011; Qazvinian et al., 2010; Jones and Staveley, 1999; Zha, 2002; Zhang et al., 2004; Hammouda et al., 2005). Due to their importance, many approaches to keyphrase extraction have been proposed in the literature along two lines of research: supervised and unsupervised (Hasan and Ng, 2014, 2010).

In the supervised line of research, keyphrase extraction is formulated as a binary classification problem, where candidate phrases are classified as either positive (i.e., keyphrases) or negative (i.e., non-keyphrases) (Frank et al., 1999; Hulth, 2003). Various feature sets and classification algorithms yield different extraction systems. For example, Frank et al. (1999) developed a system that extracts two features for each candidate phrase, i.e., the *tf-idf* of the phrase and its distance from the beginning of the target document, and uses them as input to Naïve Bayes classifiers. Although supervised approaches typically perform better than unsupervised approaches (Kim et al., 2013), the requirement for large human-annotated corpora for each field of study has led to significant attention towards the design of unsupervised approaches.

In the unsupervised line of research, keyphrase extraction is formulated as a ranking problem with graph-based ranking techniques being considered state-of-the-art (Hasan and Ng, 2014). These graph-based techniques construct a word graph from each target document, such that nodes correspond to words and edges correspond to word association patterns. Nodes are then ranked using graph centrality measures such as PageRank (Mihalcea and Tarau, 2004; Liu et al., 2010) or HITS (Litvak and Last, 2008), and the top ranked phrases are returned as keyphrases. Since their introduction, many graph-based extensions have been proposed, which aim at modeling various types of information. For example, Wan and Xiao (2008) proposed a model that incorporates a local

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1105–1115
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1102

Factorizing Personalized **Markov Chains** for Next-**Basket Recommendation**
by Steffen Rendle, Christoph Freudenthaler and Lars Schmidt-Thieme

Recommender systems are an important component of many websites. Two of the most popular approaches are based on **matrix factorization** (MF) and **Markov chains** (MC). MF methods learn the general taste of a user by factorizing the matrix over observed user-item preferences. [...] we present a method bringing both approaches together. Our method is based on personalized transition graphs over underlying **Markov chains**. [...] our factorized personalized MC (FPMC) model subsumes both a common **Markov chain** and the normal **matrix factorization** model. [...] we introduce an adaption of the Bayesian Personalized Ranking (BPR) framework for sequential basket data. [...]

Author-input keyphrases: *Basket Recommendation, Markov Chain, Matrix Factorization*

Figure 1: The title and abstract of a WWW paper by Rendle et al. (2010) and the author-input keyphrases for the paper. Red bold phrases represent the gold-standard keyphrases for the document.

neighborhood of the target document corresponding to its textually-similar documents, computed using the cosine similarity between the *tf-idf* vectors of documents. Liu et al. (2010) assumed a mixture of topics over documents and proposed to use topic models to decompose these topics in order to select keyphrases from all major topics. Keyphrases are then ranked by aggregating the topic-specific scores obtained from several topic-biased PageRanks. We posit that other information can be leveraged that has the potential to improve unsupervised keyphrase extraction.

For example, in a scholarly domain, keyphrases generally occur on positions very close to the beginning of a document and occur frequently. Figure 1 shows an anecdotal example illustrating this behavior using the 2010 best paper award winner in the World Wide Web conference. The author input keyphrases are marked with red bold in the figure. Notice in this example the high frequency of the keyphrase "Markov chain" that occurs very early in the document (even from its title). Hence, *can we design an effective unsupervised approach to keyphrase extraction by jointly exploiting words' position information and their frequency in documents?* We specifically address this question using *research papers* as a case study. The result of this extraction task will aid indexing of documents in digital libraries, and hence, will lead to improved organization, search, retrieval, and recommendation of scientific documents. The importance of keyphrase extraction from research papers is also emphasized by the SemEval Shared Tasks on this topic from 2017[1] and 2010 (Kim et al., 2010). Our contributions are as follows:

- We propose an unsupervised graph-based model, called PositionRank, that incorporates information from all positions of a word's occurrences into a biased PageRank to score keywords that are later used to score and rank keyphrases in research papers.
- We show that PositionRank that aggregates information from all positions of a word's occurrences performs better than a model that uses only the first position of a word.
- We experimentally evaluate PositionRank on three datasets of research papers and show statistically significant improvements over PageRank-based models that do not take into account word positions, as well as over strong baselines for keyphrase extraction.

The rest of the paper is organized as follows. We summarize related work in the next section. PositionRank is described in Section 3. We then present the datasets of research papers, and our experiments and results in Section 4. Finally, we conclude the paper in Section 5.

2 Related Work

Many supervised and unsupervised approaches to keyphrase extraction have been proposed in the literature (Hasan and Ng, 2014).

Supervised approaches use annotated documents with "correct" keyphrases to train classifiers for discriminating keyphrases from non-keyphrases for a document. KEA (Frank et al., 1999) and GenEx (Turney, 2000) are two representative supervised approaches with the most important features being the frequency and the position of a phrase in a target document. Hulth

[1] http://alt.qcri.org/semeval2017/task10/

(2003) used a combination of lexical and syntactic features such as the collection frequency and the part-of-speech tag of a phrase in conjunction with a bagging technique. Nguyen and Kan (2007) extended KEA to include features such as the distribution of candidate phrases in different sections of a research paper, and the acronym status of a phrase. In a different work, Medelyan et al. (2009) extended KEA to integrate information from Wikipedia. Lopez and Romary (2010) used bagged decision trees learned from a combination of features including structural features (e.g., the presence of a phrase in particular sections of a document) and lexical features (e.g., the presence of a candidate phrase in WordNet or Wikipedia). Chuang et al. (2012) proposed a model that incorporates a set of statistical and linguistic features (e.g., *tf-idf*, BM25, part-of-speech filters) for identifying descriptive terms in a text. Caragea et al. (2014a) designed features based on information available in a document network (such as a citation network) and used them with traditional features in a supervised framework.

In unsupervised approaches, various measures such as *tf-idf* and topic proportions are used to score words, which are later aggregated to obtain scores for phrases (Barker and Cornacchia, 2000; Zhang et al., 2007; Liu et al., 2009). The ranking based on *tf-idf* has been shown to work well in practice (Hasan and Ng, 2014, 2010), despite its simplicity. Graph-based ranking methods and centrality measures are considered state-of-the-art for unsupervised keyphrase extraction. Mihalcea and Tarau (2004) proposed TextRank for scoring keyphrases by applying PageRank on a word graph built from adjacent words within a document. Wan and Xiao (2008) extended TextRank to SingleRank by adding weighted edges between words that co-occur in a window of variable size $w \geq 2$. Textually-similar neighboring documents are included in ExpandRank (Wan and Xiao, 2008) to compute more accurate word co-occurrence information. Gollapalli and Caragea (2014) extended ExpandRank to integrate information from citation networks where papers cite one another.

Lahiri et al. (2014) extracted keyphrases from documents using various centrality measures such as node degree, clustering coefficient and closeness. Martinez-Romo et al. (2016) used information from WordNet to enrich the semantic relationships between the words in the graph.

Several unsupervised approaches leverage word clustering techniques such as first grouping candidate words into topics and then, extracting one representative keyphrase from each topic (Liu et al., 2009; Bougouin et al., 2013). Liu et al. (2010) extended topic-biased PageRank (Haveliwala, 2003) to kephrase extraction. In particular, they decomposed a document into multiple topics, using topic models, and applied a separate topic-biased PageRank for each topic. The PageRank scores from each topic were then combined into a single score, using as weights the topic proportions returned by topic models for the document.

The best performing keyphrase extraction system in SemEval 2010 (El-Beltagy and Rafea, 2010) used statistical observations such as term frequencies to filter out phrases that are unlikely to be keyphrases. More precisely, thresholding on the frequency of phrases is applied, where the thresholds are estimated from the data. The candidate phrases are then ranked using the *tf-idf* model in conjunction with a boosting factor which aims at reducing the bias towards single word terms. Danesh et al. (2015) computed an initial weight for each phrase based on a combination of statistical heuristics such as the *tf-idf* score and the first position of a phrase in a document. Phrases and their initial weights are then incorporated into a graph-based algorithm which produces the final ranking of keyphrase candidates. Le et al. (2016) showed that the extraction of keyphrases from a document can benefit from considering candidate phrases with part of speech tags other than nouns or adjectives. Adar and Datta (2015) extracted keyphrases by mining abbreviations from scientific literature and built a semantically hierarchical keyphrase database. Word embedding vectors were also employed to measure the relatedness between words in graph based models (Wang et al., 2014). Many of the above approaches, both supervised and unsupervised, are compared and analyzed in the ACL survey on keyphrase extraction by Hasan and Ng (2014).

In contrast to the above approaches, we propose PositionRank, aimed at capturing both highly frequent words or phrases and their position in a document. Despite that the *relative position* of a word in a document is shown to be a very effective feature in supervised keyphrase extraction (Hulth, 2003; Zhang et al., 2007), to our knowledge, the position information has not been used before in unsupervised methods. The strong contribution of

this paper is the design of a position-biased PageRank model that successfully incorporates all positions of a word's occurrences, which is different from supervised models that use only the first position of a word. Our model assigns higher probabilities to words found early on in a document instead of using a uniform distribution over words.

3 Proposed Model

In this section, we describe PositionRank, our fully unsupervised, graph-based model, that simultaneously incorporates the position of words and their frequency in a document to compute a biased PageRank score for each candidate word. Graph-based ranking algorithms such as PageRank (Page et al., 1998) measure the importance of a vertex within a graph by taking into account global information computed recursively from the entire graph. For each word, we compute a weight by aggregating information from all positions of the word's occurrences. This weight is then incorporated into a biased PageRank algorithm in order to assign a different "preference" to each word.

3.1 PositionRank

The PositionRank algorithm involves three essential steps: (1) the graph construction at word level; (2) the design of Position-Biased PageRank; and (3) the formation of candidate phrases. These steps are detailed below.

3.1.1 Graph Construction

Let d be a target document for extracting keyphrases. We first apply the part-of-speech filter using the NLP Stanford toolkit and then select as candidate words only nouns and adjectives, similar to previous works (Mihalcea and Tarau, 2004; Wan and Xiao, 2008). We build a word graph $G = (V, E)$ for d such that each unique word that passes the part-of-speech filter corresponds to a node in G. Two nodes v_i and v_j are connected by an edge $(v_i, v_j) \in E$ if the words corresponding to these nodes co-occur within a window of w contiguous tokens in the content of d. The weight of an edge $(v_i, v_j) \in E$ is computed based on the co-occurrence count of the two words within a window of w successive tokens in d. Note that the graph can be constructed both directed and undirected. However, Mihalcea and Tarau (2004) showed that the type of graph used to represent the text does not significantly influence the performance of keyphrase extraction. Hence, in this work, we build undirected graphs.

3.1.2 Position-Biased PageRank

Formally, let G be an undirected graph constructed as above and let M be its adjacency matrix. An element $m_{ij} \in M$ is set to the weight of edge (v_i, v_j) if there exist an edge between nodes v_i and v_j, and is set to 0 otherwise. The PageRank score of a node v_i is recursively computed by summing the normalized scores of nodes v_j, which are linked to v_i (as explained below).

Let S denote the vector of PageRank scores, for all $v_i \in V$. The initial values of S are set to $\frac{1}{|V|}$. The PageRank score of each node at step $t+1$, can then be computed recursively using:

$$S(t+1) = \widetilde{M} \cdot S(t) \tag{1}$$

where $\widetilde{M}$ is the normalized form of matrix M with $\widetilde{m_{ij}} \in \widetilde{M}$ defined as:

$$\widetilde{m_{ij}} = \begin{cases} m_{ij} / \sum_{j=1}^{|V|} m_{ij} & \text{if } \sum_{j=1}^{|V|} m_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

The PageRank computation can be seen as a Markov Chain process in which nodes represent states and the links between them are the transitions. By recursively applying Eq. (1), we obtain the principal eigenvector, which represents the stationary probability distribution of each state, in our case of each node (Manning et al., 2008).

To ensure that the PageRank (or the random walk) does not get stuck into cycles of the graph, a damping factor α is added to allow the "teleport" operation to another node in the graph. Hence, the computation of S becomes:

$$S = \alpha \cdot \widetilde{M} \cdot S + (1 - \alpha) \cdot \widetilde{p} \tag{2}$$

where S is the principal eigenvector and $\widetilde{p}$ is a vector of length $|V|$ with all elements $\frac{1}{|V|}$. The vector $\widetilde{p}$ indicates that, being in a node v_i, the random walk can jump to any other node in the graph with equal probability.

By biasing $\widetilde{p}$, the random walk would prefer nodes that have higher probability in the graph (Haveliwala, 2003).

The idea of PositionRank is to assign larger weights (or probabilities) to words that are found early in a document and are frequent. Specifically, we want to assign a higher probability to a word found on the 2^{nd} position as compared to a word

found on the 50^{th} position in the same document. We weigh each candidate word with its inverse position in the document before any filters are applied. If the same word appears multiple times in the target document, then we sum all its position weights. For example, if a word is found on the following positions: 2^{nd}, 5^{th} and 10^{th}, its weight is: $\frac{1}{2} + \frac{1}{5} + \frac{1}{10} = \frac{4}{5} = 0.8$. Summing up the position weights for a given word aims to grant more confidence to frequently occurring words by taking into account the position weight of each occurrence. Then, the vector $\widetilde{p}$ is set to the normalized weights for each candidate word as follows:

$$\widetilde{p} = \left[\frac{p_1}{p_1+p_2+...+p_{|V|}}, \frac{p_2}{p_1+p_2+...+p_{|V|}}, ..., \frac{p_{|V|}}{p_1+p_2+...+p_{|V|}}\right]$$

The PageRank score of a vertex v_i, i.e., $S(v_i)$, can be obtained in an algebraic way by recursively computing the following equation:

$$S(v_i) = (1-\alpha) \cdot \widetilde{p}_i + \alpha \cdot \sum_{v_j \in Adj(v_i)} \frac{w_{ji}}{O(v_j)} S(v_j)$$

where $O(v_j) = \sum_{v_k \in Adj(v_j)} w_{jk}$ and $\widetilde{p}_i$ is the weight found in the vector $\widetilde{p}$ for vertex v_i.

In our experiments, the words' PageRank scores are recursively computed until the difference between two consecutive iterations is less than 0.001 or a number of 100 iterations is reached.

3.1.3 Forming Candidate Phrases

Candidate words that have contiguous positions in a document are concatenated into phrases. We consider noun phrases that match the regular expression (adjective)*(noun)+, of length up to three, (i.e., unigrams, bigrams, and trigrams).

Finally, phrases are scored by using the sum of scores of individual words that comprise the phrase (Wan and Xiao, 2008). The top-scoring phrases are output as predictions (i.e., the predicted keyphrases for the document).

4 Experiments and Results

4.1 Datasets and Evaluation Metrics

In order to evaluate the performance of PositionRank, we carried out experiments on three datasets. The first and second datasets were made available by Gollapalli and Caragea (2014).[2] These datasets are compiled from the CiteSeerX digital library (Giles et al., 1998) and consist of research papers from the ACM Conference on Knowledge Discovery and Data Mining (KDD) and the World Wide Web Conference (WWW). The third dataset was made available by Nguyen and Kan (2007) and consist of research papers from various disciplines. In experiments, we use the title and abstract of each paper to extract keyphrases. The author-input keyphrases are used as gold-standard for evaluation. All three datasets are summarized in Table 1, which shows the number of papers in each dataset, the total number of keyphrases (Kp), the average number of keyphrases per document (AvgKp), and a brief insight into the length and number of available keyphrases.

Evaluation Metrics. We use mean reciprocal rank (MRR) curves to illustrate our experimental findings. MRR gives the averaged ranking of the first correct prediction and is defined as:

$$MRR = \frac{1}{|D|} \sum_{d \in D} \frac{1}{r_d}$$

where D is the collection of documents and r_d is the rank at which the first correct keyphrase of document d was found. We also summarize the results in terms of Precision, Recall, and F1-score in a table to contrast PositionRank with previous models since these metrics are widely used in previous works (Hulth, 2003; Wan and Xiao, 2008; Mihalcea and Tarau, 2004; Hasan and Ng, 2014). To compute "performance@k" (such as MRR@k), we examine the top-k predictions (with k ranging from 1 to 10). We use *average* k to refer to the average number of keyphrases for a particular dataset as listed in Table 1. For example, *average* $k = 5$ for the WWW dataset. For comparison purposes, we used Porter Stemmer to reduce both predicted and gold keyphrases to a base form.

4.2 Results and Discussion

Our experiments are organized around several questions, which are discussed below.

How sensitive is PositionRank to its parameters? One parameter of our model that can influence its performance is the window size w, which determines how edges are added between candidate words in the graph. We experimented with values of w ranging from 2 to 10 in steps of 1 and chose several configurations for illustration. Figure 2 shows the MRR curves of PositionRank for different values of w, on all three datasets. As can be seen from the figure, the performance of our model does not change significantly as w changes.

[2] http://www.cse.unt.edu/∼ccaragea/keyphrases.html

Dataset	#Docs	Kp	AvgKp	unigrams	bigrams	trigrams	n-grams ($n \geq 4$)
KDD	834	3093	3.70	810	1770	471	42
WWW	1350	6405	4.74	2254	3139	931	81
Nguyen	211	882	4.18	260	457	132	33

Table 1: A summary of our datasets.

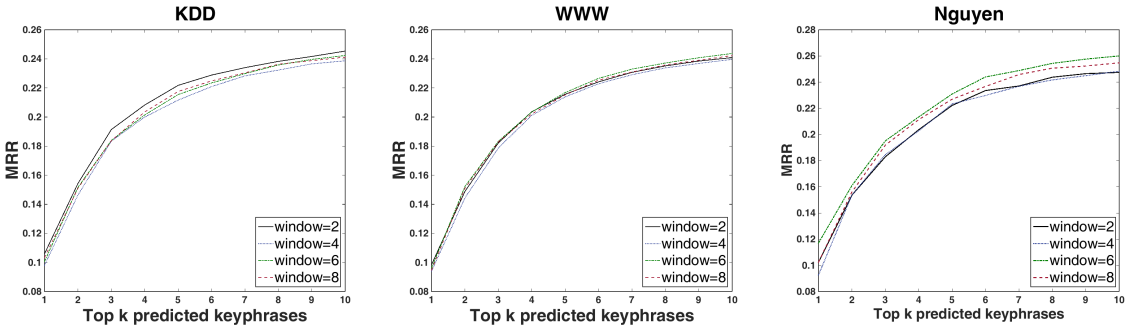


Figure 2: MRR curves for PositionRank that uses different values for the window size.

In addition to the window size, our model has one more parameter, i.e., the damping factor α. In order to understand its influence on the performance of PositionRank, we experimented with several values of α, e.g., $0.75, 0.8, 0.85, 0.9$, and did not find significant differences in the performance of PositionRank (results not shown due to highly overlapping curves). Hence, in Equation 2, we set $\alpha = 0.85$ as in (Haveliwala, 2003).

What is the impact of aggregating information from all positions of a word over using a word's first position only? In this experiment, we analyze the influence that position-weighted frequent words in a document would have on the performance of PositionRank. Specifically, we compare the performance of the model that aggregates information from all positions of a word's occurrences, referred as PositionRank - *full model* with that of the model that uses only the first position of a word, referred as PositionRank - *fp*. In the example from the previous section, a word occurring on positions 2^{nd}, 5^{th}, and 10^{th} will have a weight of $\frac{1}{2} + \frac{1}{5} + \frac{1}{10} = \frac{4}{5} = 0.8$ in the *full model*, and a weight of $\frac{1}{2} = 0.5$ in the *first position* (*fp*) model. Note that the weights of words are normalized before they are used in the biased PageRank.

Figure 3 shows the results of this experiment in terms of MRR for the top k predicted keyphrases, with k from 1 to 10, for all datasets, KDD, WWW, and Nguyen. As we can see from the figure, the performance of PositionRank - *full model* consistently outperforms its counterpart that uses the *first position* only, on all datasets. We can conclude from this experiment that aggregating information from all occurrences of a word acts as an important component in PositionRank. Hence, we use PositionRank - *full model* for further comparisons.

How well does position information aid in unsupervised keyphrase extraction from research papers? In this experiment, we compare our position-biased PageRank model (PositionRank) with two PageRank-based models, TextRank and SingleRank, that do not make use of the position information. In TextRank, an undirected graph is built for each target paper, so that nodes correspond to words and edges are drawn between two words that occur next to each other in text, i.e., the window size w is 2. SingleRank extends TextRank by adding edges between two words that co-occur in a window of $w \geq 2$ contiguous words in text.

Figure 4 shows the MRR curves comparing PositionRank with TextRank and SingleRank. As can be seen from the figure, PositionRank substantially outperforms both TextRank and SingleRank on all three datasets, illustrating that the words' positions contain significant hints that aid the keyphrase extraction task. PositionRank can successfully harness this information in an unsupervised setting to obtain good improvements in the extraction performance. For example, PositionRank that uses information from all positions of a word's occurrences yields improvements in MRR@average k of 17.46% for KDD, 20.18% for WWW, and 17.03% for Nguyen over SingleRank.

How does PositionRank compare with other existing state-of-the-art methods? In Figure 5, we

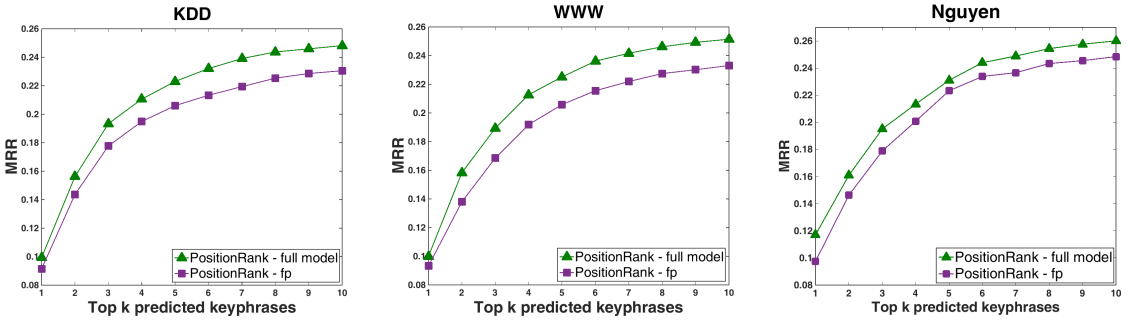


Figure 3: The comparison of PositionRank that aggregates information from all positions of a word's occurrences (full model) with the PositionRank that uses only the first position of a word (fp).

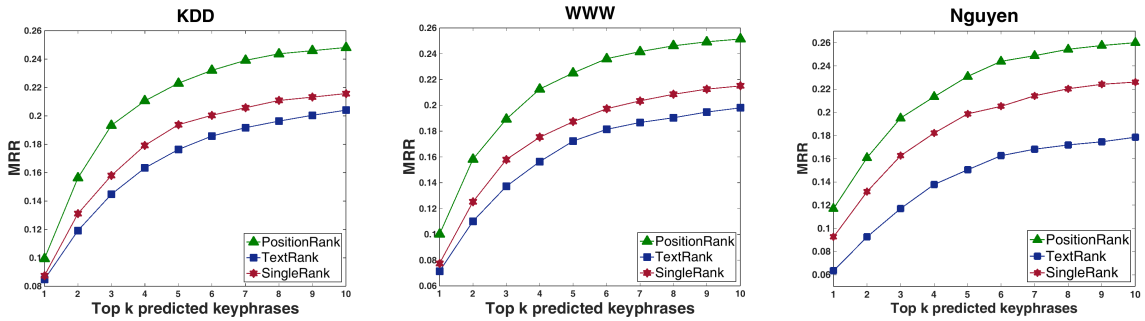


Figure 4: MRR curves for PositionRank and two unbiased PageRank-based models that do not consider position information.

compare PositionRank with several strong baselines: TF-IDF, ExpandRank, and TopicalPageRank (TPR) (Hasan and Ng, 2014; Wan and Xiao, 2008; Liu et al., 2010). We selected these baselines based on the ACL survey on keyphrase extraction by Hasan and Ng (2014). In TF-IDF, we calculate the *tf* score of each candidate word in the target document, whereas the *idf* component is estimated from all three datasets. In ExpandRank, we build an undirected graph from each paper and its local textual neighborhood and calculate the candidate words' importance scores using PageRank. We performed experiments with various numbers of textually-similar neighbors and present the best results for each dataset. In TPR, we build an undirected graph using information from the target paper. We then perform topic decomposition of the target document using topic models to infer the topic distribution of a document and to compute the probability of words in these topics. Last, we calculate the candidate words' importance scores by aggregating the scores from several topic-biased PageRanks (one PageRank per topic). We used the implementation of topic models from Mallet.[3] To train the topic model, we used a subset of about $45,000$ paper abstracts extracted from the CiteSeer^x scholarly big dataset introduced by Caragea et al. (2014b). For all models, the score of a phrase is obtained by summing the score of the constituent words in the phrase.

From Figure 5, we can see that PositionRank achieves a significant increase in MRR over the baselines, on all datasets. For example, the highest relative improvement in MRR@average k for this experiment is as high as 29.09% achieved on the Nguyen collection. Among all models compared in Figure 5, ExpandRank is clearly the best performing baseline, while TPR achieves the lowest MRR values, on all datasets.

4.3 Overall Performance

As already mentioned, prior works on keyphrase extraction report results also in terms of precision (P), recall (R), and F1-score (F1) (Hulth, 2003; Hasan and Ng, 2010; Liu et al., 2010; Wan and Xiao, 2008). Consistent with these works, in Table 2, we show the results of the comparison of PositionRank with all baselines, in terms of P, R and F1 for top $k = 2, 4, 6, 8$ predicted keyphrases, on all three datasets. As can be seen from the ta-

[3] http://mallet.cs.umass.edu/

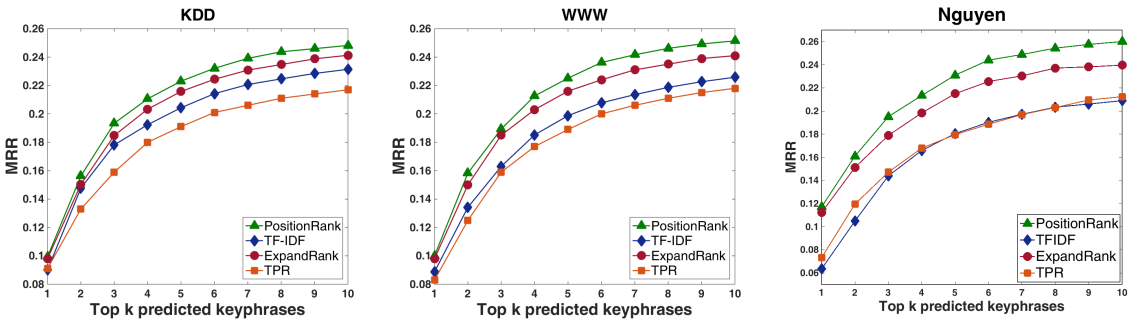


Figure 5: MRR curves for PositionRank and baselines on the three datasets.

Dataset	Unsupervised method	Top2			Top4			Top6			Top8		
		P%	R%	F1%	P%	R%	F1%	P%	R%	F1%	P%	R%	F1%
KDD	PositionRank	**11.1**	**5.6**	**7.3**	**10.8**	**11.1**	**10.6**	**9.8**	**15.3**	**11.6**	**9.2**	**18.9**	**12.1**
	PositionRank-fp	10.3	5.3	6.8	10.2	10.4	10.0	9.1	13.8	10.9	8.6	17.2	11.3
	TF-IDF	10.5	5.2	6.8	9.6	9.7	9.4	9.2	13.8	10.7	8.7	17.4	11.3
	TextRank	8.1	4.0	5.3	8.3	8.5	8.1	8.1	12.3	9.4	7.6	15.3	9.8
	SingleRank	9.1	4.6	6.0	9.3	9.4	9.0	8.7	13.1	10.1	8.1	16.4	10.6
	ExpandRank	10.3	5.5	6.9	10.4	10.7	10.1	9.2	14.5	10.9	8.4	17.5	11.0
	TPR	9.3	4.8	6.2	9.1	9.3	8.9	8.8	13.4	10.3	8.0	16.2	10.4
WWW	PositionRank	**11.3**	**5.3**	**7.0**	**11.3**	**10.5**	**10.5**	**10.8**	**14.9**	**12.1**	**9.9**	**18.1**	**12.3**
	PositionRank-fp	9.6	4.5	6.0	10.3	9.6	9.6	10.1	13.8	11.2	9.4	17.2	11.7
	TF-IDF	9.5	4.5	5.9	10.0	9.3	9.3	9.6	13.3	10.7	9.1	16.8	11.4
	TextRank	7.7	3.7	4.8	8.6	7.9	8.0	8.1	12.3	9.8	8.2	15.2	10.2
	SingleRank	9.1	4.2	5.6	9.6	8.9	8.9	9.3	13.0	10.5	8.8	16.3	11.0
	ExpandRank	10.4	5.3	6.7	10.4	10.6	10.1	9.5	14.7	11.2	8.6	17.7	11.2
	TPR	8.8	4.2	5.5	9.6	8.9	8.9	9.5	13.2	10.7	9.0	16.5	11.2
Nguyen	PositionRank	**10.5**	**5.8**	**7.3**	**10.6**	**11.4**	**10.7**	**11.0**	**17.2**	**13.0**	**10.2**	**21.1**	**13.5**
	PositionRank-fp	10.0	5.4	6.8	10.4	11.1	10.5	11.2	17.4	13.2	10.1	21.2	13.3
	TF-IDF	7.3	4.0	5.0	9.5	10.3	9.6	9.1	14.4	10.9	8.9	18.9	11.8
	TextRank	6.3	3.6	4.5	7.4	7.4	7.2	7.8	11.9	9.1	7.2	14.8	9.4
	SingleRank	9.0	5.2	6.4	9.5	9.9	9.4	9.2	14.5	11.0	8.9	18.3	11.6
	ExpandRank	9.5	5.3	6.6	9.5	10.2	9.5	9.1	14.4	10.8	8.7	18.3	11.4
	TPR	8.7	4.9	6.1	9.1	9.5	9.0	8.8	13.8	10.5	8.8	18.0	11.5

Table 2: PositionRank against baselines in terms of Precision, Recall and F1-score. Best results are shown in **bold blue**.

ble, PositionRank outperforms all baselines, on all datasets. For example, on WWW at top 6 predicted keyphrases, PositionRank achieves an F1-score of 12.1% as compared to 11.2% achieved by ExpandRank and 10.7% achieved by both TF-IDF and TPR. From the table, we can also see that ExpandRank is generally the best performing baseline on all datasets. However, it is interesting to note that, unlike PositionRank that uses information only from the target paper, ExpandRank adds external information from a textually-similar neighborhood of the target paper, and hence, is computationally more expensive.

PositionRank-*first position only* (*fp*) typically performs worse than PositionRank-*full model*, but it still outperforms the baseline methods for most top k predicted keyphrases, on all datasets. For example, on Nguyen at top 4, PositionRank-*fp* achieves an F1-score of 10.5% compared to the best baseline (TF-IDF in this case), which reaches only a score of 9.6%.

A striking observation is that PositionRank outperforms TPR on all datasets. Compared with our model, TPR is a very complex model, which uses topic models to learn topics of words and infer the topic proportion of documents. Additionally, TPR has more parameters (e.g., the number of topics) that need to be tuned separately for each dataset. PositionRank is much less complex, it does not require an additional dataset (e.g., to train a topic model) and its performance is better than that of TPR. TF-IDF and ExpandRank are the best performing baselines, on all datasets, KDD, WWW, and Nguyen. For example, on KDD at $k = 4$, TF-IDF and ExpandRank yield an F1-score of 9.4% and 10.1%, respectively, compared with 8.4%, 9.0% and 8.9% achieved by TextRank, SingleRank and TPR, respectively.

Geographically$^{0.274}$ Focused$^{0.134}$ **Collaborative**$^{0.142}$ **Crawling**$^{0.165}$
by Weizheng Gao, Hyun Chul Lee and Yingbo Miao A **collaborative**$^{0.142}$ **crawler**$^{0.165}$ is a group$^{0.025}$ of crawling$^{0.165}$ nodes$^{0.033}$, in which each crawling$^{0.165}$ node$^{0.033}$ is responsible$^{0.012}$ for a specific$^{0.010}$ portion$^{0.010}$ of the web$^{0.015}$. We study the problem$^{0.007}$ of collecting$^{0.011}$ geographically$^{0.274}$ aware$^{0.006}$ pages$^{0.018}$ using **collaborative**$^{0.142}$ **crawling**$^{0.165}$ strategies$^{0.017}$. We first propose several **collaborative**$^{0.142}$ **crawling**$^{0.165}$ strategies$^{0.017}$ for the **geographically**$^{0.274}$ **focused**$^{0.134}$ **crawling**$^{0.165}$, whose goal$^{0.004}$ is to collect web$^{0.015}$ pages$^{0.018}$ about specified$^{0.010}$ geographic$^{0.274}$ locations$^{0.003}$ by considering features$^{0.005}$ like URL$^{0.006}$ address$^{0.005}$ of page$^{0.018}$ [...] More precisely, features$^{0.005}$ like URL$^{0.006}$ address$^{0.005}$ of page$^{0.018}$ and extended$^{0.004}$ anchor$^{0.004}$ text$^{0.004}$ of link$^{0.004}$ are shown to yield the best overall performance$^{0.003}$ for the **geographically**$^{0.274}$ **focused**$^{0.134}$ **crawling**$^{0.165}$.

Author-input keyphrases: *collaborative crawling, geographically focused crawling, geographic entities*

Figure 6: The title and abstract of a WWW paper by Gao et al. (2006) and the author-input keyphrases for the paper. **Bold dark red** phrases represent predicted keyphrases for the document.

With a paired t-test on our results, we found that the improvements in MRR, precision, recall, and F1-score for PositionRank are statistically significant (p-values < 0.05).

4.4 Anecdotal Evidence

We show anecdotal evidence using a paper by Gao et al. (2006) that is part of the Nguyen dataset. Figure 6 shows the title and abstract of this paper together with the author-input keyphrases. We marked in bold dark red the candidate phrases that are predicted as keyphrases by our proposed model (PositionRank), in black the words that are selected as candidate phrases and in gray the words that are filtered out based on their part-of-speech tags or the stopwords list being used. We show the probability (or weight) of each candidate word in its upper right corner. These weights are computed based on both the word's position and its frequency in the text. Note that our model uses these weights to bias the PageRank algorithm to prefer specific nodes in the graph.

As we can see from the figure, component words of author's keyphrases such as: "collaborative," "crawling," "focused," and "geographically" are assigned the highest scores while candidates such as "performance," "anchor," or "features" are assigned very low weights, making them less likely to be chosen as keyphrases.

5 Conclusion and Future Work

We proposed a novel unsupervised graph-based algorithm, called PositionRank, which incorporates both the position of words and their frequency in a document into a biased PageRank. To our knowledge, we are the first to integrate the position information in novel ways in unsupervised keyphrase extraction. Specifically, unlike supervised approaches that use only the first position information, we showed that modeling the entire distribution of positions for a word outperforms models that use only the first position.

Our experiments on three datasets of research papers show that our proposed model achieves better results than strong baselines, with relative improvements in performance as high as 29.09%. In the future, it would be interesting to explore the performance of PositionRank on other types of documents, e.g., web pages and emails.

Acknowledgments

We are grateful to Dr. C. Lee Giles for the CiteSeerX data that we used to create our KDD and WWW datasets as well as to train the topic models. We very much thank our anonymous reviewers for their constructive comments and feedback. This research was supported by the NSF award #1423337 to Cornelia Caragea. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of NSF.

References

Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based summarization of scientific papers. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 500–509.

Eytan Adar and Srayan Datta. 2015. Building a scientific concept hierarchy database (schbase). In *Proceedings of the Association for Computational Linguistics*. pages 606–615.

Ken Barker and Nadia Cornacchia. 2000. Using noun phrase heads to extract document keyphrases. In *Advances in Artificial Intelligence*. pages 40–52.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International Joint Conference on Natural Language Processing (IJCNLP)*. pages 543–551.

Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014a. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 1435–1446.

Cornelia Caragea, Jian Wu, Alina Maria Ciobanu, Kyle Williams, Juan Pablo Fernández Ramírez, Hung-Hsuan Chen, Zhaohui Wu, and C. Lee Giles. 2014b. Citeseer x : A scholarly big dataset. In *Proceedings of the 36th European Conference on Information Retrieval*. pages 311–322.

Jason Chuang, Christopher D Manning, and Jeffrey Heer. 2012. Without the clutter of unimportant words: Descriptive keyphrases for text visualization. *ACM Transactions on Computer-Human Interaction* 19(3):19.

Soheil Danesh, Tamara Sumner, and James H Martin. 2015. Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction. *Lexical and Computational Semantics* page 117.

Samhaa R El-Beltagy and Ahmed Rafea. 2010. Kp-miner: Participation in semeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, pages 190–193.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. pages 668–673.

Weizheng Gao, Hyun Chul Lee, and Yingbo Miao. 2006. Geographically focused collaborative crawling. In *Proceedings of the 15th international conference on World Wide Web*. ACM, pages 287–296.

C Lee Giles, Kurt D Bollacker, and Steve Lawrence. 1998. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*. pages 89–98.

Sujatha Das Gollapalli and Cornelia Caragea. 2014. Extracting keyphrases from research papers using citation networks. In *Proceedings of the 28th American Association for Artificial Intelligence*. pages 1629–1635.

Khaled M Hammouda, Diego N Matute, and Mohamed S Kamel. 2005. Corephrase: Keyphrase extraction for document clustering. In *Machine Learning and Data Mining in Pattern Recognition*, Springer, pages 265–274.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics*. pages 365–373.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 27th International Conference on Computational Linguistics*. pages 1262–1273.

Taher H Haveliwala. 2003. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering* pages 784–796.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 216–223.

Steve Jones and Mark S. Staveley. 1999. Phrasier: A system for interactive document retrieval using keyphrases. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 160–167.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 Task 5: Automatic Keyphrase Extraction from Scientific Articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. pages 21–26.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic keyphrase extraction from scientific articles. *Language Resources and Evaluation, Springer* 47(3):723–742.

Shibamouli Lahiri, Sagnik Ray Choudhury, and Cornelia Caragea. 2014. Keyword and keyphrase extraction using centrality measures on collocation networks. *CoRR* abs/1401.6571.

Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. 2016. Unsupervised keyphrase extraction: Introducing new kinds of words to keyphrases. In *Australasian Joint Conference on Artificial Intelligence*. Springer, pages 665–671.

Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*. pages 17–24.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 366–376.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. pages 257–266.

Patrice Lopez and Laurent Romary. 2010. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 248–251.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Juan Martinez-Romo, Lourdes Araujo, and Andres Duque Fernandez. 2016. Semgraph: Extracting keyphrases following a novel semantic graph-based approach. *Journal of the Association for Information Science and Technology* 67(1):71–82.

Olena Medelyan, Eibe Frank, and Ian H Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. ACL, pages 1318–1327.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. pages 404–411.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Asian Digital Libraries*. Springer, pages 317–326.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The pagerank citation ranking: bringing order to the web. *Technical report, Standford Digital Library Technologies Project* .

Vahed Qazvinian, Dragomir R. Radev, and Arzucan Özgür. 2010. Citation summarization through keyphrase extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*. COLING '10, pages 895–903.

Peter D Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval* 2(4):303–336.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 2008 American Association for Artificial Intelligence*. pages 855–860.

Rui Wang, Wei Liu, and Chris McDonald. 2014. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*. page 39.

Hongyuan Zha. 2002. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. pages 113–120.

Yongzheng Zhang, Evangelos Milios, and Nur Zincir-Heywood. 2007. A comparative study on key phrase extraction methods in automatic web site summarization. *Journal of Digital Information Management* 5(5):323.

Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. 2004. World wide web site summarization. *Web Intelligence and Agent Systems* 2(1):39–53.

Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses

Ryan Lowe[♡*] **Michael Noseworthy**[♡*] **Iulian V. Serban**[♢]

Nicolas A.-Gontier[♡] **Yoshua Bengio**[♢‡] **Joelle Pineau**[♡‡]

[♡] Reasoning and Learning Lab, School of Computer Science, McGill University
[♢] Montreal Institute for Learning Algorithms, Université de Montréal
[‡] CIFAR Senior Fellow

Abstract

Automatically evaluating the quality of dialogue responses for unstructured domains is a challenging problem. Unfortunately, existing automatic evaluation metrics are biased and correlate very poorly with human judgements of response quality. Yet having an accurate automatic evaluation procedure is crucial for dialogue research, as it allows rapid prototyping and testing of new models with fewer expensive human evaluations. In response to this challenge, we formulate automatic dialogue evaluation as a learning problem. We present an evaluation model (ADEM) that learns to predict human-like scores to input responses, using a new dataset of human response scores. We show that the ADEM model's predictions correlate significantly, and at a level much higher than word-overlap metrics such as BLEU, with human judgements at both the utterance and system-level. We also show that ADEM can generalize to evaluating dialogue models unseen during training, an important step for automatic dialogue evaluation.

1 Introduction

Building systems that can naturally and meaningfully converse with humans has been a central goal of artificial intelligence since the formulation of the Turing test (Turing, 1950). Research on one type of such systems, sometimes referred to as non-task-oriented dialogue systems, goes back to the mid-60s with Weizenbaum's famous program *ELIZA*: a rule-based system mimicking a Rogerian psychotherapist by persistently either rephrasing statements or asking questions (Weizenbaum, 1966). Recently, there has been a surge of interest towards building large-scale non-task-oriented dialogue systems using neural networks (Sordoni et al., 2015b; Shang et al., 2015; Vinyals and Le, 2015; Serban et al., 2016a; Li et al., 2015). These models are trained in an end-to-end manner to optimize a single objective, usually the likelihood of generating the responses from a fixed corpus. Such models have already had a substantial impact in industry, including Google's Smart Reply system (Kannan et al., 2016), and Microsoft's Xiaoice chatbot (Markoff and Mozur, 2015), which has over 20 million users.

Context of Conversation
Speaker A: Hey, what do you want to do tonight?
Speaker B: Why don't we go see a movie?
Model Response
Nah, let's do something active.
Reference Response
Yeah, the film about Turing looks great!

Figure 1: Example where word-overlap scores fail for dialogue evaluation; although the model response is reasonable, it has no words in common with the reference response, and thus would be given low scores by metrics such as BLEU.

One of the challenges when developing such systems is to have a good way of measuring progress, in this case the performance of the chatbot. The Turing test provides one solution to the evaluation of dialogue systems, but there are limitations with its original formulation. The test requires live human interactions, which is expensive and difficult to scale up. Furthermore, the test requires carefully designing the instructions to the human interlocutors, in order to balance their behaviour and expectations so that different systems may be ranked accurately by performance. Although unavoidable, these instructions introduce bias into the evaluation measure. The more common approach of having

[*] Indicates equal contribution.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1116–1126
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1103

humans evaluate the quality of dialogue system responses, rather than distinguish them from human responses, induces similar drawbacks in terms of time, expense, and lack of scalability. In the case of chatbots designed for specific conversation domains, it may also be difficult to find sufficient human evaluators with appropriate background in the topic (Lowe et al., 2015).

Despite advances in neural network-based models, evaluating the quality of dialogue responses automatically remains a challenging and understudied problem in the non-task-oriented setting. The most widely used metric for evaluating such dialogue systems is BLEU (Papineni et al., 2002), a metric measuring word overlaps originally developed for machine translation. However, it has been shown that BLEU and other word-overlap metrics are biased and correlate poorly with human judgements of response quality (Liu et al., 2016). There are many obvious cases where these metrics fail, as they are often incapable of considering the semantic similarity between responses (see Figure 1). Despite this, many researchers still use BLEU to evaluate their dialogue models (Ritter et al., 2011; Sordoni et al., 2015b; Li et al., 2015; Galley et al., 2015; Li et al., 2016a), as there are few alternatives available that correlate with human judgements. While human evaluation should always be used to evaluate dialogue models, it is often too expensive and time-consuming to do this for every model specification (for example, for every combination of model hyperparameters). Therefore, having an accurate model that can evaluate dialogue response quality automatically — what could be considered an *automatic Turing test* — is critical in the quest for building human-like dialogue agents.

To make progress towards this goal, we make the simplifying assumption that a 'good' chatbot is one whose responses are scored highly on appropriateness by human evaluators. We believe this is sufficient for making progress as current dialogue systems often generate inappropriate responses. We also find empirically that asking evaluators for other metrics results in either low inter-annotator agreement, or the scores are highly correlated with appropriateness (see supp. material). Thus, we collect a dataset of appropriateness scores to various dialogue responses, and we use this dataset to train an *automatic dialogue evaluation model* (ADEM). The model is trained in a semi-supervised manner using a hierarchical recurrent neural network (RNN) to predict human scores. We show that ADEM scores correlate significantly with human judgement at both the utterance-level and system-level. We also show that ADEM can often generalize to evaluating new models, whose responses were unseen during training, making ADEM a strong first step towards effective automatic dialogue response evaluation.[1]

# Examples	4104
# Contexts	1026
# Training examples	2,872
# Validation examples	616
# Test examples	616
κ score (inter-annotator correlation)	0.63

Table 1: Statistics of the dialogue response evaluation dataset. Each example is in the form *(context, model response, reference response, human score)*.

2 Data Collection

To train a model to predict human scores to dialogue responses, we first collect a dataset of human judgements (scores) of Twitter responses using the crowdsourcing platform Amazon Mechanical Turk (AMT).[2] The aim is to have accurate human scores for a variety of conversational responses — conditioned on dialogue contexts — which span the full range of response qualities. For example, the responses should include both relevant and irrelevant responses, both coherent and non-coherent responses and so on. To achieve this variety, we use candidate responses from several different models. Following (Liu et al., 2016), we use the following 4 sources of candidate responses: (1) a response selected by a TF-IDF retrieval-based model, (2) a response selected by the Dual Encoder (DE) (Lowe et al., 2015), (3) a response generated using the hierarchical recurrent encoder-decoder (HRED) model (Serban et al., 2016a), and (4) human-generated responses. It should be noted that the human-generated candidate responses are *not* the reference responses from a fixed corpus, but novel human responses that are different from the reference. In addition to increasing response variety, this is necessary because we want our evaluation model to learn to compare the reference responses to the candidate responses. We provide the details of our

[1] Code and trained model parameters are available online: https://github.com/mike-n-7/ADEM.

[2] All data collection was conducted in accordance with the policies of the host institutions' ethics board.

AMT experiments in the supplemental material, including additional experiments suggesting that several other metrics are currently unlikely to be useful for building evaluation models. Note that, in order to maximize the number of responses obtained with a fixed budget, we only obtain one evaluation score per dialogue response in the dataset.

To train evaluation models on human judgements, it is crucial that we obtain scores of responses that lie near the distribution produced by advanced models. This is why we use the Twitter Corpus (Ritter et al., 2011), as such models are pre-trained and readily available. Further, the set of topics discussed is quite broad — as opposed to the very specific Ubuntu Dialogue Corpus (Lowe et al., 2015) — and therefore the model may also be suited to other chit-chat domains. Finally, since it does not require domain specific knowledge (e.g. technical knowledge), it should be easy for AMT workers to annotate.

3 Technical Background

3.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a type of neural network with time-delayed connections between the internal units. This leads to the formation of a *hidden state* h_t, which is updated for every input: $h_t = f(W_{hh}h_{t-1} + W_{ih}x_t)$, where W_{hh} and W_{ih} are parameter matrices, f is a non-linear activation function such as tanh, and x_t is the input at time t. The hidden state allows for RNNs to better model sequential data, such as language.

In this paper, we consider RNNs augmented with long-short term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). LSTMs add a set of gates to the RNN that allow it to learn how much to update the hidden state. LSTMs are one of the most well-established methods for dealing with the vanishing gradient problem in recurrent networks (Hochreiter, 1991; Bengio et al., 1994).

3.2 Word-Overlap Metrics

One of the most popular approaches for automatically evaluating the quality of dialogue responses is by computing their *word overlap* with the reference response. In particular, the most popular metrics are the BLEU and METEOR scores used for machine translation, and the ROUGE score used for automatic summarization. While these metrics tend to correlate with human judgements in their target domains, they have recently been shown to highly biased and correlate very poorly with human judgements for dialogue response evaluation (Liu et al., 2016). We briefly describe BLEU here, and provide a more detailed summary of word-overlap metrics in the supplemental material.

BLEU BLEU (Papineni et al., 2002) analyzes the co-occurrences of n-grams in the reference and the proposed responses. It computes the n-gram precision for the whole dataset, which is then multiplied by a brevity penalty to penalize short translations. For BLEU-N, N denotes the largest value of n-grams considered (usually $N = 4$).

Drawbacks One of the major drawbacks of word-overlap metrics is their failure in capturing the semantic similarity (and other structure) between the model and reference responses when there are few or no common words. This problem is less critical for machine translation; since the set of reasonable translations of a given sentence or document is rather small, one can reasonably infer the quality of a translated sentence by only measuring the word-overlap between it and one (or a few) reference translations. However, in dialogue, the set of appropriate responses given a context is much larger (Artstein et al., 2009); in other words, there is a very high *response diversity* that is unlikely to be captured by word-overlap comparison to a single response.

Further, word-overlap scores are computed directly between the model and reference responses. As such, they do not consider the context of the conversation. While this may be a reasonable assumption in machine translation, it is not the case for dialogue; whether a model response is an adequate substitute for the reference response is clearly context-dependent. For example, the two responses in Figure 1 are equally appropriate given the context. However, if we simply change the context to: *"Have you heard of any good movies recently?"*, the model response is no longer relevant while the reference response remains valid.

4 An Automatic Dialogue Evaluation Model (ADEM)

To overcome the problems of evaluation with word-overlap metrics, we aim to construct a dialogue evaluation model that: (1) captures semantic similarity beyond word overlap statistics, and (2) exploits both the context and the reference response to calculate its score for the model response. We

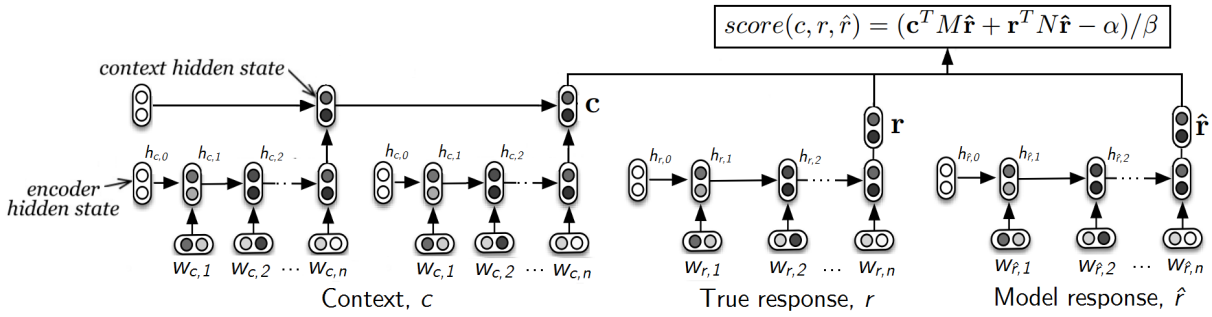


Figure 2: The ADEM model, which uses a hierarchical encoder to produce the context embedding $\mathbf{c}$.

call this evaluation model ADEM.

ADEM learns distributed representations of the context, model response, and reference response using a hierarchical RNN encoder. Given the dialogue context c, reference response r, and model response $\hat{r}$, ADEM first encodes each of them into vectors ($\mathbf{c}$, $\hat{\mathbf{r}}$, and $\mathbf{r}$, respectively) using the RNN encoder. Then, ADEM computes the score using a dot-product between the vector representations of c, r, and $\hat{r}$ in a linearly transformed space: :

$$score(c, r, \hat{r}) = (\mathbf{c}^T M \hat{\mathbf{r}} + \mathbf{r}^T N \hat{\mathbf{r}} - \alpha)/\beta \quad (1)$$

where $M, N \in \mathbb{R}^n$ are learned matrices initialized to the identity, and α, β are scalar constants used to initialize the model's predictions in the range $[1, 5]$. The model is shown in Figure 2.

The matrices M and N can be interpreted as linear projections that map the model response $\hat{\mathbf{r}}$ into the space of contexts and reference responses, respectively. The model gives high scores to responses that have similar vector representations to the context and reference response after this projection. The model is end-to-end differentiable; all the parameters can be learned by backpropagation. In our implementation, the parameters $\theta = \{M, N\}$ of the model are trained to minimize the squared error between the model predictions and the human score, with L2-regularization:

$$\mathcal{L} = \sum_{i=1:K} [score(c_i, r_i, \hat{r}_i) - human_i]^2 + \gamma ||\theta||_2 \quad (2)$$

where γ is a scalar constant. The simplicity of our model leads to both accurate predictions and fast evaluation (see supp. material), which is important to allow rapid prototyping of dialogue systems.

The hierarchical RNN encoder in our model consists of two layers of RNNs (El Hihi and Bengio, 1995; Sordoni et al., 2015a). The lower-level RNN, the *utterance-level encoder*, takes as input words from the dialogue, and produces a vector output at the end of each utterance. The *context-level encoder* takes the representation of each utterance as input and outputs a vector representation of the context. This hierarchical structure is useful for incorporating information from early utterances in the context (Serban et al., 2016a). Following previous work, we take the last hidden state of the context-level encoder as the vector representation of the input utterance or context. The parameters of the RNN encoder are pretrained and are not learned from the human scores.

An important point is that the ADEM procedure above *is not a dialogue retrieval model*: the fundamental difference is that ADEM has access to the reference response. Thus, ADEM can compare a model's response to a known good response, which is significantly easier than inferring response quality from solely the context.

Pre-training with VHRED We would like an evaluation model that can make accurate predictions from few labeled examples, since these examples are expensive to obtain. We therefore employ semi-supervised learning, and use a pre-training procedure to learn the parameters of the encoder. In particular, we train the encoder as part of a neural dialogue model; we attach a third *decoder RNN* that takes the output of the encoder as input, and train it to predict the next utterance of a dialogue conditioned on the context.

The dialogue model we employ for pre-training is the latent variable hierarchical recurrent encoder-decoder (VHRED) model (Serban et al., 2016b), shown in Figure 3. The VHRED model is an extension of the original hierarchical recurrent encoder-decoder (HRED) model (Serban et al., 2016a) with a turn-level stochastic latent variable. The dialogue context is encoded into a vector using our hierarchical encoder, and the VHRED then samples a Gaus-

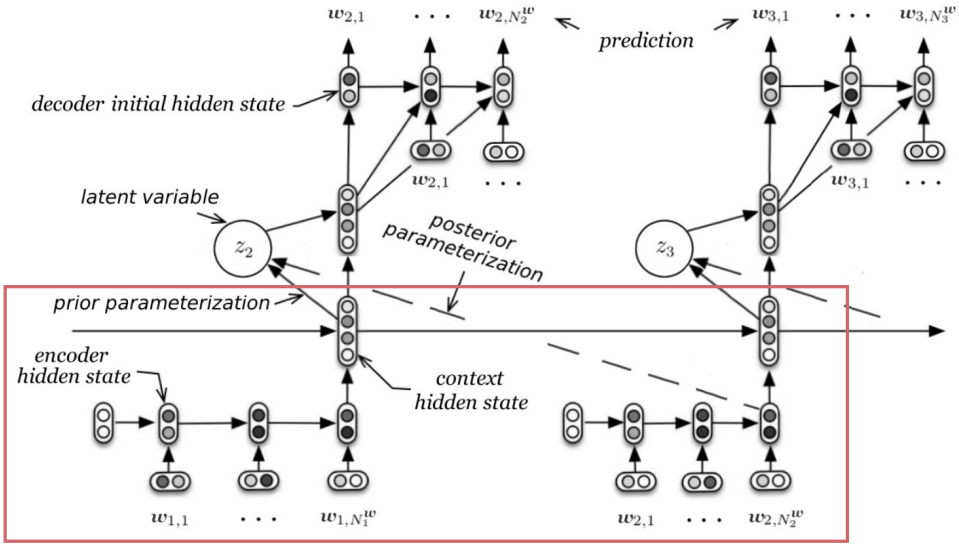


Figure 3: The VHRED model used for pre-training. The hierarchical structure of the RNN encoder is shown in the red box around the bottom half of the figure. After training using the VHRED procedure, the last hidden state of the context-level encoder is used as a vector representation of the input text.

sian variable that is used to condition the decoder (see supplemental material for further details). After training VHRED, we use the last hidden state of the context-level encoder, when c, r, and $\hat{r}$ are fed as input, as the vector representations for $\mathbf{c}$, $\mathbf{r}$, and $\hat{\mathbf{r}}$, respectively. We use representations from the VHRED model as it produces more diverse and coherent responses compared to HRED.

5 Experiments

5.1 Experimental Procedure

In order to reduce the effective vocabulary size, we use byte pair encoding (BPE) (Gage, 1994; Sennrich et al., 2015), which splits each word into sub-words or characters. We also use layer normalization (Ba et al., 2016) for the hierarchical encoder, which we found worked better at the task of dialogue generation than the related recurrent batch normalization (Ioffe and Szegedy, 2015; Cooijmans et al., 2016). To train the VHRED model, we employed several of the same techniques found in (Serban et al., 2016b) and (Bowman et al., 2016): we drop words in the decoder with a fixed rate of 25%, and we anneal the KL-divergence term linearly from 0 to 1 over the first 60,000 batches. We use Adam as our optimizer (Kingma and Ba, 2014).

When training ADEM, we also employ a subsampling procedure based on the model response length. In particular, we divide the training examples into bins based on the number of words in a response and the score of that response. We then over-sample from bins across the same score to ensure that ADEM does not use response length to predict the score. This is because humans have a tendency to give a higher rating to shorter responses than to longer responses (Serban et al., 2016b), as shorter responses are often more generic and thus are more likely to be suitable to the context. Indeed, the test set Pearson correlation between response length and human score is 0.27.

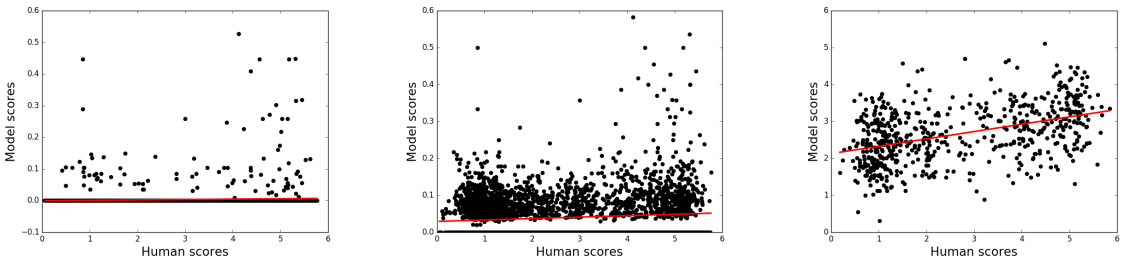
For training VHRED, we use a context embedding size of 2000. However, we found the ADEM model learned more effectively when this embedding size was reduced. Thus, after training VHRED, we use principal component analysis (PCA) (Pearson, 1901) to reduce the dimensionality of the context, model response, and reference response embeddings to n. We found experimentally that $n = 50$ provided the best performance.

When training our models, we conduct early stopping on a separate validation set. For the evaluation dataset, we split the train/ validation/ test sets such that there is no context overlap (i.e. the contexts in the test set are unseen during training).

5.2 Results

Utterance-level correlations We first present new utterance-level correlation results[3] for existing

[3]We present both the Spearman correlation (computed on ranks, depicts monotonic relationships) and Pearson correlation (computed on true values, depicts linear relationships)



(a) BLEU-2 (b) ROUGE (c) ADEM

Figure 4: Scatter plot showing model against human scores, for BLEU-2 and ROUGE on the full dataset, and ADEM on the test set. We add Gaussian noise drawn from $\mathcal{N}(0, 0.3)$ to the integer human scores to better visualize the density of points, at the expense of appearing less correlated.

	Full dataset		**Test set**	
Metric	**Spearman**	**Pearson**	**Spearman**	**Pearson**
BLEU-2	0.039 (0.013)	0.081 (<0.001)	0.051 (0.254)	0.120 (<0.001)
BLEU-4	0.051 (0.001)	0.025 (0.113)	0.063 (0.156)	0.073 (0.103)
ROUGE	0.062 (<0.001)	0.114 (<0.001)	0.096 (0.031)	0.147 (<0.001)
METEOR	0.021 (0.189)	0.022 (0.165)	0.013 (0.745)	0.021 (0.601)
T2V	0.140 (<0.001)	0.141 (<0.001)	0.140 (<0.001)	0.141 (<0.001)
VHRED	-0.035 (0.062)	-0.030 (0.106)	-0.091 (0.023)	-0.010 (0.805)
	Validation set		**Test set**	
C-ADEM	0.338 (<0.001)	0.355 (<0.001)	0.366 (<0.001)	0.363 (<0.001)
R-ADEM	0.404 (<0.001)	0.404 (<0.001)	0.352 (<0.001)	0.360 (<0.001)
ADEM (T2V)	0.252 (<0.001)	0.265 (<0.001)	0.280 (<0.001)	0.287 (<0.001)
ADEM	**0.410** (<0.001)	**0.418** (<0.001)	**0.428** (<0.001)	**0.436** (<0.001)

Table 2: Correlation between metrics and human judgements, with p-values shown in brackets. 'ADEM (T2V)' indicates ADEM with tweet2vec embeddings (Dhingra et al., 2016), and 'VHRED' indicates the dot product of VHRED embeddings (i.e. ADEM at initialization). C- and R-ADEM represent the ADEM model trained to only compare the model response to the context or reference response, respectively. We compute the baseline metric scores (top) on the full dataset to provide a more accurate estimate of their scores (as they are not trained on a training set).

word-overlap metrics, in addition to results with embedding baselines and ADEM, in Table 2. The baseline metrics are evaluated on the entire dataset of 4,104 responses to provide the most accurate estimate of the score. [4] We measure the correlation for ADEM on the validation and test sets, which constitute 616 responses each.

We also conduct an analysis of the response data from (Liu et al., 2016), where the pre-processing is standardized by removing '<first_speaker>' tokens at the beginning of each utterance. The results are detailed in the supplemental material. We can observe from both this data, and the new data in Table 2, that the correlations for the word-overlap metrics are even lower than estimated in previous studies (Liu et al., 2016; Galley et al., 2015). In particular, this is the case for BLEU-4, which has frequently been used for dialogue response evaluation (Ritter et al., 2011; Sordoni et al., 2015b; Li et al., 2015; Galley et al., 2015; Li et al., 2016a).

We can see from Table 2 that ADEM correlates far better with human judgement than the word-overlap baselines. This is further illustrated by the scatterplots in Figure 4. We also compare with ADEM using tweet2vec embeddings (Dhingra et al., 2016). In this case, instead of using the VHRED pre-training method presented in Section 4, we use off-the-shelf embeddings for $\mathbf{c}$, $\mathbf{r}$, and $\hat{\mathbf{r}}$, and fine-tune M and N on our dataset. These tweet2vec embeddings are computed at the character-level with a bidirectional GRU on a Twitter dataset for hashtag prediction (Dhingra et al., 2016). We find that they obtain reasonable but inferior performance compared to using VHRED embeddings.

scores.

[4] Note that our word-overlap correlation results in Table 2 are also lower than those presented in (Galley et al., 2015). This is because Galley et al. measure corpus-level correlation, i.e. correlation averaged across different subsets (of size 100) of the data, and pre-filter for high-quality reference responses.

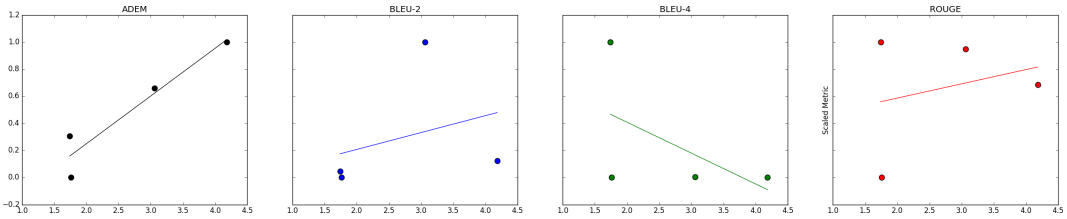


Figure 5: Scatterplots depicting the system-level correlation results for ADEM, BLEU-2, BLEU-4,and ROUGE on the test set. Each point represents the average scores for the responses from a dialogue model (TFIDF, DE, HRED, human). Human scores are shown on the horizontal axis, with normalized metric scores on the vertical axis. The ideal metric has a perfectly linear relationship.

System-level correlations We show the system-level correlations for various metrics in Table 3, and present it visually in Figure 5. Each point in the scatterplots represents a dialogue model; humans give low scores to TFIDF and DE responses, higher scores to HRED and the highest scores to other human responses. It is clear that existing word-overlap metrics are incapable of capturing this relationship for even 4 models. This renders them completely deficient for dialogue evaluation. However, ADEM produces almost the same model ranking as humans, achieving a significant Pearson correlation of 0.954.[5] Thus, ADEM correlates well with humans both at the response and system level.

Metric	Pearson
BLEU-1	-0.079 (0.921)
BLEU-2	0.308 (0.692)
BLEU-3	-0.537 (0.463)
BLEU-4	-0.536 (0.464)
ROUGE	0.268 (0.732)
ADEM	**0.954** (0.046)

Table 3: System-level correlation, with the p-value in brackets.

Generalization to previously unseen models When ADEM is used in practice, it will take as input responses from a new model that it has not seen during training. Thus, it is crucial that ADEM correlates with human judgements for new models. We test ADEM's generalization ability by performing a leave-one-out evaluation. For each dialogue model that was the source of response data for training ADEM (TF-IDF, Dual Encoder, HRED, humans), we conduct an experiment where we train on all model responses *except* those from the chosen model, and test *only* on the model that was unseen during training.

The results are given in Table 4. We observe that the ADEM model is able to generalize for all models except the Dual Encoder. This is particularly surprising for the HRED model; in this case, ADEM was trained only on responses that were written by humans (from retrieval models or human-generated), but is able to generalize to responses produced by a generative neural network model. When testing on the entire test set, the model achieves comparable correlations to the ADEM model that was trained on 25% less data selected at random.

Qualitative Analysis To illustrate some strengths and weaknesses of ADEM, we show human and ADEM scores for each of the responses to various contexts in Table 5. There are several instances where ADEM predicts accurately: in particular, ADEM is often very good at assigning low scores to poor responses. This seen in the first two contexts, where most of the responses given a score of 1 from humans are given scores less than 2 by ADEM. The single exception in response (4) for the second context seems somewhat appropriate and should perhaps have been scored higher by the human evaluator. There are also several instances where the model assigns high scores to suitable responses, as in the first two contexts.

One drawback we observed is that ADEM tends to be too conservative when predicting response scores. This is the case in the third context, where the model assigns low scores to most of the responses that a human rated highly. This behaviour is likely due to the squared error loss used to train ADEM; since the model receives a large penalty for incorrectly predicting an extreme value, it learns to predict scores closer to the average human score. We provide many more experiments, including investigation of evaluation speed, learning curves, data efficiency, a failure analysis, and the primary source of improvement over word-overlap metrics

[5] For comparison, BLEU achieves a system-level correlation of 0.99 on 5 models in the translation domain (Papineni et al., 2002).

	Test on full dataset		Test on removed model responses	
Data Removed	**Spearman**	**Pearson**	**Spearman**	**Pearson**
TF-IDF	0.406 (<0.001)	0.409 (<0.001)	0.186 (0.021)	0.196 (0.015)
Dual Encoder	0.364 (<0.001)	0.373 (<0.001)	0.026 (0.749)	0.027 (0.736)
HRED	0.393 (<0.001)	0.396 (<0.001)	0.151 (0.060)	0.148 (<0.070)
Human	0.292 (<0.001)	0.298 (<0.001)	0.216 (<0.010)	0.148 (<0.070)
Average	**0.364**	**0.369**	**0.145**	**0.130**
25% at random	0.378 (<0.001)	0.384 (<0.001)	—	—

Table 4: Correlation for ADEM when various model responses are removed from the training set. The left two columns show performance on the entire test set, and the right two columns show performance on responses only from the dialogue model not seen during training. The last row (25% at random) corresponds to the ADEM model trained on all model responses, but with the same amount of training data as the model above (i.e. 25% less data than the full training set).

Context	**Reference response**	**Model responses**	**Human score**	ADEM **score**
photo to see my television debut go to - some. some on <url> - hehe <url> → it really was you? i thought ppl were recognizing someone who looked like you! were the oysters worth the wait?	yeah it was me . haha i 'd kinda forgotten about it it was filmed a while ago	1) i'm not sure. i just don't know what to do with it.	3	1.602
		2) you heard the horsepower productions remix of lee scratch perry's 'exercising' off his 'mighty upsetter' album?	1	1.513
		3) you wont chug a fuzzy peach navel	1	1.744
		4) they were!	5	3.274
just beat call of duty!! → want a cookie? → yes!! → come get it	im in kenmore at the moment	1) i'm gonna get a new phone some moro	1	1.848
		2) no way man.	5	4.265
		3) wow i just got a free pizza coupon! get yours before theres no more! <url>	1	0.921
		4) i'm going to go to the mall.	1	2.634
am i out of twitter jail yet? testing → yeah. i posted bail → thanks. i am a right chatter tweetbox on sundays. same happened last sunday lol	any news on meeting our user ? i go to the us on friday and i don 't want to miss anything arranged	1) i'm not sure if i'm going to be able to get it.	3	1.912
		2) good to see another mac user in the leadership ranks	4	1.417
		3) awww poor baby hope u get to feeling better soon. maybe some many work days at piedmont	2	1.123
		4) did you tweet too much?	5	2.539

Table 5: Examples of scores given by the ADEM model.

in the supplemental material.

6 Related Work

Related to our approach is the literature on novel methods for the evaluation of machine translation systems, especially through the WMT evaluation task (Callison-Burch et al., 2011; Macháček and Bojar, 2014; Stanojevic et al., 2015). In particular, (Albrecht and Hwa, 2007; Gupta et al., 2015) have proposed to evaluate machine translation systems using Regression and Tree-LSTMs respectively. Their approach differs from ours as, in the dialogue domain, we must additionally condition our score on the context of the conversation, which is not necessary in translation.

There has also been related work on estimating the quality of responses in chat-oriented dialogue systems. (DeVault et al., 2011) train an automatic dialogue policy evaluation metric from 19 structured role-playing sessions, enriched with paraphrases and external referee annotations. (Gandhe and Traum, 2016) propose a semi-automatic evaluation metric for dialogue coherence, similar to BLEU and ROUGE, based on 'wizard of Oz' type data.[6] (Xiang et al., 2014) propose a framework to predict utterance-level problematic situations in a dataset of Chinese dialogues using intent and sentiment factors. Finally, (Higashinaka et al., 2014) train a classifier to distinguish user utterances from system-generated utterances using various dialogue features, such as dialogue acts, question types, and predicate-argument structures.

Several recent approaches use hand-crafted reward features to train dialogue models using reinforcement learning (RL). For example, (Li et al., 2016b) use features related to ease of answering and information flow, and (Yu et al., 2016) use metrics related to turn-level appropriateness and conversational depth. These metrics are based on hand-crafted features, which only capture a small set of relevant aspects; this inevitably leads to sub-optimal performance, and it is unclear whether such objectives are preferable over retrieval-based cross-entropy or word-level maximum log-likelihood objectives. Furthermore, many of these metrics are computed at the conversation-level, and are not available for evaluating single dialogue responses.

[6] In 'wizard of Oz' scenarios, humans play the role of the dialogue system, usually unbeknown to the interlocutors.

The metrics that can be computed at the response-level could be incorporated into our framework, for example by adding a term to equation 1 consisting of a dot product between these features and a vector of learned parameters.

There has been significant work on evaluation methods for task-oriented dialogue systems, which attempt to solve a user's task such as finding a restaurant. These methods include the PARADISE framework (Walker et al., 1997) and MeMo (Möller et al., 2006), which consider a task completion signal. PARADISE in particular is perhaps the first work on learning an automatic evaluation function for dialogue, accomplished through linear regression. However, PARADISE requires that one can measure task completion and task complexity, which are not available in our setting.

7 Discussion

We use the Twitter Corpus to train our models as it contains a broad range of non-task-oriented conversations and it has been used to train many state-of-the-art models. However, our model could easily be extended to other general-purpose datasets, such as Reddit, once similar pre-trained models become publicly available. Such models are necessary even for creating a test set in a new domain, which will help us determine if ADEM generalizes to related dialogue domains. We leave investigating the domain transfer ability of ADEM for future work.

The evaluation model proposed in this paper favours dialogue models that generate responses that are rated as highly appropriate by humans. It is likely that this property does not fully capture the desired end-goal of chatbot systems. For example, one issue with building models to approximate human judgements of response quality is the problem of generic responses. Since humans often provide high scores to generic responses due to their appropriateness for many given contexts (Shang et al., 2016), a model trained to predict these scores will exhibit the same behaviour. An important direction for future work is modifying ADEM such that it is not subject to this bias. This could be done, for example, by censoring ADEM's representations (Edwards and Storkey, 2016) such that they do not contain any information about length. Alternatively, one can combine this with an *adversarial evaluation model* (Kannan and Vinyals, 2017; Li et al., 2017) that assigns a score based on how easy it is to distinguish the dialogue model responses from human responses. In this case, a model that generates generic responses will easily be distinguishable and obtain a low score.

An important direction of future research is building models that can evaluate the capability of a dialogue system to have an engaging and meaningful interaction with a human. Compared to evaluating a single response, this evaluation is arguably closer to the end-goal of chatbots. However, such an evaluation is extremely challenging to do in a completely automatic way. We view the evaluation procedure presented in this paper as an important step towards this goal; current dialogue systems are incapable of generating responses that are rated as highly appropriate by humans, and we believe our evaluation model will be useful for measuring and facilitating progress in this direction.

References

Joshua Albrecht and Rebecca Hwa. 2007. Regression for sentence-level mt evaluation with pseudo references. In *ACL*.

Ron Artstein, Sudeep Gandhe, Jillian Gerten, Anton Leuski, and David Traum. 2009. Semi-formal evaluation of conversational characters. In *Languages: From Formal to Natural*, Springer, pages 22–35.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5(2):157–166.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *COLING* .

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar F Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 22–64.

Tim Cooijmans, Nicolas Ballas, César Laurent, and Aaron Courville. 2016. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025* .

David DeVault, Anton Leuski, and Kenji Sagae. 2011. Toward learning and evaluation of dialogue policies with text examples. In *Proceedings of the SIGDIAL 2011 Conference*. Association for Computational Linguistics, pages 39–48.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481* .

Harrison Edwards and Amos Storkey. 2016. Censoring representations with an adversary. *ICLR* .

Salah El Hihi and Yoshua Bengio. 1995. Hierarchical recurrent neural networks for long-term dependencies. In *NIPS*. Citeseer, volume 400, page 409.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal* 12(2):23–38.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltableu: A discriminative metric for generation tasks with intrinsically diverse targets. *arXiv preprint arXiv:1506.06863* .

Sudeep Gandhe and David Traum. 2016. A semi-automated evaluation metric for dialogue model coherence. In *Situated Dialog in Speech-Based Human-Computer Interaction*, Springer, pages 217–225.

Rohit Gupta, Constantin Orasan, and Josef van Genabith. 2015. Reval: A simple and effective machine translation evaluation metric based on recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ryuichiro Higashinaka, Toyomi Meguro, Kenji Imamura, Hiroaki Sugiyama, Toshiro Makino, and Yoshihiro Matsuo. 2014. Evaluating coherence in open domain conversational systems. In *INTERSPEECH*. pages 130–134.

Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München* page 91.

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

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* .

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, et al. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. volume 36, pages 495–503.

Anjuli Kannan and Oriol Vinyals. 2017. Adversarial evaluation of dialogue models. *arXiv preprint arXiv:1701.08198* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155* .

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Learning to decode for future success. *arXiv preprint arXiv:1701.06549* .

Jiwei Li, Will Monroe, Alan Ritter, and Dan Jurafsky. 2016b. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* .

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023* .

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909* .

Matouš Machácek and Ondrej Bojar. 2014. Results of the wmt14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Citeseer, pages 293–301.

J. Markoff and P. Mozur. 2015. For sympathetic ear, more chinese turn to smartphone program. *NY Times* .

Sebastian Möller, Roman Englert, Klaus-Peter Engelbrecht, Verena Vanessa Hafner, Anthony Jameson, Antti Oulasvirta, Alexander Raake, and Norbert Reithinger. 2006. Memo: towards automatic usability evaluation of spoken dialogue services by user error simulations. In *INTERSPEECH*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Karl Pearson. 1901. Principal components analysis. *The London, Edinburgh and Dublin Philosophical Magazine and Journal* 6(2):566.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 583–593.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* .

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016a. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*. pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016b. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069* .

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364* .

Lifeng Shang, Tetsuya Sakai, Zhengdong Lu, Hang Li, Ryuichiro Higashinaka, and Yusuke Miyao. 2016. Overview of the ntcir-12 short text conversation task. *Proceedings of NTCIR-12* pages 473–484.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015a. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, pages 553–562.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015b. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

Miloš Stanojevic, Amir Kamran, Philipp Koehn, and Ondrej Bojar. 2015. Results of the wmt15 metrics shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. pages 256–273.

Alan M Turing. 1950. Computing machinery and intelligence. *Mind* 59(236):433–460.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 271–280.

J. Weizenbaum. 1966. ELIZAa computer program for the study of natural language communication between man and machine. *Communications of the ACM* 9(1):36–45.

Yang Xiang, Yaoyun Zhang, Xiaoqiang Zhou, Xiaolong Wang, and Yang Qin. 2014. Problematic situation analysis and automatic recognition for chi-nese online conversational system. *Proc. CLP* pages 43–51.

Zhou Yu, Ziyu Xu, Alan W Black, and Alex I Rudnicky. 2016. Strategy and policy learning for non-task-oriented conversational systems. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 404.

A Transition-Based Directed Acyclic Graph Parser for UCCA

Daniel Hershcovich[1,2] **Omri Abend**[2] **Ari Rappoport**[2]
[1]The Edmond and Lily Safra Center for Brain Sciences
[2]School of Computer Science and Engineering
Hebrew University of Jerusalem
{danielh,oabend,arir}@cs.huji.ac.il

Abstract

We present the first parser for UCCA, a cross-linguistically applicable framework for semantic representation, which builds on extensive typological work and supports rapid annotation. UCCA poses a challenge for existing parsing techniques, as it exhibits reentrancy (resulting in DAG structures), discontinuous structures and non-terminal nodes corresponding to complex semantic units. To our knowledge, the conjunction of these formal properties is not supported by any existing parser. Our transition-based parser, which uses a novel transition set and features based on bidirectional LSTMs, has value not just for UCCA parsing: its ability to handle more general graph structures can inform the development of parsers for other semantic DAG structures, and in languages that frequently use discontinuous structures.

1 Introduction

Universal Conceptual Cognitive Annotation (UCCA, Abend and Rappoport, 2013) is a cross-linguistically applicable semantic representation scheme, building on the established Basic Linguistic Theory typological framework (Dixon, 2010a,b, 2012), and Cognitive Linguistics literature (Croft and Cruse, 2004). It has demonstrated applicability to multiple languages, including English, French, German and Czech, support for rapid annotation by non-experts (assisted by an accessible annotation interface (Abend et al., 2017)), and stability under translation (Sulem et al., 2015). It has also proven useful for machine translation evaluation (Birch et al., 2016). UCCA differs from syntactic schemes in terms of content and formal structure. It exhibits reentrancy, discontinuous nodes and non-terminals, which no single existing parser supports. Lacking a parser, UCCA's applicability has been so far limited, a gap this work addresses.

We present the first UCCA parser, TUPA (Transition-based UCCA Parser), building on recent advances in discontinuous constituency and dependency graph parsing, and further introducing novel transitions and features for UCCA. Transition-based techniques are a natural starting point for UCCA parsing, given the conceptual similarity of UCCA's distinctions, centered around predicate-argument structures, to distinctions expressed by dependency schemes, and the achievements of transition-based methods in dependency parsing (Dyer et al., 2015; Andor et al., 2016; Kiperwasser and Goldberg, 2016). We are further motivated by the strength of transition-based methods in related tasks, including dependency graph parsing (Sagae and Tsujii, 2008; Ribeyre et al., 2014; Tokgöz and Eryiğit, 2015), constituency parsing (Sagae and Lavie, 2005; Zhang and Clark, 2009; Zhu et al., 2013; Maier, 2015; Maier and Lichte, 2016), AMR parsing (Wang et al., 2015a,b, 2016; Misra and Artzi, 2016; Goodman et al., 2016; Zhou et al., 2016; Damonte et al., 2017) and CCG parsing (Zhang and Clark, 2011; Ambati et al., 2015, 2016).

We evaluate TUPA on the English UCCA corpora, including in-domain and out-of-domain settings. To assess the ability of existing parsers to tackle the task, we develop a conversion procedure from UCCA to bilexical graphs and trees. Results show superior performance for TUPA, demonstrating the effectiveness of the presented approach.[1]

The rest of the paper is structured as follows:

[1]All parsing and conversion code, as well as trained parser models, are available at https://github.com/danielhers/tupa.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1127–1138
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1104

Section 2 describes UCCA in more detail. Section 3 introduces TUPA. Section 4 discusses the data and experimental setup. Section 5 presents the experimental results. Section 6 summarizes related work, and Section 7 concludes the paper.

2 The UCCA Scheme

UCCA graphs are labeled, directed acyclic graphs (DAGs), whose leaves correspond to the tokens of the text. A node (or *unit*) corresponds to a terminal or to several terminals (not necessarily contiguous) viewed as a single entity according to semantic or cognitive considerations. Edges bear a category, indicating the role of the sub-unit in the parent relation. Figure 1 presents a few examples.

UCCA is a multi-layered representation, where each layer corresponds to a "module" of semantic distinctions. UCCA's *foundational layer*, targeted in this paper, covers the predicate-argument structure evoked by predicates of all grammatical categories (verbal, nominal, adjectival and others), the inter-relations between them, and other major linguistic phenomena such as coordination and multi-word expressions. The layer's basic notion is the *scene*, describing a state, action, movement or some other relation that evolves in time. Each scene contains one main relation (marked as either a Process or a State), as well as one or more Participants. For example, the sentence "After graduation, John moved to Paris" (Figure 1a) contains two scenes, whose main relations are "graduation" and "moved". "John" is a Participant in both scenes, while "Paris" only in the latter. Further categories account for inter-scene relations and the internal structure of complex arguments and relations (e.g. coordination, multi-word expressions and modification).

One incoming edge for each non-root node is marked as *primary*, and the rest (mostly used for implicit relations and arguments) as *remote* edges, a distinction made by the annotator. The primary edges thus form a tree structure, whereas the remote edges enable reentrancy, forming a DAG.

While parsing technology in general, and transition-based parsing in particular, is well-established for syntactic parsing, UCCA has several distinct properties that distinguish it from syntactic representations, mostly UCCA's tendency to abstract away from syntactic detail that do not affect argument structure. For instance, consider the following examples where the concept of a scene

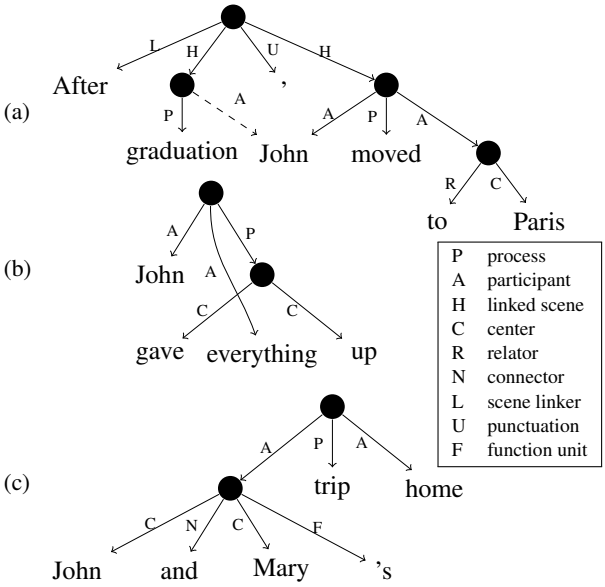


Figure 1: UCCA structures demonstrating three structural properties exhibited by the scheme. (a) includes a remote edge (dashed), resulting in "John" having two parents. (b) includes a discontinuous unit ("gave ... up"). (c) includes a coordination construction ("John and Mary"). Pre-terminal nodes are omitted for brevity. Right: legend of edge labels.

has a different rationale from the syntactic concept of a clause. First, non-verbal predicates in UCCA are represented like verbal ones, such as when they appear in copula clauses or noun phrases. Indeed, in Figure 1a, "graduation" and "moved" are considered separate events, despite appearing in the same clause. Second, in the same example, "John" is marked as a (remote) Participant in the graduation scene, despite not being overtly marked. Third, consider the possessive construction in Figure 1c. While in UCCA "trip" evokes a scene in which "John and Mary" is a Participant, a syntactic scheme would analyze this phrase similarly to "John and Mary's shoes".

These examples demonstrate that a UCCA parser, and more generally semantic parsers, face an additional level of ambiguity compared to their syntactic counterparts (e.g., "after graduation" is formally very similar to "after 2pm", which does not evoke a scene). Section 6 discusses UCCA in the context of other semantic schemes, such as AMR (Banarescu et al., 2013).

Alongside recent progress in dependency parsing into projective trees, there is increasing interest in parsing into representations with more general structural properties (see Section 6). One such property is *reentrancy*, namely the sharing of semantic units between predicates. For instance, in Figure 1a, "John" is an argument of both "gradu-

ation" and "moved", yielding a DAG rather than a tree. A second property is *discontinuity*, as in Figure 1b, where "gave up" forms a discontinuous semantic unit. Discontinuities are pervasive, e.g., with multi-word expressions (Schneider et al., 2014). Finally, unlike most dependency schemes, UCCA uses *non-terminal nodes* to represent units comprising more than one word. The use of non-terminal nodes is motivated by constructions with no clear head, including coordination structures (e.g., "John and Mary" in Figure 1c), some multi-word expressions (e.g., "The Haves and the *Have Nots*"), and prepositional phrases (either the preposition or the head noun can serve as the constituent's head). To our knowledge, no existing parser supports all structural properties required for UCCA parsing.

3 Transition-based UCCA Parsing

We now turn to presenting TUPA. Building on previous work on parsing reentrancies, discontinuities and non-terminal nodes, we define an extended set of transitions and features that supports the conjunction of these properties.

Transition-based parsers (Nivre, 2003) scan the text from start to end, and create the parse incrementally by applying a *transition* at each step to the parser's state, defined using three data structures: a buffer B of tokens and nodes to be processed, a stack S of nodes currently being processed, and a graph $G = (V, E, \ell)$ of constructed nodes and edges, where V is the set of *nodes*, E is the set of *edges*, and $\ell : E \to L$ is the *label* function, L being the set of possible labels. Some states are marked as *terminal*, meaning that G is the final output. A classifier is used at each step to select the next transition based on features encoding the parser's current state. During training, an oracle creates training instances for the classifier, based on gold-standard annotations.

Transition Set. Given a sequence of tokens $w_1, \ldots, w_n$, we predict a UCCA graph G over the sequence. Parsing starts with a single node on the stack (an artificial root node), and the input tokens in the buffer. Figure 2 shows the transition set.

In addition to the standard SHIFT and REDUCE operations, we follow previous work in transition-based constituency parsing (Sagae and Lavie, 2005), adding the NODE transition for creating new non-terminal nodes. For every $X \in L$, NODE$_X$ creates a new node on the buffer as a parent of the first element on the stack, with an X-labeled edge. LEFT-EDGE$_X$ and RIGHT-EDGE$_X$ create a new primary X-labeled edge between the first two elements on the stack, where the parent is the left or the right node, respectively. As a UCCA node may only have one incoming primary edge, EDGE transitions are disallowed if the child node already has an incoming primary edge. LEFT-REMOTE$_X$ and RIGHT-REMOTE$_X$ do not have this restriction, and the created edge is additionally marked as *remote*. We distinguish between these two pairs of transitions to allow the parser to create remote edges without the possibility of producing invalid graphs. To support the prediction of multiple parents, node and edge transitions leave the stack unchanged, as in other work on transition-based dependency graph parsing (Sagae and Tsujii, 2008; Ribeyre et al., 2014; Tokgöz and Eryiğit, 2015). REDUCE pops the stack, to allow removing a node once all its edges have been created. To handle discontinuous nodes, SWAP pops the second node on the stack and adds it to the top of the buffer, as with the similarly named transition in previous work (Nivre, 2009; Maier, 2015). Finally, FINISH pops the root node and marks the state as terminal.

Classifier. The choice of classifier and feature representation has been shown to play an important role in transition-based parsing (Chen and Manning, 2014; Andor et al., 2016; Kiperwasser and Goldberg, 2016). To investigate the impact of the type of transition classifier in UCCA parsing, we experiment with three different models.

1. Starting with a simple and common choice (e.g., Maier and Lichte, 2016), **TUPA$_{\text{Sparse}}$** uses a linear classifier with sparse features, trained with the averaged structured perceptron algorithm (Collins and Roark, 2004) and MIN-UPDATE (Goldberg and Elhadad, 2011): each feature requires a minimum number of updates in training to be included in the model.[2]

2. Changing the model to a feedforward neural network with dense embedding features, **TUPA$_{\text{MLP}}$** ("multi-layer perceptron"), uses an architecture similar to that of Chen and Manning (2014), but with two rectified linear layers

[2] We also experimented with a linear model using dense embedding features, trained with the averaged structured perceptron algorithm. It performed worse than the sparse perceptron model and was hence discarded.

Before Transition				Transition	After Transition					Condition
Stack	Buffer	Nodes	Edges		Stack	Buffer	Nodes	Edges	Terminal?	
S	$x \mid B$	V	E	SHIFT	$S \mid x$	B	V	E	$-$	
$S \mid x$	B	V	E	REDUCE	S	B	V	E	$-$	
$S \mid x$	B	V	E	NODE$_X$	$S \mid x$	$y \mid B$	$V \cup \{y\}$	$E \cup \{(y,x)_X\}$	$-$	$x \neq \text{root}$
$S \mid y, x$	B	V	E	LEFT-EDGE$_X$	$S \mid y, x$	B	V	$E \cup \{(x,y)_X\}$	$-$	$\begin{cases} x \notin w_{1:n}, \\ y \neq \text{root}, \\ y \not\rightsquigarrow_G x \end{cases}$
$S \mid x, y$	B	V	E	RIGHT-EDGE$_X$	$S \mid x, y$	B	V	$E \cup \{(x,y)_X\}$	$-$	
$S \mid y, x$	B	V	E	LEFT-REMOTE$_X$	$S \mid y, x$	B	V	$E \cup \{(x,y)^*_X\}$	$-$	
$S \mid x, y$	B	V	E	RIGHT-REMOTE$_X$	$S \mid x, y$	B	V	$E \cup \{(x,y)^*_X\}$	$-$	
$S \mid x, y$	B	V	E	SWAP	$S \mid y$	$x \mid B$	V	E	$-$	$\mathrm{i}(x) < \mathrm{i}(y)$
[root]	$\emptyset$	V	E	FINISH	$\emptyset$	$\emptyset$	V	E	$+$	

Figure 2: The transition set of TUPA. We write the stack with its top to the right and the buffer with its head to the left. $(\cdot,\cdot)_X$ denotes a primary X-labeled edge, and $(\cdot,\cdot)^*_X$ a remote X-labeled edge. $\mathrm{i}(x)$ is a running index for the created nodes. In addition to the specified conditions, the prospective child in an EDGE transition must not already have a primary parent.

instead of one layer with cube activation. The embeddings and classifier are trained jointly.

3. Finally, **TUPA$_{\text{BiLSTM}}$** uses a bidirectional LSTM for feature representation, on top of the dense embedding features, an architecture similar to Kiperwasser and Goldberg (2016). The BiLSTM runs on the input tokens in forward and backward directions, yielding a vector representation that is then concatenated with dense features representing the parser state (e.g., existing edge labels and previous parser actions; see below). This representation is then fed into a feedforward network similar to TUPA$_{\text{MLP}}$. The feedforward layers, BiLSTM and embeddings are all trained jointly.

For all classifiers, inference is performed greedily, i.e., without beam search. Hyperparameters are tuned on the development set (see Section 4).

Features. TUPA$_{\text{Sparse}}$ uses binary indicator features representing the words, POS tags, syntactic dependency labels and existing edge labels related to the top four stack elements and the next three buffer elements, in addition to their children and grandchildren in the graph. We also use bi- and trigram features based on these values (Zhang and Clark, 2009; Zhu et al., 2013), features related to discontinuous nodes (Maier, 2015, including separating punctuation and gap type), features representing existing edges and the number of parents and children, as well as the past actions taken by the parser. In addition, we use use a novel, UCCA-specific feature: number of remote children.[3]

For TUPA$_{\text{MLP}}$ and TUPA$_{\text{BiLSTM}}$, we replace all indicator features by a concatenation of the vector embeddings of all represented elements: words, POS tags, syntactic dependency labels, edge labels, punctuation, gap type and parser actions. These embeddings are initialized randomly. We additionally use external word embeddings initialized with pre-trained word2vec vectors (Mikolov et al., 2013),[4] updated during training. In addition to dropout between NN layers, we apply word dropout (Kiperwasser and Goldberg, 2016): with a certain probability, the embedding for a word is replaced with a zero vector. We do not apply word dropout to the external word embeddings.

Finally, for all classifiers we add a novel real-valued feature to the input vector, **ratio**, corresponding to the ratio between the number of terminals to number of nodes in the graph G. This feature serves as a regularizer for the creation of new nodes, and should be beneficial for other transition-based constituency parsers too.

Training. For training the transition classifiers, we use a dynamic oracle (Goldberg and Nivre, 2012), i.e., an oracle that outputs a set of optimal transitions: when applied to the current parser state, the gold standard graph is reachable from the resulting state. For example, the oracle would predict a NODE transition if the stack has on its top a parent in the gold graph that has not been created, but would predict a RIGHT-EDGE transition if the second stack element is a parent of the first element according to the gold graph and the edge between them has not been created. The transition predicted by the classifier is deemed correct and is applied to the parser state to reach the subsequent state, if the transition is included in the set of optimal transitions. Otherwise, a random optimal transition is applied, and for the perceptron-based parser, the classifier's weights are updated

[3] See Appendix A for a full list of used feature templates.

[4] https://goo.gl/6ovEhC

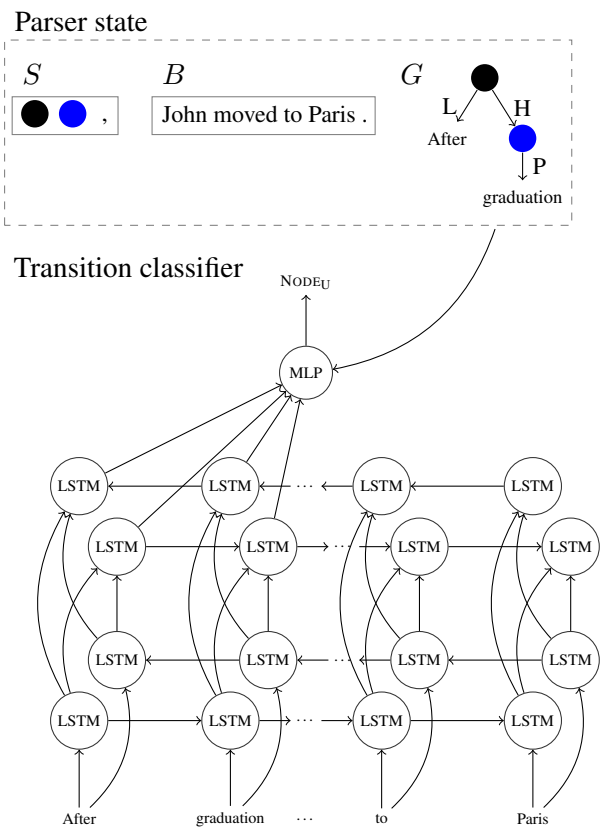


Figure 3: Illustration of the TUPA model. Top: parser state (stack, buffer and intermediate graph). Bottom: $\text{TUPA}_{\text{BiLTSM}}$ architecture. Vector representation for the input tokens is computed by two layers of bidirectional LSTMs. The vectors for specific tokens are concatenated with embedding and numeric features from the parser state (for existing edge labels, number of children, etc.), and fed into the MLP for selecting the next transition.

according to the perceptron update rule.

POS tags and syntactic dependency labels are extracted using spaCy (Honnibal and Johnson, 2015).[5] We use the categorical cross-entropy objective function and optimize the NN classifiers with the Adam optimizer (Kingma and Ba, 2014).

4 Experimental Setup

Data. We conduct our experiments on the UCCA Wikipedia corpus (henceforth, *Wiki*), and use the English part of the UCCA *Twenty Thousand Leagues Under the Sea* English-French parallel corpus (henceforth, *20K Leagues*) as out-of-domain data.[6] Table 1 presents some statistics for the two corpora. We use passages of indices up to 676 of the *Wiki* corpus as our training set, passages 688–808 as development set, and passages 942–1028 as in-domain test set. While

[5] https://spacy.io

[6] http://cs.huji.ac.il/~oabend/ucca.html

	Wiki			20K
	Train	Dev	Test	Leagues
# passages	300	34	33	154
# sentences	4268	454	503	506
# nodes	298,993	33,704	35,718	29,315
% terminal	42.96	43.54	42.87	42.09
% non-term.	58.33	57.60	58.35	60.01
% discont.	0.54	0.53	0.44	0.81
% reentrant	2.38	1.88	2.15	2.03
# edges	287,914	32,460	34,336	27,749
% primary	98.25	98.75	98.74	97.73
% remote	1.75	1.25	1.26	2.27
Average per non-terminal node				
# children	1.67	1.68	1.66	1.61

Table 1: Statistics of the *Wiki* and *20K Leagues* UCCA corpora. All counts exclude the root node, implicit nodes, and linkage nodes and edges.

UCCA edges can cross sentence boundaries, we adhere to the common practice in semantic parsing and train our parsers on individual sentences, discarding inter-relations between them (0.18% of the edges). We also discard linkage nodes and edges (as they often express inter-sentence relations and are thus mostly redundant when applied at the sentence level) as well as implicit nodes.[7] In the out-of-domain experiments, we apply the same parsers (trained on the *Wiki* training set) to the *20K Leagues* corpus without parameter re-tuning.

Implementation. We use the DyNet package (Neubig et al., 2017) for implementing the NN classifiers. Unless otherwise noted, we use the default values provided by the package. See Appendix C for the hyperparameter values we found by tuning on the development set.

Evaluation. We define a simple measure for comparing UCCA structures $G_p = (V_p, E_p, \ell_p)$ and $G_g = (V_g, E_g, \ell_g)$, the predicted and gold-standard graphs, respectively, over the same sequence of terminals $W = \{w_1, \ldots, w_n\}$. For an edge $e = (u, v)$ in either graph, u being the parent and v the child, its yield $y(e) \subseteq W$ is the set of terminals in W that are descendants of v. Define the set of *mutual edges* between G_p and G_g:

$$M(G_p, G_g) = \{(e_1, e_2) \in E_p \times E_g \mid y(e_1) = y(e_2) \wedge \ell_p(e_1) = \ell_g(e_2)\}$$

Labeled precision and recall are defined by dividing $|M(G_p, G_g)|$ by $|E_p|$ and $|E_g|$, respectively, and F-score by taking their harmonic mean.

[7] Appendix B further discusses linkage and implicit units.

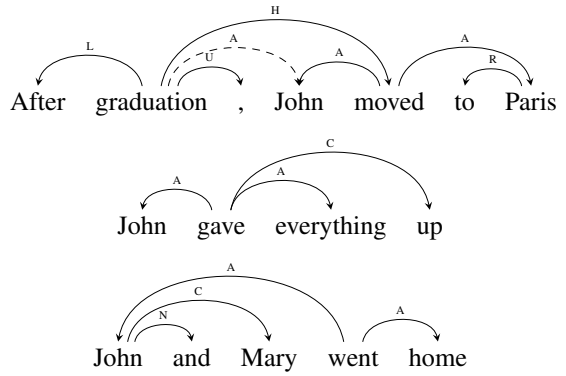


Figure 4: Bilexical graph approximation (dependency graph) for the sentences in Figure 1.

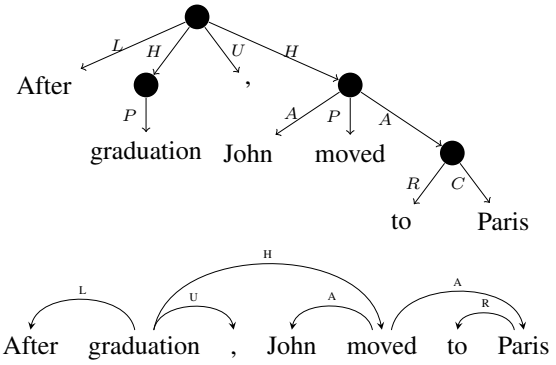


Figure 5: Tree approximation (constituency) for the sentence in Figure 1a (top), and bilexical tree approximation (dependency) for the same sentence (bottom). These are identical to the original graphs, apart from the removal of remote edges.

We report two variants of this measure: one where we consider only primary edges, and another for remote edges (see Section 2). Performance on remote edges is of pivotal importance in this investigation, which focuses on extending the class of graphs supported by statistical parsers.

We note that the measure collapses to the standard PARSEVAL constituency evaluation measure if G_p and G_g are trees. Punctuation is excluded from the evaluation, but not from the datasets.

Comparison to bilexical graph parsers. As no direct comparison with existing parsers is possible, we compare TUPA to bilexical dependency graph parsers, which support reentrancy and discontinuity but not non-terminal nodes.

To facilitate the comparison, we convert our training set into bilexical graphs (see examples in Figure 4), train each of the parsers, and evaluate them by applying them to the test set and then reconstructing UCCA graphs, which are compared with the gold standard. The conversion to bilexical graphs is done by heuristically selecting a head terminal for each non-terminal node, and attaching all terminal descendents to the head terminal. In the inverse conversion, we traverse the bilexical graph in topological order, creating non-terminal parents for all terminals, and attaching them to the previously-created non-terminals corresponding to the bilexical heads.[8]

In Section 5 we report the upper bounds on the achievable scores due to the error resulting from the removal of non-terminal nodes.

Comparison to tree parsers. For completeness, and as parsing technology is considerably more mature for tree (rather than graph) parsing, we also perform a *tree approximation* experiment, converting UCCA to (bilexical) trees and evaluating constituency and dependency tree parsers on them (see examples in Figure 5). Our approach is similar to the tree approximation approach used for dependency graph parsing (Agić et al., 2015; Fernández-González and Martins, 2015), where dependency graphs were converted into dependency trees and then parsed by dependency tree parsers. In our setting, the conversion to trees consists simply of removing remote edges from the graph, and then to bilexical trees by applying the same procedure as for bilexical graphs.

Baseline parsers. We evaluate two bilexical graph semantic dependency parsers: DAGParser (Ribeyre et al., 2014), the leading transition-based parser in SemEval 2014 (Oepen et al., 2014) and TurboParser (Almeida and Martins, 2015), a graph-based parser from SemEval 2015 (Oepen et al., 2015); UPARSE (Maier and Lichte, 2016), a transition-based constituency parser supporting discontinuous constituents; and two bilexical tree parsers: MaltParser (Nivre et al., 2007), and the stack LSTM-based parser of Dyer et al. (2015, henceforce "LSTM Parser"). Default settings are used in all cases.[9] DAGParser and UPARSE use beam search by default, with a beam size of 5 and 4 respectively. The other parsers are greedy.

5 Results

Table 2 presents our main experimental results, as well as upper bounds for the baseline parsers, re-

[8] See Appendix D for a detailed description of the conversion procedures.

[9] For MaltParser we use the ARCEAGER transition set and SVM classifier. Other configurations yielded lower scores.

	Wiki (in-domain)						20K Leagues (out-of-domain)					
	Primary			Remote			Primary			Remote		
	LP	**LR**	**LF**	**LP**	**LR**	**LF**	**LP**	**LR**	**LF**	**LP**	**LR**	**LF**
$\text{TUPA}_{\text{Sparse}}$	64.5	63.7	64.1	19.8	13.4	16	59.6	59.9	59.8	22.2	7.7	11.5
TUPA_{MLP}	65.2	64.6	64.9	23.7	13.2	16.9	62.3	62.6	62.5	20.9	6.3	9.7
$\text{TUPA}_{\text{BiLSTM}}$	74.4	72.7	**73.5**	47.4	51.6	**49.4**	68.7	68.5	**68.6**	38.6	18.8	**25.3**
Bilexical Approximation (Dependency DAG Parsers)												
Upper Bound			91			58.3			91.3			43.4
DAGParser	61.8	55.8	58.6	9.5	0.5	1	56.4	50.6	53.4	–	0	0
TurboParser	57.7	46	51.2	77.8	1.8	3.7	50.3	37.7	43.1	100	0.4	0.8
Tree Approximation (Constituency Tree Parser)												
Upper Bound			100			–			100			–
UPARSE	60.9	61.2	61.1	–	–	–	52.7	52.8	52.8	–	–	–
Bilexical Tree Approximation (Dependency Tree Parsers)												
Upper Bound			91			–			91.3			–
MaltParser	62.8	57.7	60.2	–	–	–	57.8	53	55.3	–	–	–
LSTM Parser	73.2	66.9	69.9	–	–	–	66.1	61.1	63.5	–	–	–

Table 2: Experimental results, in percents, on the *Wiki* test set (left) and the *20K Leagues* set (right). Columns correspond to labeled precision, recall and F-score, for both primary and remote edges. F-score upper bounds are reported for the conversions. For the tree approximation experiments, only primary edges scores are reported, as they are unable to predict remote edges. $\text{TUPA}_{\text{BiLSTM}}$ obtains the highest F-scores in all metrics, surpassing the bilexical parsers, tree parsers and other classifiers.

flecting the error resulting from the conversion.[10]

DAGParser and UPARSE are most directly comparable to $\text{TUPA}_{\text{Sparse}}$, as they also use a perceptron classifier with sparse features. $\text{TUPA}_{\text{Sparse}}$ considerably outperforms both, where DAGParser does not predict any remote edges in the out-of-domain setting. TurboParser fares worse in this comparison, despite somewhat better results on remote edges. The LSTM parser of Dyer et al. (2015) obtains the highest primary F-score among the baseline parsers, with a considerable margin.

Using a feedforward NN and embedding features, TUPA_{MLP} obtains higher scores than $\text{TUPA}_{\text{Sparse}}$, but is outperformed by the LSTM parser on primary edges. However, using better input encoding allowing virtual look-ahead and look-behind in the token representation, $\text{TUPA}_{\text{BiLSTM}}$ obtains substantially higher scores than TUPA_{MLP} and all other parsers, on both primary and remote edges, both in the in-domain and out-of-domain settings. Its performance in absolute terms, of 73.5% F-score on primary edges, is encouraging in light of UCCA's inter-annotator agreement of 80–85% F-score on them (Abend and Rappoport, 2013).

The parsers resulting from tree approximation are unable to recover any remote edges, as these are removed in the conversion.[11] The bilexical DAG parsers are quite limited in this respect as well. While some of the DAG parsers' difficulty can be attributed to the conversion upper bound of 58.3%, this in itself cannot account for their poor performance on remote edges, which is an order of magnitude lower than that of $\text{TUPA}_{\text{BiLSTM}}$.

6 Related Work

While earlier work on anchored[12] semantic parsing has mostly concentrated on shallow semantic analysis, focusing on semantic role labeling of verbal argument structures, the focus has recently shifted to parsing of more elaborate representations that account for a wider range of phenomena (Abend and Rappoport, 2017).

Grammar-Based Parsing. Linguistically expressive grammars such as HPSG (Pollard and Sag, 1994), CCG (Steedman, 2000) and TAG (Joshi and Schabes, 1997) provide a theory of the syntax-semantics interface, and have been used as a basis for semantic parsers by defining com-

[10] The low upper bound for remote edges is partly due to the removal of implicit nodes (not supported in bilexical representations), where the whole sub-graph headed by such nodes, often containing remote edges, must be discarded.

[11] We also experimented with a simpler version of TUPA lacking REMOTE transitions, obtaining an increase of up to 2 labeled F-score points on primary edges, at the cost of not being able to predict remote edges.

[12] By *anchored* we mean that the semantic representation directly corresponds to the words and phrases of the text.

positional semantics on top of them (Flickinger, 2000; Bos, 2005, among others). Depending on the grammar and the implementation, such semantic parsers can support some or all of the structural properties UCCA exhibits. Nevertheless, this line of work differs from our approach in two important ways. First, the *representations* are different. UCCA does not attempt to model the syntax-semantics interface and is thus less coupled with syntax. Second, while grammar-based *parsers* explicitly model syntax, our approach directly models the relation between tokens and semantic structures, without explicit composition rules.

Broad-Coverage Semantic Parsing. Most closely related to this work is Broad-Coverage Semantic Dependency Parsing (SDP), addressed in two SemEval tasks (Oepen et al., 2014, 2015). Like UCCA parsing, SDP addresses a wide range of semantic phenomena, and supports discontinuous units and reentrancy. In SDP, however, bilexical dependencies are used, and a head must be selected for every relation—even in constructions that have no clear head, such as coordination (Ivanova et al., 2012). The use of non-terminal nodes is a simple way to avoid this liability. SDP also differs from UCCA in the type of distinctions it makes, which are more tightly coupled with syntactic considerations, where UCCA aims to capture purely semantic cross-linguistically applicable notions. For instance, the "poss" label in the DM target representation is used to annotate syntactic possessive constructions, regardless of whether they correspond to semantic ownership (e.g., "John's dog") or other semantic relations, such as marking an argument of a nominal predicate (e.g., "John's kick"). UCCA reflects the difference between these constructions.

Recent interest in SDP has yielded numerous works on graph parsing (Ribeyre et al., 2014; Thomson et al., 2014; Almeida and Martins, 2015; Du et al., 2015), including tree approximation (Agić and Koller, 2014; Schluter et al., 2014) and joint syntactic/semantic parsing (Henderson et al., 2013; Swayamdipta et al., 2016).

Abstract Meaning Representation. Another line of work addresses parsing into AMRs (Flanigan et al., 2014; Vanderwende et al., 2015; Pust et al., 2015; Artzi et al., 2015), which, like UCCA, abstract away from syntactic distinctions and represent meaning directly, using OntoNotes predicates (Weischedel et al., 2013). Events in AMR may also be evoked by non-verbal predicates, including possessive constructions.

Unlike in UCCA, the alignment between AMR concepts and the text is not explicitly marked. While sharing much of this work's motivation, not anchoring the representation in the text complicates the parsing task, as it requires the alignment to be automatically (and imprecisely) detected. Indeed, despite considerable technical effort (Flanigan et al., 2014; Pourdamghani et al., 2014; Werling et al., 2015), concept identification is only about 80%–90% accurate. Furthermore, anchoring allows breaking down sentences into semantically meaningful sub-spans, which is useful for many applications (Fernández-González and Martins, 2015; Birch et al., 2016).

Several transition-based AMR parsers have been proposed: CAMR assumes syntactically parsed input, processing dependency trees into AMR (Wang et al., 2015a,b, 2016; Goodman et al., 2016). In contrast, the parsers of Damonte et al. (2017) and Zhou et al. (2016) do not require syntactic pre-processing. Damonte et al. (2017) perform concept identification using a simple heuristic selecting the most frequent graph for each token, and Zhou et al. (2016) perform concept identification and parsing jointly. UCCA parsing does not require separately aligning the input tokens to the graph. TUPA creates non-terminal units as part of the parsing process.

Furthermore, existing transition-based AMR parsers are not general DAG parsers. They are only able to predict a subset of reentrancies and discontinuities, as they may remove nodes before their parents have been predicted (Damonte et al., 2017). They are thus limited to a sub-class of AMRs in particular, and specifically cannot produce arbitrary DAG parses. TUPA's transition set, on the other hand, allows general DAG parsing.[13]

7 Conclusion

We present TUPA, the first parser for UCCA. Evaluated in in-domain and out-of-domain settings, we show that coupled with a NN classifier and BiLSTM feature extractor, it accurately predicts UCCA graphs from text, outperforming a variety of strong baselines by a margin.

Despite the recent diversity of semantic pars-

[13] See Appendix E for a proof sketch for the completeness of TUPA's transition set.

ing work, the effectiveness of different approaches for structurally and semantically different schemes is not well-understood (Kuhlmann and Oepen, 2016). Our contribution to this literature is a general parser that supports multiple parents, discontinuous units and non-terminal nodes.

Future work will evaluate TUPA in a multilingual setting, assessing UCCA's cross-linguistic applicability. We will also apply the TUPA transition scheme to different target representations, including AMR and SDP, exploring the limits of its generality. In addition, we will explore different conversion procedures (Kong et al., 2015) to compare different representations, suggesting ways for a data-driven design of semantic annotation.

A parser for UCCA will enable using the framework for new tasks, in addition to existing applications such as machine translation evaluation (Birch et al., 2016). We believe UCCA's merits in providing a cross-linguistically applicable, broad-coverage annotation will support ongoing efforts to incorporate deeper semantic structures into various applications, such as sentence simplification (Narayan and Gardent, 2014) and summarization (Liu et al., 2015).

Acknowledgments

This work was supported by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI). The first author was supported by a fellowship from the Edmond and Lily Safra Center for Brain Sciences. We thank Wolfgang Maier, Nathan Schneider, Elior Sulem and the anonymous reviewers for their helpful comments.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL*. pages 228–238. http://aclweb.org/anthology/P13-1023.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proc. of ACL*. To appear.

Omri Abend, Shai Yerushalmi, and Ari Rappoport. 2017. UCCAApp: Web-application for syntactic and semantic phrase-based annotation. In *Proc. of ACL: System Demonstration Papers*. To appear.

Željko Agić and Alexander Koller. 2014. Potsdam: Semantic dependency parsing by bidirectional graph-tree transformations and syntactic parsing. In *Proc. of SemEval*. pages 465–470. http://aclweb.org/anthology/S14-2081.

Željko Agić, Alexander Koller, and Stephan Oepen. 2015. Semantic dependency graph parsing using tree approximations. In *Proc. of IWCS*. pages 217–227. http://aclweb.org/anthology/W15-0126.

Mariana S. C. Almeida and André F. T. Martins. 2015. Lisbon: Evaluating TurboSemanticParser on multiple languages and out-of-domain data. In *Proc. of SemEval*. pages 970–973. http://aclweb.org/anthology/S15-2162.

Bharat Ram Ambati, Tejaswini Deoskar, Mark Johnson, and Mark Steedman. 2015. An incremental algorithm for transition-based CCG parsing. In *Proc. of NAACL*. pages 53–63. http://aclweb.org/anthology/N15-1006.

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2016. Shift-reduce CCG parsing using neural network models. In *Proc. of NAACL-HLT*. pages 447–453. http://aclweb.org/anthology/N16-1052.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of ACL*. pages 2442–2452. http://aclweb.org/anthology/P16-1231.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proc. of EMNLP*. pages 1699–1710. http://aclweb.org/anthology/D15-1198.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proc. of the Linguistic Annotation Workshop*. http://aclweb.org/anthology/W13-2322.

Alexandra Birch, Omri Abend, Ondřej Bojar, and Barry Haddow. 2016. HUME: Human UCCA-based evaluation of machine translation. In *Proc. of EMNLP*. pages 1264–1274. http://aclweb.org/anthology/D16-1134.

Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *Proc. of IWCS*. volume 6, pages 42–53. http://www.let.rug.nl/bos/pubs/Bos2005IWCS.pdf.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*. pages 740–750. http://aclweb.org/anthology/D14-1082.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. of ACL*. pages 111–118. http://aclweb.org/anthology/P04-1015.

William Croft and D Alan Cruse. 2004. *Cognitive linguistics*. Cambridge University Press.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of EACL*. http://homepages.inf.ed.ac.uk/scohen/eacl17amr.pdf.

Robert M. W. Dixon. 2010a. *Basic Linguistic Theory: Grammatical Topics*, volume 2. Oxford University Press.

Robert M. W. Dixon. 2010b. *Basic Linguistic Theory: Methodology*, volume 1. Oxford University Press.

Robert M. W. Dixon. 2012. *Basic Linguistic Theory: Further Grammatical Topics*, volume 3. Oxford University Press.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015. Peking: Building semantic dependency graphs with a hybrid parser. In *Proc. of SemEval*. pages 927–931. http://aclweb.org/anthology/S15-2154.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependeny parsing with stack long short-term memory. In *Proc. of ACL*. pages 334–343. http://aclweb.org/anthology/P15-1033.

Daniel Fernández-González and André FT Martins. 2015. Parsing as reduction. In *Proc. of ACL*. pages 1523–1533. http://aclweb.org/anthology/P15-1147.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of ACL*. pages 1426–1436. http://aclweb.org/anthology/P14-1134.

Daniel Flickinger. 2000. On building a more efficient grammar by exploiting types. In *Collaborative Language Engineering*, CLSI, Stanford, CA, volume 6, pages 15–28.

Yoav Goldberg and Michael Elhadad. 2011. Learning sparser perceptron models. Technical report. http://www.cs.bgu.ac.il/˜yoavg/publications.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proc. of COLING*. pages 959–976. http://aclweb.org/anthology/C12-1059.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing. In *Proc. of ACL*. pages 1–11. http://aclweb.org/anthology/P16-1001.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics* 39(4):949–998. http://cognet.mit.edu/node/27348.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proc. of EMNLP*. pages 1373–1378. http://aclweb.org/anthology/D15-1162.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proc. of LAW*. pages 2–11. http://aclweb.org/anthology/W12-3602.

Aravind Joshi and Yves Schabes. 1997. Tree-Adjoining Grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, Springer, Berlin, volume 3, pages 69–124.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Lingpeng Kong, Alexander M. Rush, and Noah A. Smith. 2015. Transforming dependencies into phrase structures. In *Proc. of NAACL HLT*. https://aclweb.org/anthology/N15-1080.

Marco Kuhlmann and Stephan Oepen. 2016. Towards a catalogue of linguistic graph banks. *Computational Linguistics* https://mn.uio.no/ifi/english/people/aca/oe/cl.pdf.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proc. of NAACL*. pages 1077–1086. http://aclweb.org/anthology/N15-1114.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proc. of ACL*. pages 1202–1212. http://aclweb.org/anthology/P15-1116.

Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proc. of Workshop on Discontinuous Structures in NLP*. pages 47–57. http://aclweb.org/anthology/W16-0906.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. https://arxiv.org/pdf/1301.3781.

Dipendra K Misra and Yoav Artzi. 2016. Neural shift-reduce CCG semantic parsing. In *Proc. of EMNLP*. pages 1775–1786. http://aclweb.org/anthology/D16-1183.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proc. of ACL*. pages 435–445. http://aclweb.org/anthology/P14-1041.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* https://arxiv.org/abs/1701.03980.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT*. pages 149–160. http://aclweb.org/anthology/W06-2933.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proc. of ACL*. pages 351–359. http://aclweb.org/anthology/P09-1040.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering* 13(02):95–135.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*. pages 915–926. http://aclweb.org/anthology/S15-2153.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*. pages 63–72. http://aclweb.org/anthology/S14-2008.

Carl Pollard and Ivan Sag. 1994. *Head Driven Phrase Structure Grammar*. CSLI Publications, Stanford, CA.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with abstract meaning representation graphs. In *Proc. of EMNLP*. pages 425–429. http://aclweb.org/anthology/D14-1048.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into abstract meaning representation using syntax-based machine translation. In *Proc. of EMNLP*. pages 1143–1154. http://aclweb.org/anthology/D15-1136.

Corentin Ribeyre, Eric Villemonte de la Clergerie, and Djamé Seddah. 2014. Alpage: Transition-based semantic graph parsing with syntactic features. In *Proc. of SemEval*. pages 97–103. http://aclweb.org/anthology/S14-2012.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proc. of IWPT*. pages 125–132. http://aclweb.org/anthology/W05-1513.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proc. of COLING*. pages 753–760. http://aclweb.org/anthology/C08-1095.

Natalie Schluter, Anders Søgaard, Jakob Elming, Dirk Hovy, Barbara Plank, Héctor Martínez Alonso, Anders Johanssen, and Sigrid Klerke. 2014. Copenhagen-Malmö: Tree approximations of semantic parsing problems. In *Proc. of SemEval*. pages 213–217. http://aclweb.org/anthology/S14-2034.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A Smith. 2014. Discriminative lexical semantic segmentation with gaps: running the MWE gamut. *TACL* 2:193–206. http://aclweb.org/anthology/Q14-1016.pdf.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations: A French-English case study. In *Proc. of S2MT*. pages 11–22. http://aclweb.org/anthology/W15-3502.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proc. of CoNLL*. pages 187–197. http://aclweb.org/anthology/K16-1019.

Sam Thomson, Brendan O'Connor, Jeffrey Flanigan, David Bamman, Jesse Dodge, Swabha Swayamdipta, Nathan Schneider, Chris Dyer, and Noah A. Smith. 2014. CMU: Arc-factored, discriminative semantic dependency parsing. In *Proc. of SemEval*. pages 176–180. http://aclweb.org/anthology/S14-2027.

Alper Tokgöz and Gülsen Eryiğit. 2015. Transition-based dependency DAG parsing using dynamic oracles. In *Proc. of ACL Student Research Workshop*. pages 22–27. http://aclweb.org/anthology/P15-3004.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus. In *Proc. of NAACL*. pages 26–30. http://aclweb.org/anthology/N15-3006.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at SemEval-2016 task 8: An extended transition-based amr parser. In *Proc. of SemEval*. pages 1173–1178. http://aclweb.org/anthology/S16-1181.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proc. of ACL*. pages 857–862. http://aclweb.org/anthology/P15-2141.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proc. of NAACL*. pages 366–375. http://aclweb.org/anthology/N15-1040.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. OntoNotes release 5.0 LDC2013T19. *Linguistic Data Consortium, Philadelphia, PA* https://catalog.ldc.upenn.edu/LDC2013T19.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proc. of ACL*. pages 982–991. http://aclweb.org/anthology/P15-1095.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proc. of IWPT*. Association for Computational Linguistics, pages 162–171. http://aclweb.org/anthology/W09-3825.

Yue Zhang and Stephen Clark. 2011. Shift-reduce CCG parsing. In *Proc. of ACL*. pages 683–692. http://aclweb.org/anthology/P11-1069.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. 2016. AMR parsing with an incremental joint model. In *Proc. of EMNLP*. pages 680–689. http://aclweb.org/anthology/D16-1065.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proc. of ACL*. pages 434–443. http://aclweb.org/anthology/P13-1043.

Abstract Syntax Networks for Code Generation and Semantic Parsing

Maxim Rabinovich* **Mitchell Stern*** **Dan Klein**
Computer Science Division
University of California, Berkeley
{rabinovich,mitchell,klein}@cs.berkeley.edu

Abstract

Tasks like code generation and semantic parsing require mapping unstructured (or partially structured) inputs to well-formed, executable outputs. We introduce abstract syntax networks, a modeling framework for these problems. The outputs are represented as abstract syntax trees (ASTs) and constructed by a decoder with a dynamically-determined modular structure paralleling the structure of the output tree. On the benchmark HEARTHSTONE dataset for code generation, our model obtains 79.2 BLEU and 22.7% exact match accuracy, compared to previous state-of-the-art values of 67.1 and 6.1%. Furthermore, we perform competitively on the ATIS, JOBS, and GEO semantic parsing datasets with no task-specific engineering.

1 Introduction

Tasks like semantic parsing and code generation are challenging in part because they are *structured* (the output must be well-formed) but not *synchronous* (the output structure diverges from the input structure).

Sequence-to-sequence models have proven effective for both tasks (Dong and Lapata, 2016; Ling et al., 2016), using encoder-decoder frameworks to exploit the sequential structure on both the input and output side. Yet these approaches do not account for much richer structural constraints on outputs—including well-formedness, well-typedness, and executability. The well-formedness case is of particular interest, since it can readily be enforced by representing outputs as abstract syntax trees (ASTs) (Aho et al., 2006), an approach that can be seen as a much lighter weight


```
name: [
  'D', 'i', 'r', 'e', ' ',
  'W', 'o', 'l', 'f', ' ',
  'A', 'l', 'p', 'h', 'a']
cost: ['2']
type: ['Minion']
rarity: ['Common']
race: ['Beast']
class: ['Neutral']
description: [
  'Adjacent', 'minions', 'have',
  '+', '1', 'Attack', '.']
health: ['2']
attack: ['2']
durability: ['-1']
```

```
class DireWolfAlpha(MinionCard):
  def __init__(self):
    super().__init__(
      "Dire Wolf Alpha", 2, CHARACTER_CLASS.ALL,
      CARD_RARITY.COMMON, minion_type=MINION_TYPE.BEAST)
  def create_minion(self, player):
    return Minion(2, 2, auras=[
      Aura(ChangeAttack(1), MinionSelector(Adjacent()))
    ])
```

Figure 1: Example code for the "Dire Wolf Alpha" Hearthstone card.

```
show me the fare from ci0 to ci1

lambda $0 e
  ( exists $1 ( and ( from $1 ci0 )
                    ( to $1 ci1 )
                    ( = ( fare $1 ) $0 ) ) )
```

Figure 2: Example of a query and its logical form from the ATIS dataset. The `ci0` and `ci1` tokens are entity abstractions introduced in preprocessing (Dong and Lapata, 2016).

version of CCG-based semantic parsing (Zettlemoyer and Collins, 2005).

In this work, we introduce *abstract syntax networks* (ASNs), an extension of the standard encoder-decoder framework utilizing a modular decoder whose submodels are composed to natively generate ASTs in a top-down manner. The decoding process for any given input follows a dy-

*Equal contribution.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1139–1149
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1105

namically chosen mutual recursion between the modules, where the structure of the tree being produced mirrors the call graph of the recursion. We implement this process using a decoder model built of many submodels, each associated with a specific construct in the AST grammar and invoked when that construct is needed in the output tree. As is common with neural approaches to structured prediction (Chen and Manning, 2014; Vinyals et al., 2015), our decoder proceeds greedily and accesses not only a fixed encoding but also an attention-based representation of the input (Bahdanau et al., 2014).

Our model significantly outperforms previous architectures for code generation and obtains competitive or state-of-the-art results on a suite of semantic parsing benchmarks. On the HEARTHSTONE dataset for code generation, we achieve a token BLEU score of 79.2 and an exact match accuracy of 22.7%, greatly improving over the previous best results of 67.1 BLEU and 6.1% exact match (Ling et al., 2016).

The flexibility of ASNs makes them readily applicable to other tasks with minimal adaptation. We illustrate this point with a suite of semantic parsing experiments. On the JOBS dataset, we improve on previous state-of-the-art, achieving 92.9% exact match accuracy as compared to the previous record of 90.7%. Likewise, we perform competitively on the ATIS and GEO datasets, matching or exceeding the exact match reported by Dong and Lapata (2016), though not quite reaching the records held by the best previous semantic parsing approaches (Wang et al., 2014).

1.1 Related work

Encoder-decoder architectures, with and without attention, have been applied successfully both to sequence prediction tasks like machine translation and to tree prediction tasks like constituency parsing (Cross and Huang, 2016; Dyer et al., 2016; Vinyals et al., 2015). In the latter case, work has focused on making the task look like sequence-to-sequence prediction, either by flattening the output tree (Vinyals et al., 2015) or by representing it as a sequence of construction decisions (Cross and Huang, 2016; Dyer et al., 2016). Our work differs from both in its use of a recursive top-down generation procedure.

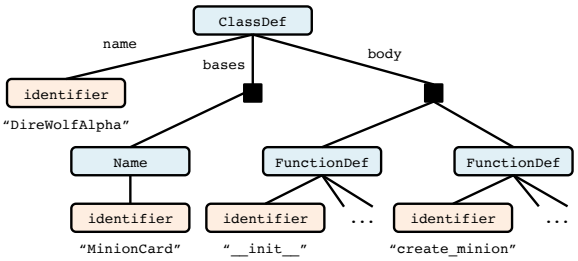
Dong and Lapata (2016) introduced a sequence-to-sequence approach to semantic parsing, including a limited form of top-down recursion, but without the modularity or tight coupling between output grammar and model characteristic of our approach.

Neural (and probabilistic) modeling of code, including for prediction problems, has a longer history. Allamanis et al. (2015) and Maddison and Tarlow (2014) proposed modeling code with a neural language model, generating concrete syntax trees in left-first depth-first order, focusing on metrics like perplexity and applications like code snippet retrieval. More recently, Shin et al. (2017) attacked the same problem using a grammar-based variational autoencoder with top-down generation similar to ours instead. Meanwhile, a separate line of work has focused on the problem of program induction from input-output pairs (Balog et al., 2016; Liang et al., 2010; Menon et al., 2013).

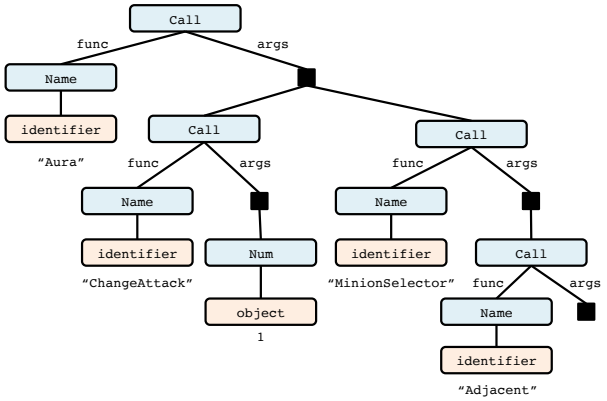
The prediction framework most similar in spirit to ours is the doubly-recurrent decoder network introduced by Alvarez-Melis and Jaakkola (2017), which propagates information down the tree using a vertical LSTM and between siblings using a horizontal LSTM. Our model differs from theirs in using a separate module for each grammar construct and learning separate vertical updates for siblings when the AST labels require all siblings to be jointly present; we do, however, use a horizontal LSTM for nodes with *variable* numbers of children. The differences between our models reflect not only design decisions, but also differences in data—since ASTs have labeled nodes and labeled edges, they come with additional structure that our model exploits.

Apart from ours, the best results on the code-generation task associated with the HEARTHSTONE dataset are based on a sequence-to-sequence approach to the problem (Ling et al., 2016). Abstract syntax networks greatly improve on those results.

Previously, Andreas et al. (2016) introduced neural module networks (NMNs) for visual question answering, with modules corresponding to linguistic substructures within the input query. The primary purpose of the modules in NMNs is to compute deep features of images in the style of convolutional neural networks (CNN). These features are then fed into a final decision layer. In contrast to the modules we describe here, NMN modules do not make decisions about what to generate or which modules to call next, nor do they



(a) The root portion of the AST.



(b) Excerpt from the same AST, corresponding to the code snippet `Aura(ChangeAttack(1),MinionSelector(Adjacent()))`.

Figure 3: Fragments from the abstract syntax tree corresponding to the example code in Figure 1. Blue boxes represent composite nodes, which expand via a constructor with a prescribed set of named children. Orange boxes represent primitive nodes, with their corresponding values written underneath. Solid black squares correspond to constructor fields with `sequential` cardinality, such as the body of a class definition (Figure 3a) or the arguments of a function call (Figure 3b).

maintain recurrent state.

2 Data Representation

2.1 Abstract Syntax Trees

Our model makes use of the Abstract Syntax Description Language (ASDL) framework (Wang et al., 1997), which represents code fragments as trees with typed nodes. *Primitive types* correspond to atomic values, like integers or identifiers. Accordingly, *primitive nodes* are annotated with a primitive type and a value of that type—for instance, in Figure 3a, the `identifier` node storing `"create_minion"` represents a function of the same name.

Composite types correspond to language constructs, like expressions or statements. Each type has a collection of *constructors*, each of which specifies the particular language construct a node of that type represents. Figure 4 shows constructors for the statement (`stmt`) and expression (`expr`) types. The associated language constructs include function and class definitions, return statements, binary operations, and function calls.

Composite types enter syntax trees via *composite nodes*, annotated with a composite type and a choice of constructor specifying how the node expands. The root node in Figure 3a, for example, is

```
primitive types: identifier, object, ...

stmt
 = FunctionDef(
     identifier name, arg* args, stmt* body)
 | ClassDef(
     identifier name, expr* bases, stmt* body)
 | Return(expr? value)
 | ...

expr
 = BinOp(expr left, operator op, expr right)
 | Call(expr func, expr* args)
 | Str(string s)
 | Name(identifier id, expr_context ctx)
 | ...

...
```

Figure 4: A simplified fragment of the Python ASDL grammar.[1]

a composite node of type `stmt` that represents a class definition and therefore uses the `ClassDef` constructor. In Figure 3b, on the other hand, the root uses the `Call` constructor because it represents a function call.

Children are specified by named and typed *fields* of the constructor, which have *cardinalities* of `singular`, `optional`, or `sequential`. By default, fields have `singular` cardinality, meaning they correspond to exactly one child. For instance, the `ClassDef` constructor has a `singular` name field of type `identifier`. Fields of `optional` cardinality are associ-

[1]The full grammar can be found online on the documentation page for the Python `ast` module: https://docs.python.org/3/library/ast.html#abstract-grammar

ated with zero or one children, while fields of `sequential` cardinality are associated with zero or more children—these are designated using `?` and $\star$ suffixes in the grammar, respectively. Fields of `sequential` cardinality are often used to represent statement blocks, as in the `body` field of the `ClassDef` and `FunctionDef` constructors.

The grammars needed for semantic parsing can easily be given ASDL specifications as well, using primitive types to represent variables, predicates, and atoms and composite types for standard logical building blocks like lambdas and counting (among others). Figure 2 shows what the resulting λ-calculus trees look like. The ASDL grammars for both λ-calculus and Prolog-style logical forms are quite compact, as Figures 9 and 10 in the appendix show.

2.2 Input Representation

We represent inputs as collections of named components, each of which consists of a sequence of tokens. In the case of semantic parsing, inputs have a single component containing the query sentence. In the case of HEARTHSTONE, the card's name and description are represented as sequences of characters and tokens, respectively, while categorical attributes are represented as single-token sequences. For HEARTHSTONE, we restrict our input and output vocabularies to values that occur more than once in the training set.

3 Model Architecture

Our model uses an encoder-decoder architecture with hierarchical attention. The key idea behind our approach is to structure the decoder as a collection of mutually recursive modules. The modules correspond to elements of the AST grammar and are composed together in a manner that mirrors the structure of the tree being generated. A vertical LSTM state is passed from module to module to propagate information during the decoding process.

The encoder uses bidirectional LSTMs to embed each component and a feedforward network to combine them. Component- and token-level attention is applied over the input at each step of the decoding process.

We train our model using negative log likelihood as the loss function. The likelihood encompasses terms for all generation decisions made by the decoder.

3.1 Encoder

Each component c of the input is encoded using a component-specific bidirectional LSTM. This results in forward and backward token encodings $(\overrightarrow{\mathbf{h}^c}, \overleftarrow{\mathbf{h}^c})$ that are later used by the attention mechanism. To obtain an encoding of the input as a whole for decoder initialization, we concatenate the final forward and backward encodings of each component into a single vector and apply a linear projection.

3.2 Decoder Modules

The decoder decomposes into several classes of modules, one per construct in the grammar, which we discuss in turn. Throughout, we let $\mathbf{v}$ denote the current vertical LSTM state, and use f to represent a generic feedforward neural network. LSTM updates with hidden state $\mathbf{h}$ and input $\mathbf{x}$ are notated as $\mathrm{LSTM}(\mathbf{h}, \mathbf{x})$.

Composite type modules Each composite type `T` has a corresponding module whose role is to select among the constructors `C` for that type. As Figure 5a exhibits, a composite type module receives a vertical LSTM state $\mathbf{v}$ as input and applies a feedforward network $f_{\mathtt{T}}$ and a softmax output layer to choose a constructor:

$$p\left(\mathtt{C} \mid \mathtt{T}, \mathbf{v}\right) = \left[\mathrm{softmax}\left(f_{\mathtt{T}}\left(\mathbf{v}\right)\right)\right]_{\mathtt{C}}.$$

Control is then passed to the module associated with constructor `C`.

Constructor modules Each constructor `C` has a corresponding module whose role is to compute an intermediate vertical LSTM state $\mathbf{v}_{u,\mathtt{F}}$ for each of its fields `F` whenever `C` is chosen at a composite node u.

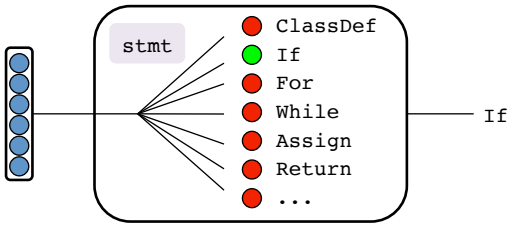
For each field `F` of the constructor, an embedding $\mathbf{e}_{\mathtt{F}}$ is concatenated with an attention-based context vector $\mathbf{c}$ and fed through a feedforward neural network $f_{\mathtt{C}}$ to obtain a context-dependent field embedding:

$$\tilde{\mathbf{e}}_{\mathtt{F}} = f_{\mathtt{C}}\left(\mathbf{e}_{\mathtt{F}},\ \mathbf{c}\right).$$

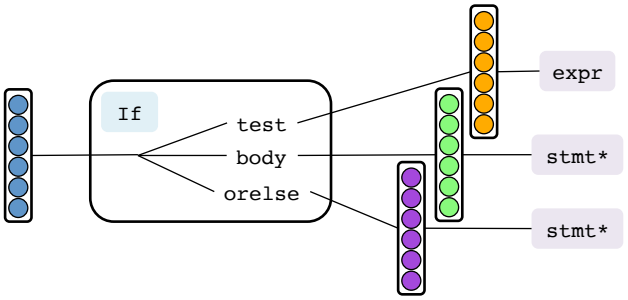
An intermediate vertical state for the field `F` at composite node u is then computed as

$$\mathbf{v}_{u,\mathtt{F}} = \mathrm{LSTM}^{v}\left(\mathbf{v}_{u},\ \tilde{\mathbf{e}}_{\mathtt{F}}\right).$$

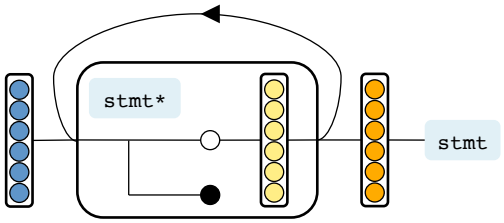
Figure 5b illustrates the process, starting with a single vertical LSTM state and ending with one updated state per field.



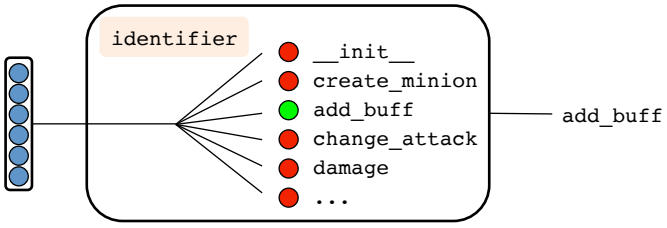
(a) A composite type module choosing a constructor for the corresponding type.



(b) A constructor module computing updated vertical LSTM states.



(c) A constructor field module (`sequential` cardinality) generating children to populate the field. At each step, the module decides whether to generate a child and continue (white circle) or stop (black circle).



(d) A primitive type module choosing a value from a closed list.

Figure 5: The module classes constituting our decoder. For brevity, we omit the cardinality modules for `singular` and `optional` cardinalities.

Constructor field modules Each field `F` of a constructor has a corresponding module whose role is to determine the number of children associated with that field and to propagate an updated vertical LSTM state to them. In the case of fields with `singular` cardinality, the decision and update are both vacuous, as exactly one child is always generated. Hence these modules forward the field vertical LSTM state $\mathbf{v}_{u,\mathtt{F}}$ unchanged to the child w corresponding to `F`:

$$\mathbf{v}_w = \mathbf{v}_{u,\mathtt{F}}. \tag{1}$$

Fields with `optional` cardinality can have either zero or one children; this choice is made using a feedforward network applied to the vertical LSTM state:

$$p(z_\mathtt{F} = 1 \mid \mathbf{v}_{u,\mathtt{F}}) = \text{sigmoid}\left(f_\mathtt{F}^{\text{gen}}(\mathbf{v}_{u,\mathtt{F}})\right). \tag{2}$$

If a child is to be generated, then as in (1), the state is propagated forward without modification.

In the case of `sequential` fields, a horizontal LSTM is employed for both child decisions and state updates. We refer to Figure 5c for an illustration of the recurrent process. After being initialized with a transformation of the vertical state, $\mathbf{s}_{\mathtt{F},0} = \mathbf{W}_\mathtt{F}\mathbf{v}_{u,\mathtt{F}}$, the horizontal LSTM iteratively decides whether to generate another child by applying a modified form of (2):

$$p\left(z_{\mathtt{F},i} = 1 \mid \mathbf{s}_{\mathtt{F},i-1},\ \mathbf{v}_{u,\mathtt{F}}\right) = \text{sigmoid}\left(f_\mathtt{F}^{\text{gen}}\left(\mathbf{s}_{\mathtt{F},i-1},\ \mathbf{v}_{u,\mathtt{F}}\right)\right).$$

If $z_{\mathtt{F},i} = 0$, generation stops and the process terminates, as represented by the solid black circle in Figure 5c. Otherwise, the process continues as represented by the white circle in Figure 5c. In that case, the horizontal state $\mathbf{s}_{u,i-1}$ is combined with the vertical state $\mathbf{v}_{u,\mathtt{F}}$ and an attention-based context vector $\mathbf{c}_{\mathtt{F},i}$ using a feedforward network $f_\mathtt{F}^{\text{update}}$ to obtain a joint context-dependent encoding of the field `F` and the position i:

$$\tilde{\mathbf{e}}_{\mathtt{F},i} = f_\mathtt{F}^{\text{update}}(\mathbf{v}_{u,\mathtt{F}},\ \mathbf{s}_{u,i-1},\ \mathbf{c}_{\mathtt{F},i}).$$

The result is used to perform a vertical LSTM update for the corresponding child w_i:

$$\mathbf{v}_{w_i} = \text{LSTM}^v(\mathbf{v}_{u,\mathtt{F}},\ \tilde{\mathbf{e}}_{\mathtt{F},i}).$$

Finally, the horizontal LSTM state is updated using the same field-position encoding, and the process continues:

$$\mathbf{s}_{u,i} = \text{LSTM}^h(\mathbf{s}_{u,i-1},\ \tilde{\mathbf{e}}_{\mathtt{F},i}).$$

Primitive type modules Each primitive type $\mathtt{T}$ has a corresponding module whose role is to select among the values $\mathtt{y}$ within the domain of that type. Figure 5d presents an example of the simplest form of this selection process, where the value $\mathtt{y}$ is obtained from a closed list via a softmax layer applied to an incoming vertical LSTM state:

$$p\left(\mathtt{y} \mid \mathtt{T}, \mathbf{v}\right) = \left[\mathrm{softmax}\left(f_{\mathtt{T}}\left(\mathbf{v}\right)\right)\right]_{\mathtt{y}}.$$

Some string-valued types are open class, however. To deal with these, we allow generation both from a closed list of previously seen values, as in Figure 5d, and synthesis of new values. Synthesis is delegated to a character-level LSTM language model (Bengio et al., 2003), and part of the role of the primitive module for open class types is to choose whether to synthesize a new value or not. During training, we allow the model to use the character LSTM only for unknown strings but include the log probability of that binary decision in the loss in order to ensure the model learns when to generate from the character LSTM.

3.3 Decoding Process

The decoding process proceeds through mutual recursion between the constituting modules, where the syntactic structure of the output tree mirrors the call graph of the generation procedure. At each step, the active decoder module either makes a generation decision, propagates state down the tree, or both.

To construct a composite node of a given type, the decoder calls the appropriate composite type module to obtain a constructor and its associated module. That module is then invoked to obtain updated vertical LSTM states for each of the constructor's fields, and the corresponding constructor field modules are invoked to advance the process to those children.

This process continues downward, stopping at each primitive node, where a value is generated but no further recursion is carried out.

3.4 Attention

Following standard practice for sequence-to-sequence models, we compute a raw bilinear attention score q_t^{raw} for each token t in the input using the decoder's current state $\mathbf{x}$ and the token's encoding $\mathbf{e}_t$:

$$q_t^{\mathrm{raw}} = \mathbf{e}_t^\top \mathbf{W} \mathbf{x}.$$

The current state $\mathbf{x}$ can be either the vertical LSTM state in isolation or a concatentation of the vertical LSTM state and either a horizontal LSTM state or a character LSTM state (for string generation). Each submodule that computes attention does so using a separate matrix $\mathbf{W}$.

A separate attention score q_c^{comp} is computed for each component of the input, independent of its content:

$$q_c^{\mathrm{comp}} = \mathbf{w}_c^\top \mathbf{x}.$$

The final token-level attention scores are the sums of the raw token-level scores and the corresponding component-level scores:

$$q_t = q_t^{\mathrm{raw}} + q_{c(t)}^{\mathrm{comp}},$$

where $c(t)$ denotes the component in which token t occurs. The attention weight vector $\mathbf{a}$ is then computed using a softmax:

$$\mathbf{a} = \mathrm{softmax}\left(\mathbf{q}\right).$$

Given the weights, the attention-based context is given by:

$$\mathbf{c} = \sum_t a_t \mathbf{e}_t.$$

Certain decision points that require attention have been highlighted in the description above; however, in our final implementation we made attention available to the decoder at all decision points.

Supervised Attention In the datasets we consider, partial or total copying of input tokens into primitive nodes is quite common. Rather than providing an explicit copying mechanism (Ling et al., 2016), we instead generate alignments where possible to define a set of tokens on which the attention at a given primitive node should be concentrated.[2] If no matches are found, the corresponding set of tokens is taken to be the whole input.

The attention supervision enters the loss through a term that encourages the final attention weights to be concentrated on the specified subset. Formally, if the matched subset of component-token pairs is S, the loss term associated with the supervision would be

$$\log \sum_t \exp\left(a_t\right) - \log \sum_{t \in S} \exp\left(a_t\right), \qquad (3)$$

[2] Alignments are generated using an exact string match heuristic that also included some limited normalization, primarily splitting of special characters, undoing camel case, and lemmatization for the semantic parsing datasets.

where a_t is the attention weight associated with token t, and the sum in the first term ranges over all tokens in the input. The loss in (3) can be interpreted as the negative log probability of attending to some token in S.

4 Experimental evaluation

4.1 Semantic parsing

Data We use three semantic parsing datasets: JOBS, GEO, and ATIS. All three consist of natural language queries paired with a logical representation of their denotations. JOBS consists of 640 such pairs, with Prolog-style logical representations, while GEO and ATIS consist of 880 and 5,410 such pairs, respectively, with λ-calculus logical forms. We use the same training-test split as Zettlemoyer and Collins (2005) for JOBS and GEO, and the standard training-development-test split for ATIS. We use the preprocessed versions of these datasets made available by Dong and Lapata (2016), where text in the input has been lowercased and stemmed using NLTK (Bird et al., 2009), and matching entities appearing in the same input-output pair have been replaced by numbered abstract identifiers of the same type.

Evaluation We compute accuracies using tree exact match for evaluation. Following the publicly released code of Dong and Lapata (2016), we canonicalize the order of the children within conjunction and disjunction nodes to avoid spurious errors, but otherwise perform no transformations before comparison.

4.2 Code generation

Data We use the HEARTHSTONE dataset introduced by Ling et al. (2016), which consists of 665 cards paired with their implementations in the open-source Hearthbreaker engine.[3] Our training-development-test split is identical to that of Ling et al. (2016), with split sizes of 533, 66, and 66, respectively.

Cards contain two kinds of components: textual components that contain the card's name and a description of its function, and categorical ones that contain numerical attributes (attack, health, cost, and durability) or enumerated attributes (rarity, type, race, and class). The name of the card is represented as a sequence of characters, while its description consists of a sequence of tokens split on whitespace and punctuation. All categorical components are represented as single-token sequences.

Evaluation For direct comparison to the results of Ling et al. (2016), we evaluate our predicted code based on exact match and token-level BLEU relative to the reference implementations from the library. We additionally compute node-based precision, recall, and F1 scores for our predicted trees compared to the reference code ASTs. Formally, these scores are obtained by defining the intersection of the predicted and gold trees as their largest common tree prefix.

4.3 Settings

For each experiment, all feedforward and LSTM hidden dimensions are set to the same value. We select the dimension from {30, 40, 50, 60, 70} for the smaller JOBS and GEO datasets, or from {50, 75, 100, 125, 150} for the larger ATIS and HEARTHSTONE datasets. The dimensionality used for the inputs to the encoder is set to 100 in all cases. We apply dropout to the non-recurrent connections of the vertical and horizontal LSTMs, selecting the noise ratio from {0.2, 0.3, 0.4, 0.5}. All parameters are randomly initialized using Glorot initialization (Glorot and Bengio, 2010).

We perform 200 passes over the data for the JOBS and GEO experiments, or 400 passes for the ATIS and HEARTHSTONE experiments. Early stopping based on exact match is used for the semantic parsing experiments, where performance is evaluated on the training set for JOBS and GEO or on the development set for ATIS. Parameters for the HEARTHSTONE experiments are selected based on development BLEU scores. In order to promote generalization, ties are broken in all cases with a preference toward higher dropout ratios and lower dimensionalities, in that order.

Our system is implemented in Python using the DyNet neural network library (Neubig et al., 2017). We use the Adam optimizer (Kingma and Ba, 2014) with its default settings for optimization, with a batch size of 20 for the semantic parsing experiments, or a batch size of 10 for the HEARTHSTONE experiments.

4.4 Results

Our results on the semantic parsing datasets are presented in Table 1. Our basic system achieves

[3] Available online at https://github.com/danielyule/hearthbreaker.

ATIS		GEO		JOBS	
System	Accuracy	System	Accuracy	System	Accuracy
ZH15	84.2	ZH15	88.9	ZH15	85.0
ZC07	84.6	KCAZ13	89.0	PEK03	88.0
WKZ14	**91.3**	WKZ14	**90.4**	LJK13	90.7
DL16	84.6	DL16	87.1	DL16	90.0
ASN	85.3	ASN	85.7	ASN	**91.4**
+ SUPATT	85.9	+ SUPATT	87.1	+ SUPATT	**92.9**

Table 1: Accuracies for the semantic parsing tasks. ASN denotes our abstract syntax network framework. SUPATT refers to the supervised attention mentioned in Section 3.4.

System	Accuracy	BLEU	F1
NEAREST	3.0	65.0	65.7
LPN	6.1	67.1	–
ASN	**18.2**	**77.6**	**72.4**
+ SUPATT	**22.7**	**79.2**	**75.6**

Table 2: Results for the HEARTHSTONE task. SUPATT refers to the system with supervised attention mentioned in Section 3.4. LPN refers to the system of Ling et al. (2016). Our nearest neighbor baseline NEAREST follows that of Ling et al. (2016), though it performs somewhat better; its nonzero exact match number stems from spurious repetition in the data.

a new state-of-the-art accuracy of 91.4% on the JOBS dataset, and this number improves to 92.9% when supervised attention is added. On the ATIS and GEO datasets, we respectively exceed and match the results of Dong and Lapata (2016). However, these fall short of the previous best results of 91.3% and 90.4%, respectively, obtained by Wang et al. (2014). This difference may be partially attributable to the use of typing information or rich lexicons in most previous semantic parsing approaches (Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2013; Wang et al., 2014; Zhao and Huang, 2015).

On the HEARTHSTONE dataset, we improve significantly over the initial results of Ling et al. (2016) across all evaluation metrics, as shown in Table 2. On the more stringent exact match metric, we improve from 6.1% to 18.2%, and on token-level BLEU, we improve from 67.1 to 77.6. When supervised attention is added, we obtain an additional increase of several points on each scale, achieving peak results of 22.7% accuracy and 79.2 BLEU.


```
class IronbarkProtector(MinionCard):
  def __init__(self):
    super().__init__(
      'Ironbark Protector', 8,
      CHARACTER_CLASS.DRUID,
      CARD_RARITY.COMMON)
  def create_minion(self, player):
    return Minion(
      8, 8, taunt=True)
```

Figure 6: Cards with minimal descriptions exhibit a uniform structure that our system almost always predicts correctly, as in this instance.


```
class ManaWyrm(MinionCard):
  def __init__(self):
    super().__init__(
      'Mana Wyrm', 1,
      CHARACTER_CLASS.MAGE,
      CARD_RARITY.COMMON)
  def create_minion(self, player):
    return Minion(
      1, 3, effects=[
        Effect(
          SpellCast(),
          ActionTag(
            Give(ChangeAttack(1)),
                 SelfSelector()))
      ])
```

Figure 7: For many cards with moderately complex descriptions, the implementation follows a functional style that seems to suit our modeling strategy, usually leading to correct predictions.

4.5 Error Analysis and Discussion

As the examples in Figures 6-8 show, classes in the HEARTHSTONE dataset share a great deal of common structure. As a result, in the simplest cases, such as in Figure 6, generating the code is simply a matter of matching the overall structure and plugging in the correct values in the initializer and a few other places. In such cases, our system generally predicts the correct code, with the


```
class MultiShot(SpellCard):
  def __init__(self):
    super().__init__(
      'Multi-Shot', 4,
      CHARACTER_CLASS.HUNTER,
      CARD_RARITY.FREE)
  def use(self, player, game):
    super().use(player, game)
    targets = copy.copy(
                  game.other_player.minions)
    for i in range(0, 2):
      target = game.random_choice(targets)
      targets.remove(target)
      target.damage(
        player.effective_spell_damage(3),
        self)
  def can_use(self, player, game):
    return (
      super().can_use(player, game) and
      (len(game.other_player.minions) >= 2))
```

```
class MultiShot(SpellCard):
  def __init__(self):
    super().__init__(
      'Multi-Shot', 4,
      CHARACTER_CLASS.HUNTER,
      CARD_RARITY.FREE)
  def use(self, player, game):
    super().use(player, game)
    minions = copy.copy(
                  game.other_player.minions)
    for i in range(0, 3):
      minion = game.random_choice(minions)
      minions.remove(minion)
  def can_use(self, player, game):
    return (
      super().can_use(player, game) and
      len(game.other_player.minions) >= 3)
```

Figure 8: Cards with nontrivial logic expressed in an imperative style are the most challenging for our system. In this example, our prediction comes close to the gold code, but misses an important statement in addition to making a few other minor errors. *(Left)* gold code; *(right)* predicted code.

exception of instances in which strings are incorrectly transduced. Introducing a dedicated copying mechanism like the one used by Ling et al. (2016) or more specialized machinery for string transduction may alleviate this latter problem.

The next simplest category of card-code pairs consists of those in which the card's logic is mostly implemented via nested function calls. Figure 7 illustrates a typical case, in which the card's effect is triggered by a game event (a spell being cast) and both the trigger and the effect are described by arguments to an `Effect` constructor. Our system usually also performs well on instances like these, apart from idiosyncratic errors that can take the form of under- or overgeneration or simply substitution of incorrect predicates.

Cards whose code includes complex logic expressed in an imperative style, as in Figure 8, pose the greatest challenge for our system. Factors like variable naming, nontrivial control flow, and interleaving of code predictable from the description with code required due to the conventions of the library combine to make the code for these cards difficult to generate. In some instances (as in the figure), our system is nonetheless able to synthesize a close approximation. However, in the most complex cases, the predictions deviate significantly from the correct implementation.

In addition to the specific errors our system makes, some larger issues remain unresolved. Existing evaluation metrics only approximate the actual metric of interest: functional equivalence. Modifications of BLEU, tree F1, and exact match that canonicalize the code—for example, by anonymizing all variables—may prove more meaningful. Direct evaluation of functional equivalence is of course impossible in general (Sipser, 2006), and practically challenging even for the HEARTHSTONE dataset because it requires integrating with the game engine.

Existing work also does not attempt to enforce *semantic* coherence in the output. Long-distance semantic dependencies, between occurrences of a single variable for example, in particular are not modeled. Nor is well-typedness or executability. Overcoming these evaluation and modeling issues remains an important open problem.

5 Conclusion

ASNs provide a modular encoder-decoder architecture that can readily accommodate a variety of tasks with structured output spaces. They are particularly applicable in the presence of recursive decompositions, where they can provide a simple decoding process that closely parallels the inherent structure of the outputs. Our results demonstrate their promise for tree prediction tasks, and we believe their application to more general output structures is an interesting avenue for future work.

Acknowledgments

MR is supported by an NSF Graduate Research Fellowship and a Fannie and John Hertz Foundation Google Fellowship. MS is supported by an NSF Graduate Research Fellowship.

References

Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. 2006. *Compilers: Principles, Techniques, and Tools (2Nd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Miltiadis Allamanis, Daniel Tarlow, Andrew D. Gordon, and Yi Wei. 2015. Bimodal modelling of source code and natural language. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. pages 2123–2132.

David Alvarez-Melis and Tommi S. Jaakkola. 2017. Tree-structured decoding with doubly-recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR) 2017*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Oral.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2016. Deepcoder: Learning to write programs. *CoRR* abs/1611.01989.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3:1137–1155. http://dl.acm.org/citation.cfm?id=944919.944966.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 740–750. http://aclweb.org/anthology/D/D14/D14-1082.pdf.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 1–11.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *CoRR* abs/1601.01280.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 199–209.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1545–1556.

Percy Liang, Michael I. Jordan, and Dan Klein. 2010. Learning programs: A hierarchical bayesian approach. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. pages 639–646.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Comput. Linguist.* 39(2):389–446. https://doi.org/10.1162/COLI_a_00127.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Chris J. Maddison and Daniel Tarlow. 2014. Structured generative models of natural source code. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*. pages 649–657.

Aditya Krishna Menon, Omer Tamuz, Sumit Gulwani, Butler W. Lampson, and Adam Kalai. 2013. A machine learning framework for programming by example. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. pages 187–195.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke

Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, pages 149–157.

Richard Shin, Alexander A. Alemi, Geoffrey Irving, and Oriol Vinyals. 2017. Tree-structured variational autoencoder. In *Proceedings of the International Conference on Learning Representations (ICLR) 2017*.

Michael Sipser. 2006. *Introduction to the Theory of Computation*. Course Technology, second edition.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 2773–2781.

Adrienne Wang, Tom Kwiatkowski, and Luke S Zettlemoyer. 2014. Morpho-syntactic lexical generalization for ccg semantic parsing. In *EMNLP*. pages 1284–1295.

Daniel C. Wang, Andrew W. Appel, Jeff L. Korn, and Christopher S. Serra. 1997. The zephyr abstract syntax description language. In *Proceedings of the Conference on Domain-Specific Languages on Conference on Domain-Specific Languages (DSL), 1997*. USENIX Association, Berkeley, CA, USA, DSL'97, pages 17–17.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*. pages 658–666.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007*. pages 678–687.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*. pages 1416–1421.

A Appendix

```
expr
  = Apply(pred predicate, arg* arguments)
  | Not(expr argument)
  | Or(expr left, expr right)
  | And(expr* arguments)

arg
  = Literal(lit literal)
  | Variable(var variable)
```

Figure 9: The Prolog-style grammar we use for the JOBS task.

```
expr
  = Variable(var variable)
  | Entity(ent entity)
  | Number(num number)
  | Apply(pred predicate, expr* arguments)
  | Argmax(var variable, expr domain, expr body)
  | Argmin(var variable, expr domain, expr body)
  | Count(var variable, expr body)
  | Exists(var variable, expr body)
  | Lambda(var variable, var_type type, expr body)
  | Max(var variable, expr body)
  | Min(var variable, expr body)
  | Sum(var variable, expr domain, expr body)
  | The(var variable, expr body)
  | Not(expr argument)
  | And(expr* arguments)
  | Or(expr* arguments)
  | Compare(cmp_op op, expr left, expr right)

cmp_op = Equal | LessThan | GreaterThan
```

Figure 10: The λ-calculus grammar used by our system.

Visualizing and Understanding Neural Machine Translation

Yanzhuo Ding[†] **Yang Liu**[†‡*] **Huanbo Luan**[†] **Maosong Sun**[†‡]
[†]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[‡]Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China
djx133@yeah.net, liuyang2011@tsinghua.edu.cn
luanhuanbo@gmail.com, sms@tsinghua.edu.cn

Abstract

While neural machine translation (NMT) has made remarkable progress in recent years, it is hard to interpret its internal workings due to the continuous representations and non-linearity of neural networks. In this work, we propose to use layer-wise relevance propagation (LRP) to compute the contribution of each contextual word to arbitrary hidden states in the attention-based encoder-decoder framework. We show that visualization with LRP helps to interpret the internal workings of NMT and analyze translation errors.

1 Introduction

End-to-end neural machine translation (NMT), which leverages neural networks to directly map between natural languages, has gained increasing popularity recently (Sutskever et al., 2014; Bahdanau et al., 2015). NMT proves to outperform conventional statistical machine translation (SMT) significantly across a variety of language pairs (Junczys-Dowmunt et al., 2016) and becomes the new *de facto* method in practical MT systems (Wu et al., 2016).

However, there still remains a severe challenge: it is hard to interpret the internal workings of NMT. In SMT (Koehn et al., 2003; Chiang, 2005), the translation process can be denoted as a derivation that comprises a sequence of translation rules (e.g., phrase pairs and synchronous CFG rules). Defined on language structures with varying granularities, these translation rules are interpretable from a linguistic perspective. In contrast, NMT takes an end-to-end approach: all internal information is represented as real-valued vectors or matrices. It is challenging to associate hidden states in neural networks with interpretable language structures. As a result, the lack of interpretability makes it very difficult to understand translation process and debug NMT systems.

Therefore, it is important to develop new methods for visualizing and understanding NMT. Existing work on visualizing and interpreting neural models has been extensively investigated in computer vision (Krizhevsky et al., 2012; Mahendran and Vedaldi, 2015; Szegedy et al., 2014; Simonyan et al., 2014; Nguyen et al., 2015; Girshick et al., 2014; Bach et al., 2015). Although visualizing and interpreting neural models for natural language processing has started to attract attention recently (Karpathy et al., 2016; Li et al., 2016), to the best of our knowledge, there is no existing work on visualizing NMT models. Note that the attention mechanism (Bahdanau et al., 2015) is restricted to demonstrate the connection between words in source and target languages and unable to offer more insights in interpreting how target words are generated (see Section 4.5).

In this work, we propose to use layer-wise relevance propagation (LRP) (Bach et al., 2015) to visualize and interpret neural machine translation. Originally designed to compute the contributions of single pixels to predictions for image classifiers, LRP back-propagates relevance recursively from the output layer to the input layer. In contrast to visualization methods relying on derivatives, a major advantage of LRP is that it does not require neural activations to be differentiable or smooth (Bach et al., 2015). We adapt LRP to the attention-based encoder-decoder framework (Bahdanau et al., 2015) to calculate relevance that measures the association degree between two arbitrary neurons in neural networks. Case studies on Chinese-English translation show that visualization helps to interpret the internal workings of

*Corresponding author.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1150–1159

https://doi.org/10.18653/v1/P17-1106

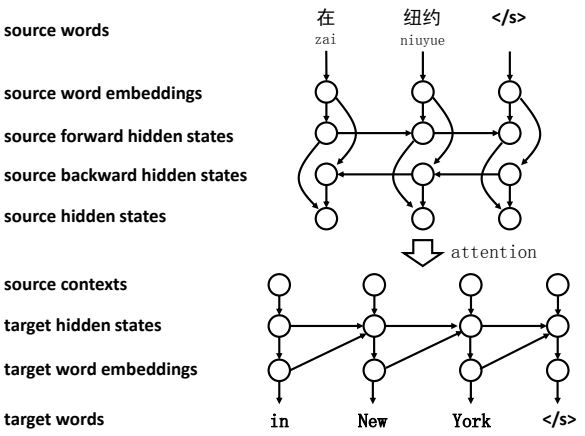


Figure 1: The attention-based encoder-decoder architecture for neural machine translation (Bahdanau et al., 2015).

NMT and analyze translation errors.

2 Background

Given a source sentence $\mathbf{x} = x_1, \ldots, x_i, \ldots, x_I$ with I source words and a target sentence $\mathbf{y} = y_1, \ldots, y_j, \ldots, y_J$ with J target words, neural machine translation (NMT) decomposes the sentence-level translation probability as a product of word-level translation probabilities:

$$P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \prod_{j=1}^{J} P(y_j|\mathbf{x}, \mathbf{y}_{<j};\boldsymbol{\theta}), \quad (1)$$

where $\mathbf{y}_{<j} = y_1, \ldots, y_{j-1}$ is a partial translation.

In this work, we focus on the attention-based encoder-decoder framework (Bahdanau et al., 2015). As shown in Figure 1, given a source sentence $\mathbf{x}$, the encoder first uses source word embeddings to map each source word x_i to a real-valued vector $\boldsymbol{x}_i$.[1]

Then, a forward recurrent neural network (RNN) with GRU units (Cho et al., 2014) runs to calculate source forward hidden states:

$$\overrightarrow{\mathbf{h}}_i = f(\overrightarrow{\mathbf{h}}_{i-1}, \boldsymbol{x}_i), \quad (2)$$

where $f(\cdot)$ is a non-linear function.

Similarly, the source backward hidden states can be obtained using a backward RNN:

$$\overleftarrow{\mathbf{h}}_i = f(\overleftarrow{\mathbf{h}}_{i+1}, \boldsymbol{x}_i). \quad (3)$$

[1]Note that we use $\mathbf{x}$ to denote a source sentence and $\boldsymbol{x}$ to denote the vector representation of a single source word.

To capture global contexts, the forward and backward hidden states are concatenated as the hidden state for each source word:

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]. \quad (4)$$

Bahdanau et al. (2015) propose an attention mechanism to dynamically determine the relevant source context $\mathbf{c}_j$ for each target word:

$$\mathbf{c}_j = \sum_{i=1}^{I+1} \alpha_{j,i}\mathbf{h}_i, \quad (5)$$

where $\alpha_{j,i}$ is an attention weight that indicates how well the source word x_i and the target word y_j match. Note that an end-of-sentence token is appended to the source sentence.

In the decoder, a target hidden state for the j-th target word is calculated as

$$\mathbf{s}_j = g(\mathbf{s}_{j-1}, \boldsymbol{y}_j, \mathbf{c}_j), \quad (6)$$

where $g(\cdot)$ is a non-linear function, $\boldsymbol{y}_{j-1}$ denotes the vector representation of the $(j-1)$-th target word.

Finally, the word-level translation probability is given by

$$P(y_j|\mathbf{x}, \mathbf{y}_{<j};\boldsymbol{\theta}) = \rho(\boldsymbol{y}_{j-1}, \mathbf{s}_j, \mathbf{c}_j), \quad (7)$$

where $\rho(\cdot)$ is a non-linear function.

Although NMT proves to deliver state-of-the-art translation performance with the capability to handle long-distance dependencies due to GRU and attention, it is hard to interpret the internal information such as $\overrightarrow{\mathbf{h}}_i$, $\overleftarrow{\mathbf{h}}_i$, $\mathbf{h}_i$, $\mathbf{c}_j$, and $\mathbf{s}_j$ in the encoder-decoder framework. Though projecting word embedding space into two dimensions (Faruqui and Dyer, 2014) and the attention matrix (Bahdanau et al., 2015) shed partial light on how NMT works, how to interpret the entire network still remains a challenge.

Therefore, it is important to develop new methods for understanding the translation process and analyzing translation errors for NMT.

3 Approach

3.1 Problem Statement

Recent efforts on interpreting and visualizing neural models has focused on calculating the contribution of a unit at the input layer to the final decision at the output layer (Simonyan et al., 2014; Mahendran and Vedaldi, 2015; Nguyen et al., 2015;

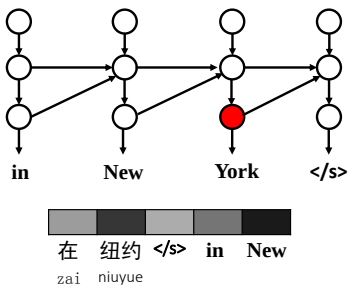


Figure 2: Visualizing the relevance between the vector representation of a target word "New York" and those of all source words and preceding target words.

Girshick et al., 2014; Bach et al., 2015; Li et al., 2016). For example, in image classification, it is important to understand the contribution of a single pixel to the prediction of classifier (Bach et al., 2015).

In this work, we are interested in calculating the contribution of source and target words to the following internal information in the attention-based encoder-decoder framework:

1. $\overrightarrow{\mathbf{h}}_i$: the i-th source forward hidden state,
2. $\overleftarrow{\mathbf{h}}_i$: the i-th source backward hidden state,
3. $\mathbf{h}_i$: the i-th source hidden state,
4. $\mathbf{c}_j$: the j-th source context vector,
5. $\mathbf{s}_j$: the j-th target hidden state,
6. $\boldsymbol{y}_j$: the j-th target word embedding.

For example, as shown in Figure 2, the generation of the third target word "York" depends on both the source context (i.e., the source sentence "zai niuyue </s>") and the target context (i.e., the partial translation "in New"). Intuitively, the source word "niuyue" and the target word "New" are more relevant to "York" and should receive higher relevance than other words. The problem is how to quantify and visualize the relevance between hidden states and contextual word vectors.

More formally, we introduce a number of definitions to facilitate the presentation.

Definition 1 *The* **contextual word set** *of a hidden state* $\mathbf{v} \in \mathbb{R}^{M\times 1}$ *is denoted as* $\mathcal{C}(\mathbf{v})$*, which is a set of source and target contextual word vectors* $\mathbf{u} \in \mathbb{R}^{N\times 1}$ *that influences the generation of* $\mathbf{v}$.

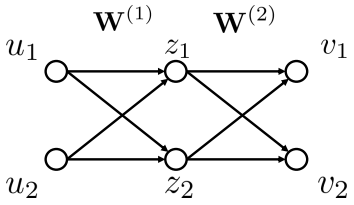


Figure 3: A simple feed-forward network for illustrating layer-wise relevance propagation (Bach et al., 2015).

For example, the context word set for $\overrightarrow{\mathbf{h}}_i$ is $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i\}$, for $\overleftarrow{\mathbf{h}}_i$ is $\{\boldsymbol{x}_i, \ldots, \boldsymbol{x}_{I+1}\}$, and for $\mathbf{h}_i$ is $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{I+1}\}$. The contextual word set for $\mathbf{c}_j$ is $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{I+1}\}$, for $\mathbf{s}_j$ and $\boldsymbol{y}_j$ is $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{I+1}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{j-1}\}$.

As both hidden states and contextual words are represented as real-valued vectors, we need to factorize vector-level relevance at the neuron level.

Definition 2 *The* **neuron-level relevance** *between the* m*-th neuron in a hidden state* $v_m \in \mathbb{R}$ *and the* n*-th neuron in a contextual word vector* $u_n \in \mathbb{R}$ *is denoted as* $r_{u_n \leftarrow v_m} \in \mathbb{R}$*, which satisfies the following constraint:*

$$v_m = \sum_{\mathbf{u}\in\mathcal{C}(\mathbf{v})} \sum_{n=1}^{N} r_{u_n \leftarrow v_m} \tag{8}$$

Definition 3 *The* **vector-level relevance** *between a hidden state* $\mathbf{v}$ *and one contextual word vector* $\mathbf{u} \in \mathcal{C}(\mathbf{v})$ *is denoted as* $R_{\mathbf{u}\leftarrow\mathbf{v}} \in \mathbb{R}$*, which quantifies the contribution of* $\mathbf{u}$ *to the generation of* $\mathbf{v}$*. It is calculated as*

$$R_{\mathbf{u}\leftarrow\mathbf{v}} = \sum_{m=1}^{M} \sum_{n=1}^{N} r_{u_n \leftarrow v_m} \tag{9}$$

Definition 4 *The* **relevance vector** *of a hidden state* $\mathbf{v}$ *is a sequence of vector-level relevance of its contextual words:*

$$R_{\mathbf{v}} = \{R_{\mathbf{u}_1\leftarrow\mathbf{v}}, \ldots, R_{\mathbf{u}_{|\mathcal{C}(\mathbf{v})|}\leftarrow\mathbf{v}}\} \tag{10}$$

Therefore, our goal is to compute relevance vectors for hidden states in a neural network, as shown in Figure 2. The key problem is how to compute neuron-level relevance.

3.2 Layer-wise Relevance Propagation

We follow (Bach et al., 2015) to use layer-wise relevance propagation (LRP) to compute neuron-level relevance. We use a simple feed-forward network shown in Figure 3 to illustrate the central idea of LRP.

Input: A neural network G for a sentence pair and a set of hidden states to be visualized $\mathcal{V}$.
Output: Vector-level relevance set $\mathcal{R}$.

for $u \in G$ *in a forward topological order* **do**
  for $v \in \mathrm{OUT}(u)$ **do**
    calculating weight ratios $w_{u \to v}$;
  end
end
for $\mathbf{v} \in \mathcal{V}$ **do**
  for $v \in \mathbf{v}$ **do**
    $r_{v \leftarrow v} = v$; `// initializing neuron-level relevance`
  end
  for $u \in G$ *in a backward topological order* **do**
    $r_{u \leftarrow v} = \sum_{z \in \mathrm{OUT}(u)} w_{u \to z} r_{z \leftarrow v}$; `// calculating neuron-level relevance`
  end
  for $\mathbf{u} \in \mathcal{C}(\mathbf{v})$ **do**
    $R_{\mathbf{u} \leftarrow \mathbf{v}} = \sum_{u \in \mathbf{u}} \sum_{v \in \mathbf{v}} r_{u \leftarrow v}$; `// calculating vector-level relevance`
    $\mathcal{R} = \mathcal{R} \cup \{R_{\mathbf{u} \leftarrow \mathbf{v}}\}$; `// Update vector-level relevance set`
  end
end

Algorithm 1: Layer-wise relevance propagation for neural machine translation.

LRP first propagates the relevance from the output layer to the intermediate layer:

$$r_{z_1 \leftarrow v_1} = \frac{\mathbf{W}^{(2)}_{1,1} z_1}{\mathbf{W}^{(2)}_{1,1} z_1 + \mathbf{W}^{(2)}_{2,1} z_2} v_1 \quad (11)$$

$$r_{z_2 \leftarrow v_1} = \frac{\mathbf{W}^{(2)}_{2,1} z_2}{\mathbf{W}^{(2)}_{1,1} z_1 + \mathbf{W}^{(2)}_{2,1} z_2} v_1 \quad (12)$$

Note that we ignore the non-linear activation function because Bach et al. (2015) indicate that LRP is *invariant* against the choice of non-linear function.

Then, the relevance is further propagated to the input layer:

$$\begin{aligned} r_{u_1 \leftarrow v_1} = & \frac{\mathbf{W}^{(1)}_{1,1} u_1}{\mathbf{W}^{(1)}_{1,1} u_1 + \mathbf{W}^{(1)}_{2,1} u_2} r_{z_1 \leftarrow v_1} + \\ & \frac{\mathbf{W}^{(1)}_{1,2} u_1}{\mathbf{W}^{(1)}_{1,2} u_1 + \mathbf{W}^{(1)}_{2,2} u_2} r_{z_2 \leftarrow v_1} \quad (13) \\ r_{u_2 \leftarrow v_1} = & \frac{\mathbf{W}^{(1)}_{2,1} u_2}{\mathbf{W}^{(1)}_{1,1} u_1 + \mathbf{W}^{(1)}_{2,1} u_2} r_{z_1 \leftarrow v_1} + \\ & \frac{\mathbf{W}^{(1)}_{2,2} u_2}{\mathbf{W}^{(1)}_{1,2} u_1 + \mathbf{W}^{(1)}_{2,2} u_2} r_{z_2 \leftarrow v_1} \quad (14) \end{aligned}$$

Note that $r_{u_1 \leftarrow v_1} + r_{u_2 \leftarrow v_1} = v_1$.

More formally, we introduce the following definitions to ease exposition.

Definition 5 *Given a neuron u, its* **incoming neuron set** $\mathrm{IN}(u)$ *comprises all its direct connected preceding neurons in the network.*

For example, in Figure 3, the incoming neuron set of z_1 is $\mathrm{IN}(z_1) = \{u_1, u_2\}$.

Definition 6 *Given a neuron u, its* **outcoming neuron set** $\mathrm{OUT}(u)$ *comprises all its direct connected descendant neurons in the network.*

For example, in Figure 3, the incoming neuron set of z_1 is $\mathrm{OUT}(z_1) = \{v_1, v_2\}$.

Definition 7 *Given a neuron v and its incoming neurons $u \in \mathrm{IN}(v)$, the* **weight ratio** *that measures the contribution of u to v is calculated as*

$$w_{u \to v} = \frac{\mathbf{W}_{u,v} u}{\sum_{u' \in \mathrm{IN}(v)} \mathbf{W}_{u',v} u'} \quad (15)$$

Although the NMT model usually involves multiple operators such as matrix multiplication, element-wise multiplication, and maximization, they only influence the way to calculate weight ratios in Eq. (15).

For matrix multiplication such as $\mathbf{v} = \mathbf{W}\mathbf{u}$, its basic form that is calculated at the neuron level is given by $v = \sum_{u \in \mathrm{IN}(v)} \mathbf{W}_{u,v} u$. We follow Bach et al. (2015) to calculate the weight ratio using Eq. (15).

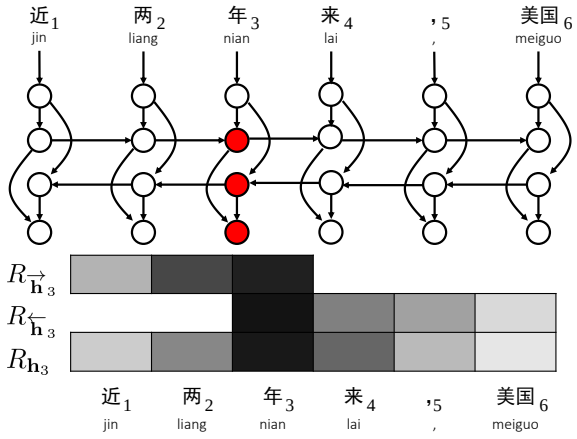


Figure 4: Visualizing source hidden states for a source content word "nian" (*years*).

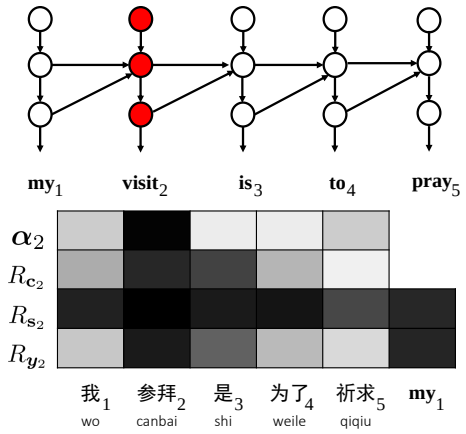


Figure 5: Visualizing target hidden states for a target content word "visit".

For element-wise multiplication such as $\mathbf{v} = \mathbf{u}_1 \circ \mathbf{u}_2$, its basic form is given by $v = \prod_{u \in \mathrm{IN}(v)} u$. We use the following method to calculate its weight ratio:

$$w_{u \to v} = \frac{u}{\sum_{u' \in \mathrm{IN}(v)} u'} \tag{16}$$

For maximization such as $v = \max\{u_1, u_2\}$, we calculate its weight ratio as follows:

$$w_{u \to v} = \begin{cases} 1 & \text{if } u = \max_{u' \in \mathrm{IN}(v)}\{u'\} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Therefore, the general local redistribution rule for LRP is given by

$$r_{u \leftarrow v} = \sum_{z \in \mathrm{OUT}(u)} w_{u \to z} r_{z \leftarrow v} \tag{18}$$

Algorithm 1 gives the layer-wise relevance propagation algorithm for neural machine translation. The input is an attention-based encoder-decoder neural network for a sentence pair after decoding G and a set of hidden states to be visualized $\mathcal{V}$. The output is a set of vector-level relevance between intended hidden states and their contextual words $\mathcal{R}$. The algorithm first computes weight ratios for each neuron in a forward pass (lines 1-4). Then, for each hidden state to be visualized (line 6), the algorithm initializes the neuron-level relevance for itself (lines 7-9). After initialization, the neuron-level relevance is back-propagated through the network (lines 10-12). Finally, vector-level relevance is calculated based on neuron-level relevance (lines 13-16). The time complexity of Algorithm 1 is $\mathcal{O}(|G| \times |\mathcal{V}| \times O_{\max})$, where $|G|$ is the number of neuron units in the neural network G, $|\mathcal{V}|$ is the number of hidden states to be visualized and $O_{\max}$ is the maximum of out-degree for neurons in the network. Calculating relevance is more computationally expensive than computing attention as it involves all neurons in the network. Fortunately, it is possible to take advantage of parallel architectures of GPUs and relevance caching for speed-up.

4 Analysis

4.1 Data Preparation

We evaluate our approach on Chinese-English translation. The training set consists of 1.25M pairs of sentences with 27.93M Chinese words and 34.51M English words. We use the NIST 2003 dataset as the development set for model selection and the NIST 2004 dataset as test set. The BLEU score on NIST 2003 is 32.73.

We use the open-source toolkit GROUNDHOG (Bahdanau et al., 2015), which implements the attention-based encoder-decoder framework. After model training and selection on the training and development sets, we use the resulting NMT model to translate the test set. Therefore, the visualization examples in the following subsections are taken from the test set.

4.2 Visualization of Hidden States

4.2.1 Source Side

Figure 4 visualizes the source hidden states for a source content word "nian" (*years*). For each word in the source string "jin liang nian lai , meiguo" (*in recent two years, USA*), we attach a number

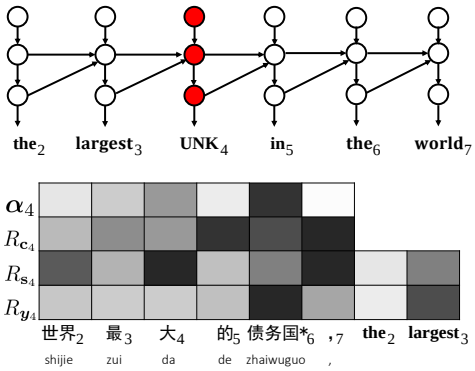


Figure 6: Visualizing target hidden states for a target UNK word.

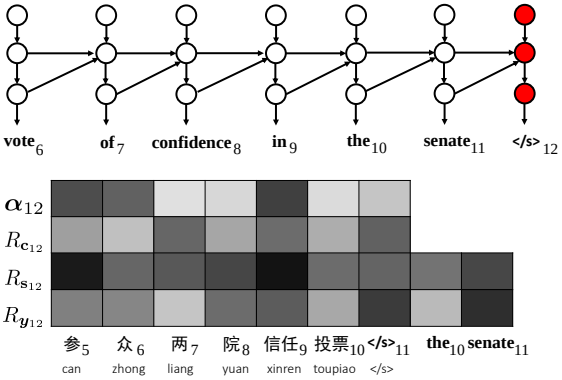


Figure 7: Analyzing translation error: word omission. The 6-th source word "zhong" is untranslated incorrectly.

to denote the position of the word in the sentence. For example, "nian" (*years*) is the third word.

We are interested in visualizing the relevance between the third source forward hidden state $\overrightarrow{\mathbf{h}}_3$ and all its contextual words "jin" (*recent*) and "liang" (*two*). We observe that the direct preceding word "liang" (*two*) contributes more to forming the forward hidden state of "nian" (*years*). For the third source backward hidden state $\overleftarrow{\mathbf{h}}_3$, the relevance of contextual words generally decreases with the increase of the distance to "nian" (*years*). Clearly, the concatenation of forward and backward hidden states $\mathbf{h}_3$ capture contexts in both directions.

The situations for function words and punctuation marks are similar but the relevance is usually more concentrated on the word itself. We omit the visualization due to space limit.

4.2.2 Target Side

Figure 5 visualizes the target-side hidden states for the second target word "visit". For comparison, we also give the attention weights $\boldsymbol{\alpha}_2$, which correctly identifies the second source word "canbai" ("visit") is most relevant to "visit".

The relevance vector of the source context $\mathbf{c}_2$ is generally consistent with the attention but reveals that the third word "shi" (*is*) also contributes to the generation of "visit".

For the target hidden state $\mathbf{s}_2$, the contextual word set includes the first target word "my". We find that most contextual words receive high values of relevance. This phenomenon has been frequently observed for most target words in other sentences. Note that relevance vector is not normalized. This is an essential difference between attention and relevance. While attention is defined to be normalized, the only constraint on relevance is that the sum of relevance of contextual words is identical to the value of intended hidden state neuron.

For the target word embedding $\mathbf{y}_2$, the relevance is generally consistent with the attention by identifying that the second source word contributes more to the generation of "visit". But $R_{\mathbf{y}_2}$ further indicates that the target word "my" is also very important for generating "visit".

Figure 6 shows the hidden states of a target UNK word, which is very common to see in NMT because of limited vocabulary. It is interesting to investigate whether the attention mechanism could put a UNK in the right place in the translation. In this example, the 6-th source word "zhaiwuguo" is a UNK. We find that the model successfully predicts the correct position of UNK by exploiting surrounding source and target contexts. But the ordering of UNK usually becomes worse if multiple UNK words exist on the source side.

4.3 Translation Error Analysis

Given the visualization of hidden states, it is possible to offer useful information for analyzing translation errors commonly observed in NMT such as word omission, word repetition, unrelated words and negation reversion.

4.3.1 Word Omission

Given a source sentence "bajisitan zongtong muxialafu yingde can zhong liang yuan xinren toupiao" (*pakistani president musharraf wins votes of confidence in senate and house*), the NMT model pro-

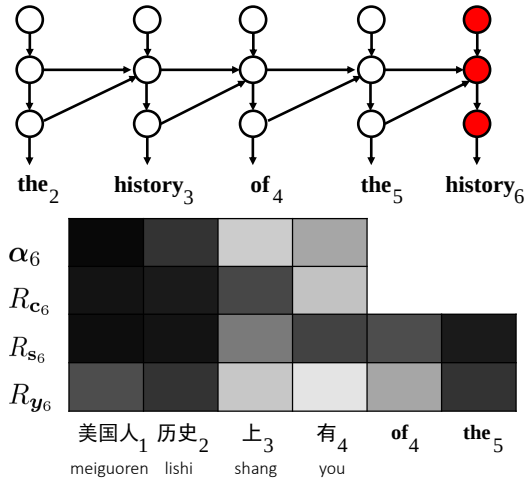


Figure 8: Analyzing translation error: word repetition. The target word "history" occurs twice in the translation incorrectly.

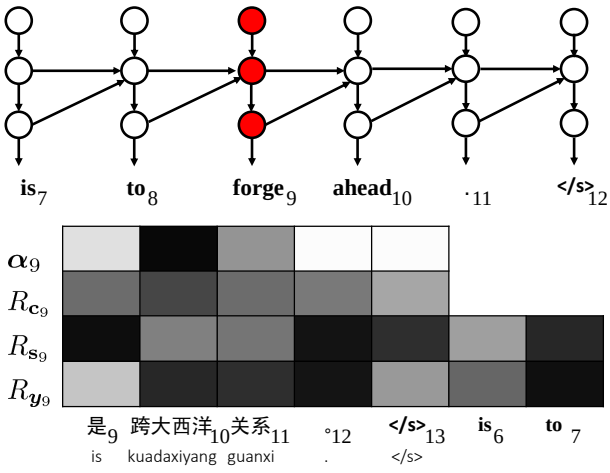


Figure 9: Analyzing translation error: unrelated words. The 9-th target word "forge" is totally unrelated to the source sentence.

duces a wrong translation "pakistani president win over democratic vote of confidence in the senate". One translation error is that the 6-th source word "zhong" (*house*) is incorrectly omitted for translation.

As the end-of-sentence token "</s>" occurs early than expected, we choose to visualize its corresponding target hidden states. Although the attention correctly identifies the 6-th source word "zhong" (*house*) to be important for generating the next target word, the relevance of source context $R_{\mathbf{c}_{12}}$ attaches more importance to the end-of-sentence token.

Finally, the relevance of target word $R_{\boldsymbol{y}_{12}}$ reveals that the end-of-sentence token and the 11-th target word "senate" become dominant in the softmax layer for generating the target word.

This example demonstrates that only using attention matrices does not suffice to analyze the internal workings of NMT. The values of relevance of contextual words might vary significantly across different layers.

4.3.2 Word Repetition

Given a source sentence "meiguoren lishi shang you jiang chengxi de chuantong , you fancuo rencuo de chuantong" (*in history , the people of america have the tradition of honesty and would not hesitate to admit their mistakes*), the NMT model produces a wrong translation "in the history of the history of the history of the americans , there is a tradition of faith in the history of mistakes". The translation error is that "history" repeats four times in the translation.

Figure 8 visualizes the target hidden states of the 6-th target word "history". According to the relevance of the target word embedding $R_{\boldsymbol{y}_6}$, the first source word "meiguoren" (*american*), the second source word "lishi" (*history*) and the 5-th target word "the" are most relevant to the generation of "history". Therefore, word repetition not only results from wrong attention but also is significantly influenced by target side context. This finding confirms the importance of controlling source and target contexts to improve fluency and adequacy (Tu et al., 2017).

4.3.3 Unrelated Words

Given a source sentence "ci ci huiyi de yi ge zhongyao yiti shi kuadaxiyang guanxi" (*one the the top agendas of the meeting is to discuss the cross-atlantic relations*), the model prediction is "a key topic of the meeting is to forge ahead". One translation error is that the 9-th English word "forge" is totally unrelated to the source sentence.

Figure 9 visualizes the hidden states of the 9-th target word "forge". We find that while the attention identifies the 10-th source word "kuadaxiyang" (*cross-atlantic*) to be most relevant, the relevance vector of the target word $R_{\boldsymbol{y}_9}$ finds that multiple source and target words should contribute to the generation of the next target word.

We observe that unrelated words are more likely to occur if multiple contextual words have high

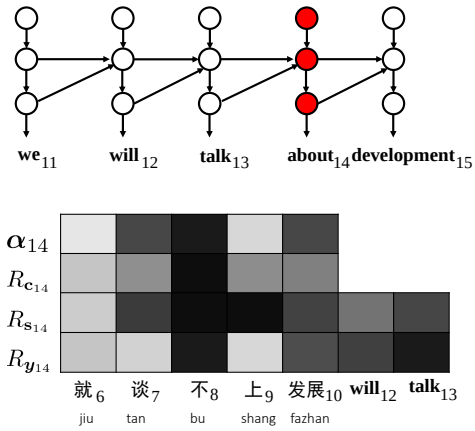


Figure 10: Analyzing translation error: negation. The 8-th negation source word "bu" (*not*) is not translated.

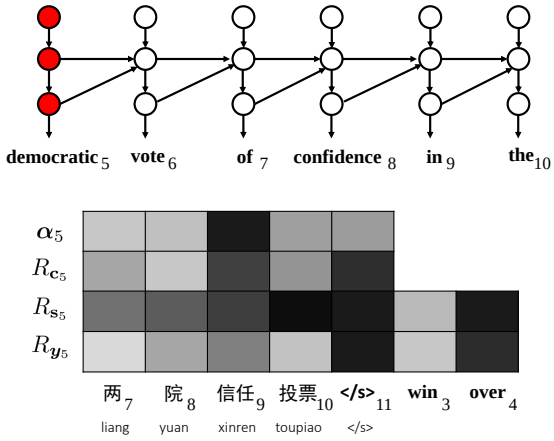


Figure 11: Analyzing translation error: extra word. The 5-th target word "democratic" is an extra word.

values in the relevance vector of the target word being generated.

4.3.4 Negation Reversion

Given a source sentence "bu jiejue shengcun wenti , jiu tan bu shang fa zhan , geng tan bu shang ke chixu fazhan" (*without solution to the issue of subsistence , there will be no development to speak of , let alone sustainable development*), the model prediction is "if we do not solve the problem of living , we will talk about development and still less can we talk about sustainable development". The translation error is that the 8-th negation source word "bu" (*not*) is untranslated. The omission of negation is a severe translation error it reverses the meaning of the source sentence.

As shown in Figure 10, while both attention and relevance correctly identify the 8-th negation word "bu" (*not*) to be most relevant, the model still generates "about" instead of a negation target word. One possible reason is that target context words "will talk" take the lead in determining the next target word.

4.4 Extra Words

Given a source sentence "bajisitan zongtong muxialafu yingde can zhong liang yuan xinren toupiao"(*pakistani president musharraf wins votes of confidence in senate and house*), the model prediction is "pakistani president win over democratic vote of confidence in the senate" The translation error is that the 5-th target word "democratic" is extra generated.

Figure 11 visualizes the hidden states of the 9-th target word "forge". We find that while the attention identifies the 9-th source word "xinren"(*confidence*) to be most relevant, the relevance vector of the target word R_{y_9} indicates that the end-of-sentence token and target words contribute more to the generation of "democratic".

4.5 Summary of Findings

We summarize the findings of visualizing and analyzing the decoding process of NMT as follows:

1. Although attention is very useful for understanding the connection between source and target words, only using attention is not sufficient for deep interpretation of target word generation (Figure 9);

2. The relevance of contextual words might vary significantly across different layers of hidden states (Figure 9);

3. Target-side context also plays a critical role in determining the next target word being generated. It is important to control both source and target contexts to produce correct translations (Figure 10);

4. Generating the end-of-sentence token too early might lead to many problems such as word omission, unrelated word generation, and truncated translation (Figures 7 and 9).

5 Related Work

Our work is closely related to previous visualization approaches that compute the contribution of a unit at the input layer to the final decision at the output layer (Simonyan et al., 2014; Mahendran and Vedaldi, 2015; Nguyen et al., 2015; Girshick et al., 2014; Bach et al., 2015; Li et al., 2016). Among them, our approach bears most resemblance to (Bach et al., 2015) since we adapt layer-wise relevance propagation to neural machine translation. The major difference is that word vectors rather than single pixels are the basic units in NMT. Therefore, we propose vector-level relevance based on neuron-level relevance for NMT. Calculating weight ratios has also been carefully designed for the operators in NMT.

The proposed approach also differs from (Li et al., 2016) in that we use relevance rather than partial derivative to quantify the contributions of contextual words. A major advantage of using relevance is that it does not require neural activations to be differentiable or smooth (Bach et al., 2015).

The relevance vector we used is significantly different from the attention matrix (Bahdanau et al., 2015). While attention only demonstrates the association degree between source and target words, relevance can be used to calculate the association degree between two arbitrary neurons in neural networks. In addition, relevance is effective in analyzing the effect of source and target contexts on generating target words.

6 Conclusion

In this work, we propose to use layer-wise relevance propagation to visualize and interpret neural machine translation. Our approach is capable of calculating the relevance between arbitrary hidden states and contextual words by back-propagating relevance along the network recursively. Analyses of the state-of-art attention-based encoder-decoder framework on Chinese-English translation show that our approach is able to offer more insights than the attention mechanism for interpreting neural machine translation.

In the future, we plan to apply our approach to more NMT approaches (Sutskever et al., 2014; Shen et al., 2016; Tu et al., 2016; Wu et al., 2016) on more language pairs to further verify its effectiveness. It is also interesting to develop relevance-based neural translation models to explicitly control relevance to produce better translations.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.61522204), the 863 Program (2015AA015407), and the National Natural Science Foundation of China (No.61432013). This research is also supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme.

References

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE* .

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Davie Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*.

Mannal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of CVPR*.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? a case study on 30 translation directions. arXiv:1610.01108v2.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2016. Visualing and understanding recurrent networks. In *Proceedings of ICLR Workshop*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. 2012. Imagenet classification with deep convolutional nerual networks. In *Proceedings of NIPS*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of NAACL*.

Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *Proceedings of CVPR*.

Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecignizable images. In *Proceedings of CVPR*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualizing image classification models and saliency maps. In *Proceedings of ICLR Workshop*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of ICLR*.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *Transactions of the ACL* .

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144v2.

Detecting annotation noise in automatically labelled data

Ines Rehbein **Josef Ruppenhofer**
IDS Mannheim/University of Heidelberg, Germany
Leibniz Science Campus "Empirical Linguistics and Computational Language Modeling"
rehbein@cl.uni-heidelberg.de, ruppenhofer@ids-mannheim.de

Abstract

We introduce a method for error detection in automatically annotated text, aimed at supporting the creation of high-quality language resources at affordable cost. Our method combines an unsupervised generative model with human supervision from active learning. We test our approach on in-domain and out-of-domain data in two languages, in AL simulations and in a real world setting. For all settings, the results show that our method is able to detect annotation errors with high precision and high recall.

1 Introduction

Until recently, most of the work in Computational Linguistics has been focussed on standard written text, often from newswire. The emergence of two new research areas, Digital Humanities and Computational Sociolinguistics, have however shifted the interest towards large, noisy text collections from various sources. More and more researchers are working with social media text, historical data, or spoken language transcripts, to name but a few. Thus the need for NLP tools that are able to process this data has become more and more apparent, and has triggered a lot of work on domain adaptation and on developing more robust preprocessing tools. Studies are usually carried out on large amounts of data, and thus fully manual annotation or even error correction of automatically prelabelled text is not feasible. Given the importance of identifying noisy annotations in automatically annotated data, it is all the more surprising that up to now this area of research has been severely understudied.

This paper addresses this gap and presents a method for error detection in *automatically* labelled text. As test cases, we use POS tagging and Named Entity Recognition, both standard preprocessing steps for many NLP applications. However, our approach is general and can also be applied to other classification tasks.

Our approach is based on the work of Hovy et al. (2013) who develop a generative model for estimating the reliability of multiple annotators in a crowdsourcing setting. We adapt the generative model to the task of finding errors in *automatically* labelled data by integrating it in an active learning (AL) framework. We first show that the approach of Hovy et al. (2013) on its own is not able to beat a strong baseline. We then present our integrated model, in which we impose human supervision on the generative model through AL, and show that we are able to achieve substantial improvements in two different tasks and for two languages.

Our contributions are the following. We provide a novel approach to error detection that is able to identify errors in *automatically* labelled text with high precision *and* high recall. To the best of our knowledge, our method is the first that addresses this task in an AL framework. We show how AL can be used to guide an unsupervised generative model, and we will make our code available to the research community.[1] Our approach works particularly well in out-of-domain settings where no annotated training data is yet available.

2 Related work

Quite a bit of work has been devoted to the identifcation of errors in *manually* annotated corpora (Eskin, 2000; van Halteren, 2000; Kveton and Oliva, 2002; Dickinson and Meurers, 2003; Loftsson, 2009; Ambati et al., 2011).

[1] Our code is available at http://www.cl.uni-heidelberg.de/~rehbein/resources.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1160–1170
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1107

Several studies have tried to identify trustworthy annotators in crowdsourcing settings (Snow et al., 2008; Bian et al., 2009), amongst them the work of Hovy et al. (2013) described in Section 3. Others have proposed selective relabelling strategies when working with non-expert annotators (Sheng et al., 2008; Zhao et al., 2011).

Manual annotations are often inconsistent and annotation errors can thus be identified by looking at the variance in the data. In contrast to this, we focus on detecting errors in *automatically* labelled data. This is a much harder problem as the annotation errors are systematic and consistent and therefore hard to detect. Only a few studies have addressed this problem. One of them is Rocio et al. (2007) who adapt a multiword unit extraction algorithm to detect automatic annotation errors in POS tagged corpora. Their semi-automatic method is geared towards finding (a small number of) high frequency errors in large datasets, often caused by tokenisation errors. Their algorithm extracts sequences that have to be manually sorted into linguistically sound patterns and erroneous patterns.

Loftsson (2009) tests several methods for error detection in POS tagged data, one of them based on the predictions of an ensemble of 5 POS taggers. Error candidates are those tokens for which the predictions of all ensemble taggers agree but that diverge from the manual annotation. This simple method yields a precision of around 16% (no. of true positives amongst the error candidates), but no information is given about the recall of the method, i.e. how many of the errors in the corpus have been identified. Rehbein (2014) extends the work of Loftsson (2009) by training a CRF classifier on the output of ensemble POS taggers. This results in a much higher precision, but with low recall (for a precision in the range of 50-60% they report a recall between 10-20%).

Also related is work that addresses the issue of learning in the presence of annotation noise (Reidsma and Carletta, 2008; Beigman and Klebanov, 2009; Bekker and Goldberger, 2016). The main difference to our work lies in its different focus. While our focus is on identifying errors with the goal of improving the quality of an existing *language resource*, their main objective is to improve the accuracy of a *machine learning system*.

In the next section we describe the approach of Hovy et al. (2013) and present our adaptation for semi-supervised error detection that combines Bayesian inference with active learning.

Algorithm 1 AL with variational inference

```
Input: classifier predictions A
 1: for 1 ... n iterations do
 2:     procedure GENERATE(A)
 3:         for i = 1 ... n classifiers do
 4:             T_i ~ Uniform
 5:             for j = 1 ... n instances do
 6:                 S_ij ~ Bernoulli(1 − θ_j)
 7:                 if S_ij = 0 then
 8:                     A_ij = T_i
 9:                 else
10:                     A_ij ~ Multinomial(ξ_j)
11:                 end if
12:             end for
13:         end for
14:         return posterior entropies E
15:     end procedure
16:     procedure ACTIVELEARNING(A)
17:         rank J → max(E)
18:         for j = 1 ... n instances do
19:             Oracle → label(j);
20:             select random classifier i;
21:             update model prediction for i(j);
22:         end for
23:     end procedure
24: end for
```

3 Method

3.1 Modelling human annotators

Hovy et al. (2013) develop a generative model for Multi-Annotator Competence Estimation (MACE) to determine which annotators to trust in a crowdsourcing setting (Algorithm 1, lines 2-15). MACE implements a simple graphical model where the input consists of all annotated instances I by a set of J annotators. The model generates the observed annotations A as follows. The (unobserved) "true" label T_i is sampled from a uniform prior, based on the assumption that the annotators always try to predict the correct label and thus the majority of the annotations should, more often than not, be correct. The model is unsupervised, meaning that no information on the real gold labels is available.

To model each annotator's behaviour, a binary variable S_{ij} (also unobserved) is drawn from a Bernoulli distribution that describes whether annotator j is trying to predict the correct label for instance i or whether s/he is just spamming (a behaviour not uncommon in a crowdsourcing setting). If S_{ij} is 0, the "true" label T_i is used to generate the annotation A_{ij}. If S_{ij} is 1, the predicted label A_{ij} for instance i comes from a multinomial distribution with parameter vector ξ_j.

The model parameter θ_j can be interpreted as a "trustworthiness" parameter that describes the probability that annotator j predicts the correct label. ξ_j, on the other hand, contains information about the actual behaviour of annotator j in the case that the annotator is not trying to predict the correct label.

The model parameters are learned by maximizing the marginal likelihood of the observed data, using Expectation Maximization (EM) (Dempster et al., 1977) and Bayesian variational inference. Bayesian inference is used to provide the model with priors on the annotators' behaviour and yields improved correlations over EM between the model estimates and the annotators' proficiency while keeping accuracy high. For details on the implementation and parameter settings refer to Hovy et al. (2013) and Johnson (2007).

We adapt the model of Hovy et al. (2013) and apply it to the task of error detection in *automatically* labelled text. To that end, we integrate the variational model in an active learning (AL) setting, with the goal of identifying as many errors as possible while keeping the number of instances to be checked as small as possible. The tasks we chose in our experiments are POS tagging and NER, but our approach is general and can easily be applied to other classification tasks.

3.2 Active learning

Active learning (Cohn et al., 1996) is a semi-supervised framework where a machine learner is trained on a small set of carefully selected instances that are informative for the learning process, and thus yield the same accuracy as when training the learner on a larger set of randomly chosen examples. The main objective is to save time and money by minimising the need for manual annotation. Many different measures of informativeness as well as selection strategies for AL have been proposed in the literature, amongst them *query-by-committee* learning (Seung et al., 1992).

The *query-by-committee* (QBC) approach uses a classifier ensemble (or committee) and selects the instances that show maximal disagreement between the predictions of the committee members. These instances are assumed to provide new information for the learning process, as the classifiers are most unsure about how to label them. The selected instances are then presented to the *oracle* (the human annotator), to be manually disambiguated and added to the training data. Then the classifier committee is retrained on the extended training set and the next AL iteration starts.

The query-by-committee strategy calls to mind previous work on error detection in *manually* labelled text that made use of disagreements between the predictions of a classifier ensemble and the manually assigned tag, to identify potential annotation errors in the data (Loftsson, 2009). This approach works surprisingly well, and the trade-off between precision and recall can be balanced by adding a threshold (i.e. by considering all instances where at least N of the ensemble classifiers disagree with the manually assigned label). Loftsson (2009) reports a precision of around 16% for using a committee of five POS taggers to identify annotation errors (see section 2).

Let us assume we follow this approach and apply a tagger with an average accuracy of 97% to a corpus with 100,000 tokens. We can then expect around 3,000 incorrectly tagged instances in the data. Trying to identify these with a precision of 16% means that when looking at 1,000 instances of potential errors, we can only expect to see around 160 true positive cases, and we would have to check a large amount of data in order to correct a substantial part of the annotation noise. This means that this approach is not feasible for correcting large automatically annotated data.

It is thus essential to improve precision *and* recall for error detection, and our goal is to minimise the number of instances that have to be manually checked while maximizing the number of true errors in the candidate set. In what follows we show how we can achieve this by using active learning to guide variational inference for error detection.

3.3 Guiding variational inference with AL

Variational inference is a method from calculus where the posterior distribution over a set of unobserved random variables Y is approximated by a variational distribution $Q(Y)$. We start with some observed data X (a set of predictions made by our committee of classifiers) The distribution of the true labels $Y = \{y_1, y_2, ..., y_n\}$ is unknown.

As it is too difficult to work with the posterior $p(y|x)$, we try to approximate it with a much simpler distribution $q(y)$ which models y for each observed x. To that end, we define a family Q of distributions that are computationally easy to work with, and pick the q in Q that best approximates the posterior, where $q(y)$ is called the variational approximation to the posterior $p(y|x)$.

For computing variational inference, we use the implementation of Hovy et al. (2013)[2] who jointly optimise p and q using variational EM. They alternate between adjusting q given the current p (E-step) and adjusting p given the current q (M-step). In the E-step, the objective is to find the q that minimises the divergence between the two distributions, $D(q||p)$. In the M-step, we keep q fixed and try to adjust p. The two steps are repeated until convergence.

We extend the model for use in AL as follows (Algorithm 1). We start with the predictions from a classifier ensemble and learn a variational inference model on the data (lines 2-15). We then use the posterior entropies according to the current model, and select the c instances with the highest entropies for manual validation. These instances are presented to the oracle who assigns the true label. We save the predictions made by the human annotator and, in the next iteration, use them in the variational E-step as a prior to guide the learning process. In addition, we randomly pick one of the classifiers and update its prediction by replacing the classifier's prediction with the label we obtained from the oracle.[3] In the next iteration, we train the variational model on the updated predictions. By doing this, we also gradually improve the quality of the input to the variational model.

In a typical AL approach, the main goal is to improve the *classifiers' accuracy* on new data. In contrast to that, our approach aims at increasing *precision and recall* for error detection in *automatically* labelled data, and thus at minimising the time needed for manual correction. Please note that in our model we do *not* need to retrain the classifiers used for predicting the labels but only retrain the model that determines which of the classifiers' predictions we can trust. This is crucial as it saves time and makes it easy to integrate the approach in a realistic scenario with a real human annotator in the loop.

4 Data and setup

In our first experiment (§5.1) we want to assess the benefits of our approach for finding POS errors in standard newspaper text (in-domain setting) where we have plenty of training data. For this setting, we use the English Penn Treebank, annotated with parts-of-speech, for training and testing.

In the second experiment (§5.2) we apply our method in an out-of-domain setting where we want to detect POS errors in text from new domains where no training data is yet available (out-of-domain setting). For this we use the Penn Treebank as training data, and test our models on data from the English Web treebank (Bies et al., 2012).

To test our method on a different task and a new language, we apply it to Named Entity Recognition (NER) (experiment 3, §5.3), using out-of-domain data from the Europarl corpus.[4] The data was created by Faruqui and Pado (2010) and includes the first two German Europarl session transcripts, manually annotated with NER labels according to the CoNLL 2003 annotation guidelines (Tjong Kim Sang and De Meulder, 2003).

The first three experiments are simulation studies. In our last experiment (§5.4), we show that our method also works well in a real AL scenario with a human annotator in the loop. For this we use the out-of-domain setting from the second experiment and let the annotators correct POS errors in two web genres (*answers, weblogs*) from the English Web treebank.

4.1 Tools for preprocessing

For the POS tagging experiments, we use the following taggers to predict the labels:

- bi-LSTM-aux (Plank et al., 2016)
- HunPos (Halácsy et al., 2007)
- Stanford postagger (Toutanova et al., 2003)
- SVMTool (Giménez and Màrquez, 2004)
- TreeTagger (Schmid, 1999)
- TWeb (Ma et al., 2014)
- Wapiti (Lavergne et al., 2010)

The taggers implement a range of different algorithms, including HMMs, decision trees, SVMs, maximum entropy and neural networks. We train the taggers on subsets of 20,000 sentences extracted from the standard training set of the PTB (sections 00-18)[5] and use the development and test set (sections 19-21 and 22-24) for testing. The training times of the taggers vary considerably, ranging from a few seconds (HunPos) to several

[2]MACE is available for download from http://www.isi.edu/publications/licensed-sw/mace

[3]We also experimented with updating more than one classifier, which resulted in lower precision and recall. We take this as evidence for the importance of keeping the variance in the predictions high.

[4]The NER taggers have been trained on written German data from the HGC and DeWaC corpora (see §4.1).

[5]For taggers that use a development set during training, we also extract the dev data from sections 00-18 of the PTB.

hours. This is a problem for the typical AL setting where it is crucial not to keep the human annotators waiting for the next instance while the system retrains. A major advantage of our setup is that we do not need to retrain the baseline classifiers as we only use them once, for preprocessing, before the actual error detection starts.

For the NER experiment, we use tools for which pretrained models for German are available, namely GermaNER (Benikova et al., 2015), and the StanfordNER system (Finkel and Manning, 2009) with models trained on the HGC and the DeWaC corpus (Baroni et al., 2009; Faruqui and Padó, 2010).[6]

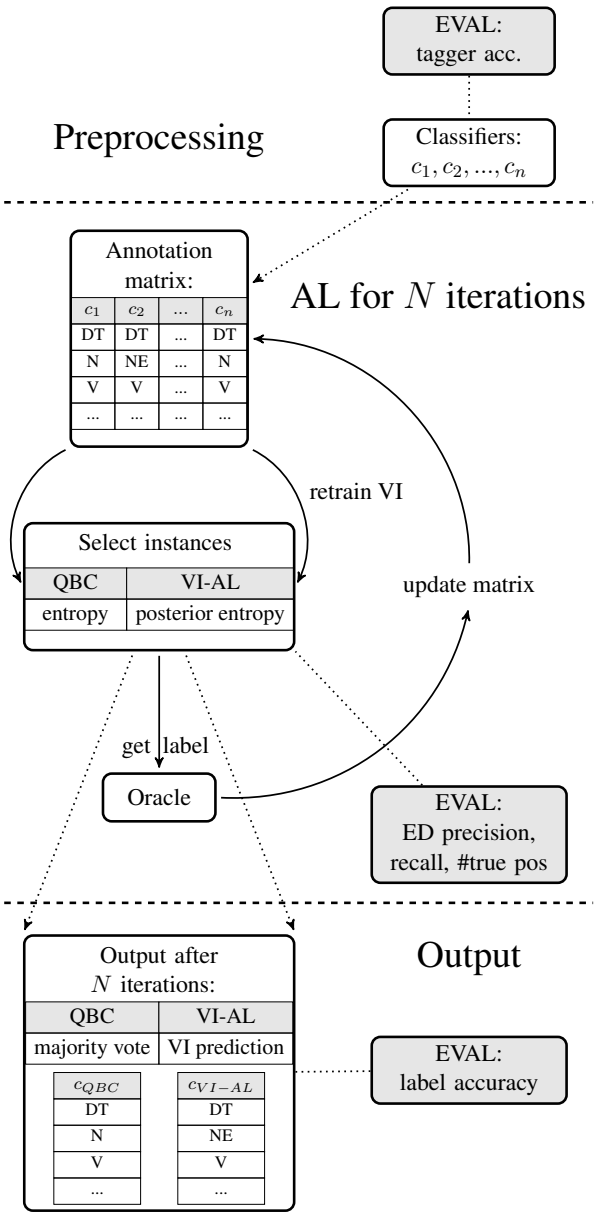
4.2 Evaluation measures

We report results for different evaluation measures to asses the usefulness of our method. First, we report **tagger accuracy** on the data, obtained during preprocessing (figure 1). This corresponds to the accuracy of the labels in the corpus *before* error correction (baseline accuracy). **Label accuracy** measures the accuracy of the labels in the corpus *after* N iterations of error correction. Please note that we do not retrain the tools used for preprocessing, but assess the quality of the data after N iterations of manual inspection and correction.

We also report precision and recall for the error detection itself. **True positives (tp)** refers to the number of instances selected for correction during AL that were actual annotation errors. We compute **Error detection (ED) precision** as the number of true positives divided by the number of all instances selected for error correction during N iterations of AL, and **recall** as the ratio of correctly identified errors to all errors in the data.

4.3 Baseline accuracies

Table 1 shows the accuracies for the individual POS taggers used in experiments 1, 2 and 4. Please note that this is not a fair comparison as each tagger was trained on a *different* randomly sampled subset of the data and, crucially, we did not optimise any of the taggers but used default settings in all experiments.[7] The accuracies of the



EVAL	*Evaluation measures used in the experiments*
tagger acc	Accuracy of preprocessing classifiers on the data.
label acc	Label accuracy in the corpus after N iterations of AL.
true pos	No. of instances selected for correction that are true errors.
ED prec	No. of true pos. / all instances selected for error correction.
recall	Correctly identified errors / all errors in the corpus.

Figure 1: Error detection procedure and overview over different evaluation measures for assessing the quality of error identification.

baseline taggers vary between 94-97%, with an average accuracy of 95.8%. The majority baseline yields better results than the best individual tagger, with an accuracy of 97.3%. Importantly, the predictions made by the variational inference model (MACE) are in the same range as the majority baseline and thus *do not improve over the*

[6]To increase the number of annotators we use an older version of the StanfordNER (2009-01-16) and a newer version (2015-12-09), with both the DeWaC and HGC models, resulting in a total of 5 annotators for the NER task.

[7]Please note that the success of our method relies on the variation in the ensemble predictions, and thus improving the accuracies for preprocessing is not guaranteed to improve precision for the error detection task.

Tagger	Acc.
bilstm	97.00
hunpos	96.18
stanford	96.93
svmtool	95.86
treetagger	94.35
tweb	95.99
wapiti	94.52
avg.	95.83
majority vote	**97.28**
MACE	97.27

Table 1: **Tagger accuracies** for POS taggers trained on subsamples of the WSJ with 20,000 tokens (for the majority vote, ties were broken randomly).

majority vote on the *automatically* labelled data.

To be able to run the variational inference model in an AL setting, we limit the size of the test data (the size of the pre-annotated data to be corrected) to batches of 5,000 tokens. This allows us to reduce the training time of the variational model and avoid unnecessary waiting times for the oracle.

For NER (experiment 3), in contrast to POS tagging, we have a much smaller label set with only 5 labels (PER, ORG, LOC, MISC, O), and a highly skewed distribution where most of the instances belong to the negative class (O). To ensure a sufficient number of NEs in the data, we increase the batch size and use the whole out-of-domain testset with 4,395 sentences in the experiment.[8] The overall accuracies of the different NER models are all in the range of 97.7-98.6%. Results for individual classes, however, vary considerably between the different models.

5 Results

5.1 Experiment 1: In-domain setting

In our first experiment, we explore the benefits of our AL approach to error detection in a setting where we have a reasonably large amount of training data, and where training and test data come from the same domain (in-domain setting).

We implement two selection strategies. The first one is a *Query-by-Committee* approach (QBC) where we use the disagreements in the predictions of our tagger ensemble to identify potential errors. For each instance i, we compute the entropy over the predicted labels M by the 7 taggers and select the N instances with the highest entropy (Equation 1).

$$H = -\sum_{m=1}^{M} P(y_i = m) \log P(y_i = m) \quad (1)$$

For each selected instance, we then replace the label predicted by majority vote with the gold label. Please note that the selected instances might already have the correct label, and thus the replacement does not necessarily increase accuracy but only does so when the algorithm selects a true error. We then evaluate the accuracy of the majority predictions after updating the N instances ranked highest for entropy[9] (figure 1).

We compare the QBC setting to our integrated approach where we guide the generative model with human supervision. Here the instances are selected according to their posterior entropy as assigned by the variational model, and after being disambiguated by the oracle, the predictions of a randomly selected classifier are updated with the oracle tags. We run the AL simulation for 500 iterations[10] and select one new instance in each iteration. After replacing the predicted label for this instance by the gold label, we retrain the variational model and select the next instance, based on the new posterior probabilities learned on the modified dataset. We refer to this setting as VI-AL.

	QBC		VI-AL	
N	**label acc**	**ED prec**	**label acc**	**ED prec**
0	97.58	-	97.56	-
100	97.84	13.0	98.42	**41.0**
200	97.86	7.0	98.90	**33.0**
300	97.90	5.3	99.16	**26.3**
400	97.82	3.0	99.26	**21.0**
500	97.92	3.4	99.34	**17.6**

Table 2: **Label accuracies** on 5,000 tokens of WSJ text after N iterations, and precision for error detection (ED prec).

Table 2 shows POS tag accuracies (lab-acc) *after* N iterations of active learning. For the QBC setting, we see a slight increase in label accuracy of 0.3% (from 97.6 to 97.9) after manually validating 10% of the instances in the data. For the first 100 instances, we see a precision of 13% for error

[8] This is possible because, given the lower number of class labels, the training time for the VI-AL model for NER is much shorter than for the POS data.

[9] Please recall that, in contrast to a traditional QBC active learning approach, we do not retrain the classifiers but only update the labels predicted by the classifiers.

[10] We stopped after 500 iterations as this was enough to detect nearly all errors in the WSJ data.

	answer	email	newsg.	review	weblog
bilstm	85.5	84.2	86.5	86.9	89.6
hun	88.5	87.4	89.2	89.7	92.2
stan	**89.0**	88.1	**89.9**	90.7	93.0
svm	87.4	86.1	88.2	88.8	91.3
tree	86.8	85.6	87.1	88.7	87.4
tweb	88.2	87.1	88.5	89.3	92.0
wapiti	85.2	82.4	84.6	86.5	87.3
avg.	87.2	85.8	87.7	88.7	90.4
major.	87.4	**88.8**	89.1	90.9	93.8
MACE	87.4	88.6	89.1	**91.0**	**93.9**

Table 3: **Tagger accuracies** on different web genres (trained on the WSJ); avg. accuracy, accuracy for majority vote (major.), and accuracy for MACE.

N	answer	email	newsg	review	weblog
0	87.4	88.6	89.1	91.0	93.9
100	88.9	90.0	90.4	92.2	95.2
200	90.3	91.1	91.3	93.4	96.2
300	91.6	92.2	92.0	94.4	97.2
400	92.9	93.3	92.8	95.4	97.5
500	93.9	94.0	93.5	96.0	97.8
600	94.8	94.9	93.9	96.5	97.9
700	95.6	95.6	94.1	96.9	98.0
800	96.2	95.9	94.7	97.3	98.4
900	96.7	96.2	94.9	97.7	98.6
1000	97.0	96.8	95.1	97.9	98.6

Table 4: Increase in POS **label accuracy** on the web genres (5,000 tokens) after N iterations of error correction with VI-AL.

detection. In the succeeding iterations, the precision slowly decreases as it gets harder to identify new errors. We even observe a slight decrease in label accuracy after 400 iterations that is due to the fact that ties are broken randomly and thus the vote for the same instance can vary between iterations.

Looking at the AL setting with variational inference, we also see the highest precision for identifying errors during the first 100 iterations. However, the precision for error dection is more than 3 times as high as for QBC (41% vs. 13%), and we are still able to detect new errors during the last 100 iterations. This results in an increase in POS label accuracy in the corpus from 97.56% to 99.34%, a near perfect result.

To find out what error types we were not able to identify, we manually checked the remaining 33 errors that we failed to detect in the first 500 iterations. Most of those are cases where an adjective (JJ) was mistaken for a past participle (VBN).

(2) Companies were **closed**$_{JJ/VBN}$ yesterday

Manning (2011), who presents a categorization of the type of errors made by a state-of-the-art POS tagger on the PTB, refers to the error type in example (2) as *underspecified/unclear*, a category that he applies to instances where "the tag is underspecified, ambiguous, or unclear in the context". These cases are also hard to disambiguate for human annotators, so it is not surprising that our system failed to detect them.

5.2 Experiment 2: Out-of-domain setting

In the second experiment, we test how our approach performs in an out-of-domain setting. For this, we use the English Web treebank (Bies et al., 2012), a corpus of over 250,000 words of English weblogs, newsgroups, email, reviews and question-answers manually annotated for parts-of-speech and syntax. Our objective is to develop and test a method for error detection that can also be applied to out-of-domain scenarios for creating and improving language resources when *no in-domain training data is available*. We thus abstain from retraining the taggers on the web data and use the tools and models from experiment 1 (§5.1) as is, trained on the WSJ. As the English Web treebank uses an extended tagset with additional tags for URLs and email addresses etc., we allow the oracle to assign new tags unknown to the preprocessing classifiers. In a traditional AL setting, this would not be possible, as all class labels have to be known from the start. In our setting, however, this can be easily implemented.

For each web genre, we extract samples of 5,000 tokens and run an active learning simulation with 500 iterations, where in each iteration one new instance is selected and disambiguated. After each iteration, we update the variational model and the predictions of a randomly selected classifier, as described in Section 5.1.

Table 3 shows the performance of the WSJ-trained taggers on the web data. As expected, the results are much lower than the ones from the in-domain setting. This allows us to explore the behaviour of our error detection approach under different conditions, in particular to test our approach on tag predictions of a lower quality. The last three rows in Table 3 give the average tagger accuracy, the accuracy for the majority vote for the ensemble (not to be confused with QBC), and the accuracy we get when using the predictions from the variational model *without AL* (MACE).

	QBC			VI-AL		
N	**# tp**	**ED prec**	**rec**	**# tp**	**ED prec**	**rec**
100	85	**85.0**	**13.5**	75	75.0	11.9
200	148	**74.0**	**23.5**	146	73.0	23.2
300	198	66.0	31.4	212	**70.7**	**33.6**
400	239	59.7	37.9	278	**69.5**	**44.1**
500	282	56.4	44.8	323	**64.6**	**51.3**
600	313	52.2	49.7	374	**62.3**	**59.4**
700	331	47.3	52.5	412	**58.9**	**65.4**
800	355	44.4	56.3	441	**55.1**	**70.0**
900	365	40.6	57.9	465	**51.7**	**73.8**
1000	371	37.1	58.9	484	**48.4**	**76.8**

Table 5: No. of true positives (# tp), precision (ED prec) and recall for error detection on 5,000 tokens from the *answers* set after N iterations.

We can see that the majority baseline often, but not always succeeds in beating the best individual tagger. Results for MACE are more or less in the same range as the majority vote, same as in experiment 1, but *do not improve over the baseline*.

Next, we employ AL in the out-of-domain setting (Tables 4, 5 and 6). Table 4 shows the increase in POS label accuracy for the five web genres after running N iterations of AL with variational inference (VI-AL). Table 5 compares the results of the two selection strategies, QBC and VI-AL, on the *answers* subcorpus after an increasing number of AL iterations.[11] Table 6 completes the picture by showing results for error detection for all web genres, for QBC and VI-AL, after inspecting 10% of the data (500 iterations).

Table 4 shows that using VI-AL for error detection results in a substantial increase in POS label accuracy for all genres. VI-AL still detects new errors after a high number of iterations, without retraining the ensemble taggers. This is especially useful in a setting where no labelled target domain data is yet available.

Table 5 shows the number of true positives amongst the selected error candidates as well as precision and recall for error detection for different stages of AL on the *answers* genre. We can see that during the early learning stages, both selection strategies have a high precision and QBC beats VI-AL. After 200 iterations it becomes more difficult to detect new errors, and the precision for both methods decreases. The decrease, however, is much slower for VI-AL, leading to higher precision after the initial rounds of training, and the gap in results becomes more and more pronounced.

[11] Due to space restrictions, we can only report detailed results for one web genre. Results for the other web genres follow the same trend (see Tables 4 and 6).

	QBC			VI-AL		
	# tp	**ED prec**	**rec**	**# tp**	**ED prec**	**rec**
answer	282	56.4	44.8	323	**64.6**	**51.3**
email	264	**52.8**	**47.1**	261	52.2	46.6
newsg.	195	39.0	36.0	214	**42.8**	**39.6**
review	227	45.4	49.7	255	**51.0**	**55.8**
weblog	166	33.2	54.6	196	**39.2**	**64.5**

Table 6: No. of true positives (# tp), precision (ED prec) and recall for error detection on 5,000 tokens after 500 iterations on all web genres.

After 600 iterations, VI-AL beats QBC by more than 10%, thus resulting in a lower number of instances that have to be checked to obtain the same POS accuracy in the final dataset. Looking at recall, we see that by manually inspecting 10% of the data VI-AL manages to detect more than 50% of all errors, and after validating 20% of the data, we are able to eliminate 75% of all errors in the corpus. In contrast, QBC detects less than 60% of the annotation errors in the dataset.

In the out-of-domain setting where we start with low-quality POS predictions, we are able to detect errors in the data with a much higher precision than in the in-domain setting, where the number of errors in the dataset is much lower. Even after 1,000 iterations, the precision for error detection is close to 50% in the answers data.

Table 6 shows that the same trend appears for the other web genres, where we observe a substantially higher precision and recall when guiding AL with variational inference (VI-AL). Only on the email data are the results below the ones for QBC, but the gap is small.

5.3 Experiment 3: A new task (and language)

We now want to test if the approach generalises well to other classification tasks, and also to new languages. To that end, we apply our approach to the task of Named Entity Recognition (NER) on German data (§4).

Table 7 shows results for error detection for NER. In comparison to the POS experiments, we observe a much lower recall, for both QBC and VI-AL. This is due to the larger size of the NER testset which results in a higher absolute number of errors in the data. Please bear in mind that recall is computed as the ratio of correctly identified errors to all errors in the testset (here we have a total of 110,405 tokens in the test set which means that we identified >35% of all errors by querying *less than 1% of the data*). Also note that the overall number of errors is higher in the QBC setting (1,756

	QBC			VI-AL		
N	**# tp**	**ED prec**	**rec**	**# tp**	**ED prec**	**rec**
100	54	54.0	3.1	76	**76.0**	**4.7**
200	113	56.5	6.4	155	**77.5**	**9.6**
300	162	54.0	9.2	217	**72.3**	**13.4**
400	209	52.2	11.9	297	**74.2**	**18.2**
500	274	54.8	15.6	352	**70.4**	**22.3**
600	341	56.8	19.4	409	**68.2**	**25.5**
700	406	58.0	23.1	452	**64.6**	**27.8**
800	480	60.0	27.3	483	**60.4**	**29.8**
900	551	**61.2**	31.4	512	56.9	**31.9**
1000	617	**61.7**	35.1	585	58.5	**35.8**
1000	remaining errors:1,139			remaining errors:1,043		

Table 7: Error detection results on the GermEval 2014 NER testset after N iterations (true positives, ED precision and recall).

errors) than in the VI-AL setting (1,628 errors), as in the first setting we used a majority vote for generating the data pool while in the second setting we relied on the predictions of MACE. For POS tagging, we did not observe a difference between the initial data pools (Table 3). For NER, however, the initial predictions of MACE are better than the majority vote.

During the first 800 iterations, precision for VI-AL is much higher than for QBC, but then slowly decreases. For QBC, however, we see the opposite trend. Here precision stays in the range of 52-56% for the first 600 iterations. After that, it slowly increases, and during the last iterations QBC precision outperforms VI-AL.

Recall, however, is higher for the VI-AL model, for *all* iterations. This means that even if precision is slightly lower than in the QBC setting after 800 iterations, it is still better to use the VI-AL model. For comparison, in the QBC setting we still have 1,139 errors left in the corpus after 1,000 iterations, while for VI-AL the number of errors remaining in the data is much lower (1,043).

5.4 Experiment 4: A real-world scenario

In our final experiment, we test our approach in a real-world scenario with a human annotator in the loop. To that end, we let two linguistically trained human annotators correct POS errors identified by AL. We use the out-of-domain data from experiment 2 (§5.2), specifically the *answers* and *weblog* subcorpora.

We run two VI-AL experiments where the oracle is presented with new error candidates for 500 iterations. The time needed for correction was 135 minutes (annotator 1, answers) and 157 minutes (annotator 2, weblog) for correcting 500 instances each. This includes the time needed to consult the annotation guidelines, as both annotators had no prior experience with the extended English Web treebank guidelines. We expect that the amount of time needed for correction will decrease when the annotators become more familiar with the annotation scheme. Results are shown in Table 8.

	VI-AL *with human annotator*					
	answers			*weblog*		
N	**# tp**	**ED prec**	**rec**	**# tp**	**ED prec**	**rec**
100	71	68.0	10.8	62	62.0	20.3
200	103	63.5	20.2	112	56.0	36.7
300	177	58.0	27.6	156	52.0	51.1
400	224	55.3	35.1	170	42.5	55.7
500	259	51.2	40.6	180	36.0	59.0

Table 8: POS results for VI-AL with a **human annotator** on 2 web genres (true positives, precision and recall for error detection on 5,000 tokens)

As expected, precision as well as recall are lower for the human annotators as compared to the simulation study (Table 6). However, even with some annotation noise we were able to detect more than 40% of all errors in the *answers* data and close to 60% of all errors in the *weblog* corpus, by manually inspecting only 10% of the data. This results in an increase in POS label accuracy from 88.8 to 92.5% for the *answers* corpus and from 93.9 to 97.5% for the *weblogs*, which is very close to the 97.8% we obtained in the simulation study (Table 4).

6 Conclusions

In the paper, we addressed a severely understudied problem, namely the detection of errors in *automatically* annotated language resources. We present an approach that combines an unsupervised generative model with human supervision in an AL framework. Using POS tagging and NER as test cases, we showed that our model can detect errors with high precision *and* recall, and works especially well in an out-of-domain setting. Our approach is language-agnostic and can be used without retraining the classifiers, which saves time and is of great practical use in an AL setting. We also showed that combining an unsupervised generative model with human supervision is superior to using a query-by-committee strategy for AL.

Our system architecture is generic and can be applied to any classification task, and we expect it to be of use in many annotation projects, especially when dealing with non-standard data or in out-of-domain settings.

Acknowledgments

This research has been conducted within the Leibniz Science Campus "Empirical Linguistics and Computational Modeling", funded by the Leibniz Association under grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art (MWK) of the state of Baden-Württemberg.

References

Bharat Ram Ambati, Mridul Gupta, Rahul Agarwal, Samar Husain, and Dipti Misra Sharma. 2011. Error detection for treebank validation. In *Proceedings of the 9th Workshop on Asian Language Resources*. Chiang Mai, Thailand, ALR9, pages 23–30.

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

Eyal Beigman and Beata Beigman Klebanov. 2009. Learning with annotation noise. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore, ACL'09, pages 280–287.

Alan Joseph Bekker and Jacob Goldberger. 2016. Training deep neural-networks based on unreliable labels. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*. ICASSP.

Darina Benikova, Seid Muhie Yimam, Prabhakaran Santhanam, and Chris Biemann. 2015. GermaNER: Free open German Named Entity Recognition tool. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology (GSCL'15)*. Essen, Germany, pages 31–38.

Jiang Bian, Yandong Liu, Ding Zhou, Eugene Agichtein, and Hongyuan Zha. 2009. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *Proceedings of the 18th International Conference on World Wide Web*. Madrid, Spain, WWW'09, pages 51–60.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank. Technical Report LDC2012T13, Philadelphia: Linguistic Data Consortium.

David Cohn, Zoubin Ghahramani, and Michael Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research* 4:129–145.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* 39(1):1–38.

Marcus Dickinson and Detmar W. Meurers. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*. Budapest, Hungary, EACL'03, pages 107–114.

Eleazar Eskin. 2000. Automatic corpus correction with anomaly detection. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*. NAACL'00, pages 148–153.

Manaal Faruqui and Sebastian Padó. 2010. Training and evaluating a German Named Entity Recognizer with semantic generalization. In *Proceedings of the Conference on Natural Language Processing*. KONVENS'10, pages 129–133.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested Named Entity Recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP'09, pages 141–150.

Jesús Giménez and Lluís Màrquez. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. Lisbon, Portugal, LREC, pages 43–46.

Péter Halácsy, András Kornai, and Csaba Oravecz. 2007. HunPos: An open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Prague, Czech Republic, ACL'07, pages 209–212.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia, USA, NAACL-HLT'13, pages 1120–1130.

Mark Johnson. 2007. Why doesn't EM find good HMM pos-taggers? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Prague, Czech Republic, EMNLP'07, pages 296–305.

Pavel Kveton and Karel Oliva. 2002. (Semi-)automatic detection of errors in pos-tagged corpora. In *Proceedings of the 19th International Conference on Computational Linguistics*. Taipei, Taiwan, COLING'02, pages 1–7.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden, ACL'10, pages 504–513.

Hrafn Loftsson. 2009. Correcting a POS-tagged corpus using three complementary methods. In *Proceedings of the 12th Conference of the European Chapter*

of the ACL. Athens, Greece, EACL'09, pages 523–531.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014. Tagging the web: Building a robust web tagger with neural network. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland, ACL'14, pages 144–154.

Christopher D. Manning. 2011. Part-of-speech tagging from 97linguistics? In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing*. Tokyo, Japan, CICLing'11, pages 171–189.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, ACL'16, pages 412–418.

Ines Rehbein. 2014. POS error detection in automatically annotated corpora. In *Proceedings of the 8th Linguistic Annotation Workshop*. LAW VIII, pages 20–28.

Dennis Reidsma and Jean Carletta. 2008. Reliability measurement without limits. *Computational Linguistics* 34(3):319–326.

Vitor Rocio, Joaquim Silva, and Gabriel Lopes. 2007. Detection of strange and wrong automatic part-of-speech tagging. In *Proceedings of the Aritficial Intelligence 13th Portuguese Conference on Progress in Artificial Intelligence*. Guimarães, Portugal, EPIA07, pages 683–690.

Helmut Schmid. 1999. Improvements in part-of-speech tagging with an application to German. In Susan Armstrong, Kenneth Church, Pierre Isabelle, Sandra Manzi, Evelyne Tzoukermann, and David Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*, Kluwer Academic Publishers, Dordrecht, volume 11 of *Text, Speech and Language Processing*, pages 13–26.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. Pittsburgh, Pennsylvania, USA, COLT'92, pages 287–294.

Victor Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD'08, pages 614–622.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii, EMNLP'08, pages 254–263.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the SIGNLL Conference on Computational Natural Language Learning*. Edmonton, Canada, CoNLL'03, pages 142–147.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Edmonton, Canada, NAACL'03, pages 173–180.

Hans van Halteren. 2000. The detection of inconsistency in manually tagged text. In *Proceedings of the COLING-2000 Workshop on Linguistically Interpreted Corpora*. Centre Universitaire, Luxembourg, pages 48–55.

Liyue Zhao, Gita Sukthankar, and Rahul Sukthankar. 2011. Incremental relabeling for active learning with noisy crowdsourced annotations. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing*. PASSAT and SocialCom, pages 728–733.

Abstractive Document Summarization with a Graph-Based Attentional Neural Model

Jiwei Tan, Xiaojun Wan and **Jianguo Xiao**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{tanjiwei,wanxiaojun,xiaojianguo}@pku.edu.cn

Abstract

Abstractive summarization is the ultimate goal of document summarization research, but previously it is less investigated due to the immaturity of text generation techniques. Recently impressive progress has been made to abstractive sentence summarization using neural models. Unfortunately, attempts on abstractive document summarization are still in a primitive stage, and the evaluation results are worse than extractive methods on benchmark datasets. In this paper, we review the difficulties of neural abstractive document summarization, and propose a novel graph-based attention mechanism in the sequence-to-sequence framework. The intuition is to address the saliency factor of summarization, which has been overlooked by prior works. Experimental results demonstrate our model is able to achieve considerable improvement over previous neural abstractive models. The data-driven neural abstractive method is also competitive with state-of-the-art extractive methods.

1 Introduction

Document summarization is a task to generate a fluent, condensed summary for a document, and keep important information. As a useful technique to alleviate the information overload people are facing today, document summarization has been extensively investigated. Efforts on document summarization can be categorized to extractive and abstractive methods. Extractive methods produce the summary of a document by extracting sentences from the original document. They have the advantage of producing fluent sentences and preserving the meaning of original documents, but also inevitably face the drawbacks of information redundancy and incoherence between sentences. Moreover, extraction is far from the way humans write summaries.

On the contrary, abstractive methods are able to generate better summaries with the use of arbitrary words and expressions, but generating abstractive summaries is much more difficult in practice. Abstractive summarization involves sophisticated techniques including meaning representation, content organization, and surface realization. Each of these techniques has large space to be improved (Yao et al., 2017). Due to the immaturity of natural language generation techniques, fully abstractive approaches are still at the beginning and cannot always ensure grammatical abstracts.

Recent neural networks enable an end-to-end framework for natural language generation. Success has been witnessed on tasks like machine translation and image captioning, together with the abstractive sentence summarization (Rush et al., 2015). Unfortunately, the extension of sentence abstractive methods to the document summarization task is not straightforward. Encoding and decoding for a long sequence of multiple sentences, currently still lack satisfactory solutions (Yao et al., 2017). Recent abstractive document summarization models are yet not able to achieve convincing performance, with a considerable gap from extractive methods.

In this paper, we review the key factors of document summarization, i.e., the saliency, fluency, coherence, and novelty requirements of the generated summary. Fluency is what neural generation models are naturally good at, but the other factors are less considered in previous neural abstractive models. A recent study (Chen et al., 2016) starts to consider the factor of novelty, using a distraction mechanism to avoid redundancy. As far as we

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1171–1181
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1108

know, however, saliency has not been addressed by existing neural abstractive models, despite its importance for summary generation.

In this work, we study how neural summarization models can discover the salient information of a document. Inspired by the graph-based extractive summarization methods, we introduce a novel graph-based attention mechanism in the encoder-decoder framework. Moreover, we investigate the challenges of accepting and generating long sequences for sequence-to-sequence (seq2seq) models, and propose a new hierarchical decoding algorithm with a reference mechanism to generate the abstractive summaries. The proposed method is able to tackle the constraints of saliency, non-redundancy, information correctness, and fluency under a unified framework.

We conduct experiments on two large-scale corpora with human generated summaries. Experimental results demonstrate that our approach consistently outperforms previous neural abstractive summarization models, and is also competitive with state-of-the-art extractive methods.

We organize the paper as follows. Section 2 introduces related work. Section 3 describes our method. In Section 4 we present the experiments and have discussion. Finally in Section 5 we conclude this paper.

2 Related Work

2.1 Extractive Summarization Methods

Document summarization can be categorized to extractive methods and abstractive methods. Extractive methods extract sentences from the original document to form the summary. Notable early works include (Edmundson, 1969; Carbonell and Goldstein, 1998; McDonald, 2007). In recent years much progress has also been made under traditional extractive frameworks (Li et al., 2013; Dasgupta et al., 2013; Nishikawa et al., 2014).

Neural networks have also been widely investigated on the extractive summarization task. Earlier works explore to use deep learning techniques in the traditional framework (Kobayashi et al., 2015; Yin and Pei, 2015; Cao et al., 2015a,b). More recent works predict the extraction of sentences in a more data-driven way. Cheng and Lapata (2016) propose an encoder-decoder approach where the encoder learns the representation of sentences and documents while the decoder classifies each sentence using an attention mechanism. Nallapati et al. (2017) propose a recurrent neural network (RNN)-based sequence model for extractive summarization of documents. Neural sentence extractive models are able to leverage large-scale training data and achieve performance better than traditional extractive summarization methods.

2.2 Abstractive Summarization Methods

Abstractive summarization aims at generating the summary based on understanding the input text. It involves multiple subproblems like simplification, paraphrasing, and fusion. Previous research is mostly restricted in one or a few of the subproblems or specific domains (Woodsend and Lapata, 2012; Thadani and McKeown, 2013; Cheung and Penn, 2014; Pighin et al., 2014; Sun et al., 2015).

As for neural network models, success is achieved on sentence abstractive summarization. Rush et al. (2015) train a neural attention model on a large corpus of news documents and their headlines, and later Chopra et al. (2016) extend their work with an attentive recurrent neural network framework. Nallapati et al. (2016) introduce various effective techniques in the RNN seq2seq framework. These neural sentence abstraction models are able to achieve state-of-the-art results on the DUC competition of generating headline-level summaries for news documents.

Some recent works investigate neural abstractive models on the document summarization task. Cheng and Lapata (2016) also adopt a word extraction model, which is restricted to use the words of the source document to generate a summary, although the performance is much worse than the sentence extractive model. Nallapati et al. (2016) extend the sentence summarization model by trying a hierarchical attention architecture and a limited vocabulary during the decoding phase. However these models still investigate few properties of the document summarization task. Chen et al. (2016) first attempt to explore the novelty factor of summarization, and propose a distraction-based attentional model. Unfortunately these state-of-the-art neural abstractive summarization models are still not competitive to extractive methods, and there are several problems remain to be solved.

3 Our Method

3.1 Overview

In this section we introduce our method. We adopt an encoder-decoder framework, which is

widely used in machine translation (Bahdanau et al., 2014) and dialog systems (Mou et al., 2016), etc. In particular, we use a hierarchical encoder-decoder framework similar to (Li et al., 2015), as shown in Figure 1. The main distinction of this work is that we introduce a graph-based attention mechanism which is illustrated in Figure 1b, and we propose a hierarchical decoding algorithm with a reference mechanism to tackle the difficulty of abstractive summary generation. In the following parts, we will first introduce the encoder-decoder framework, and then describe the graph-based attention and the hierarchical decoding algorithm.

3.2 Encoder

The goal of the encoder is to map the input document to a vector representation. A document d is a sequence of sentences $d = \{s_i\}$, and a sentence s_i is a sequence of words $s_i = \{w_{i,k}\}$. Each word $w_{i,k}$ is represented by its distributed representation $\mathbf{e}_{i,k}$, which is mapped by a word embedding matrix E_v. We adopt a hierarchical encoder framework, where we use a word encoder enc_{word} to encode the words of a sentence s_i into the sentence representation, and use a sentence encoder enc_{sent} to encode the sentences of a document d into the document representation. The input to the word encoder is the word sequence of a sentence, appended with an "<eos>" token indicating the end of a sentence. The word encoder sequentially updates its hidden state after receiving each word, as $\mathbf{h}_{i,k} = enc_{\text{word}}(\mathbf{h}_{i,k-1}, \mathbf{e}_{i,k})$. The last hidden state (after the word encoder receives "<eos>") is denoted as $\mathbf{h}_{i,-1}$, and used as the embedding representation of the sentence s_i, denoted as $\mathbf{x}_i$. A sentence encoder is used to sequentially receive the embeddings of the sentences, given by $\mathbf{h}_i = enc_{\text{sent}}(\mathbf{h}_{i-1}, \mathbf{x}_i)$. A pseudo sentence of an "<eod>" token is appended at the end of the document to indicate the end of the whole document. The hidden state after the sentence encoder receives "<eod>" is treated as the representation of the input document $\mathbf{c} = \mathbf{h}_{-1}$.

We use the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) as both the word encoder enc_{word} and sentence encoder enc_{sent}. In particular, we adopt the variant of LSTM structure in (Graves, 2013).

3.3 Decoder with Attention

The decoder is used to generate output sentences $\{s'_j\}$ according to the representation of the input sentences. We also use an LSTM-based hierarchical decoder framework to generate the summary, because the summary typically comprises several sentences. The sentence decoder dec_{sent} receives the document representation $\mathbf{c}$ as the initial state $\mathbf{h}'_0 = \mathbf{c}$, and predicts the sentence representations sequentially, by $\mathbf{h}'_j = dec_{\text{sent}}(\mathbf{h}'_{j-1}, \mathbf{x}'_{j-1})$, where $\mathbf{x}'_{j-1}$ is the encoded representation of the previously generated sentence s'_{j-1}. The word decoder dec_{word} receives a sentence representation $\mathbf{h}'_j$ as the initial state $\mathbf{h}'_{j,0} = \mathbf{h}'_j$, and predicts the word representations sequentially, by $\mathbf{h}'_{j,k} = dec_{\text{word}}(\mathbf{h}'_{j,k-1}, \mathbf{e}_{j,k-1})$, where $\mathbf{e}_{j,k-1}$ is the embedding of the previously generated word. The predicted word representations are mapped to vectors of the vocabulary size dimension, and then normalized by a softmax layer as the probability distribution of generating the words in the vocabulary. A word decoder stops when it generates the "<eos>" token and similarly the sentence decoder stops when it generates the "<eod>" token.

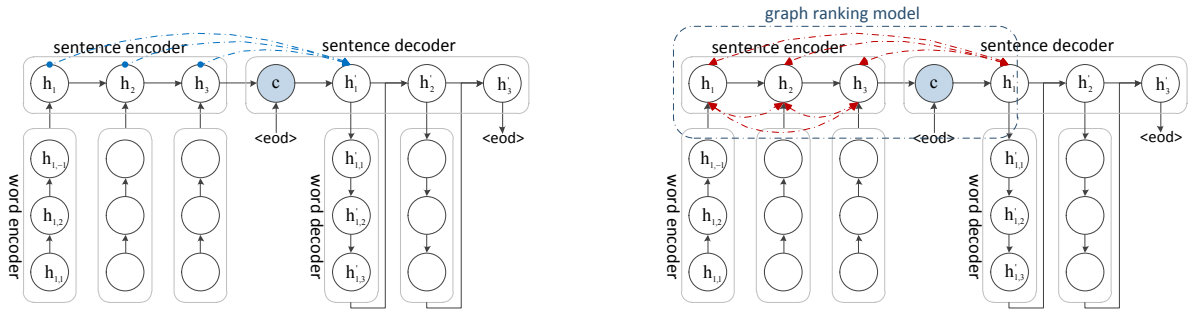
In primitive decoder models, $\mathbf{c}$ is the same for generating all the output words, which requires $\mathbf{c}$ to be a sufficient representation for the whole input sequence. The attention mechanism (Bahdanau et al., 2014) is usually introduced to alleviate the burden of remembering the whole input sequence, and to allow the decoder to pay different attention to different parts of input at different generation states. The attention mechanism sets a different $\mathbf{c}_j$ when generating sentence j, by $\mathbf{c}_j = \sum_i \alpha_i^j \mathbf{h}_i$. α_i^j indicates how much the i-th original sentence s_i contributes to generating the j-th sentence. α_i^j is usually computed as:

$$\alpha_i^j = \frac{e^{\eta\left(\mathbf{h}_i, \mathbf{h}'_j\right)}}{\sum_l e^{\eta\left(\mathbf{h}_l, \mathbf{h}'_j\right)}} \tag{1}$$

where η is the function modeling the relation between $\mathbf{h}_i$ and $\mathbf{h}'_j$. η can be defined using various functions including $\eta(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T\mathbf{b}$, $\eta(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T M \mathbf{b}$, and even a non-linear function achieved by a multi-layer neural network. In this paper we use $\eta(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T M \mathbf{b}$ where M is a parameter matrix.

3.4 Graph-based Attention Mechanism

Traditional attention computes the importance score of a sentence s_i, when generating sentence s'_j, according to the relation between the hidden state $\mathbf{h}_i$ and current decoding state $\mathbf{h}'_j$, as shown



(a) Traditional attention.

(b) Graph-based attention.

Figure 1: Hierarchical encoder-decoder framework and comparison of the attention mechanisms.

in Figure 1a. This attention mechanism is useful in scenarios like machine translation and image captioning, because the model is able to learn a relevance mapping between the input and output. However, for document summarization, it is not easy for the model to learn how to summarize the salient information of a document, i.e., which sentences are more important to a document.

To tackle this challenge, we learn from graph-based extractive summarization models TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004), which are based on the PageRank (Page et al., 1999) algorithm. These unsupervised graph-based models show good ability to identify important sentences in a document. The underlying idea is that a sentence is important in a document if it is heavily linked with many important sentences (Wan, 2010).

In graph-based extractive summarization, a graph G is constructed to rank the original sentences. The vertices V are the set of n sentences to be considered, and the edges E are the relations between the sentences, which are typically modeled by the similarity of sentences. Let $W \in \mathcal{R}^{n\times n}$ be the adjacent matrix. Then the saliency scores of the sentences are determined by making use of the global information on the graph recursively, as:

$$\mathbf{f}(t+1) = \lambda W D^{-1}\mathbf{f}(t) + (1-\lambda)\mathbf{y} \quad (2)$$

where $\mathbf{f} = [f_1, \ldots, f_n] \in \mathcal{R}^n$ denotes the rank scores of the n sentences. $\mathbf{f}(t)$ denotes the rank scores at the t-th iteration. D is a diagonal matrix with its (i,i)-element equal to the sum of the i-th column of W. Assume we use $\mathbf{h}_i$ as the representation of s_i, and $W(i,j) = \mathbf{h}_i^T M \mathbf{h}_j$, where M is a parameter matrix to be learned. λ is a damping factor. $\mathbf{y} \in \mathcal{R}^n$ with all elements equal to $1/n$. The solution of $\mathbf{f}$ can be calculated using the closed-form:

$$\mathbf{f} = (1-\lambda)(I - \lambda W D^{-1})^{-1}\mathbf{y} \quad (3)$$

In the graph model, the importance score of a sentence s_i is determined by the relation between $\mathbf{h}_i$ and the $\{\mathbf{h}_l\}$ of all other sentences. Relatively, in traditional attention mechanisms, the importance (attention) score α_i^j is determined by the relation between $\mathbf{h}_i$ and $\mathbf{h}_j^{'}$, regardless of other original sentences. In our model we hope to combine the two effects, and compute the rank scores of the original sentences regarding $\mathbf{h}_j^{'}$, so that the importance scores of original sentences are different when decoding different state $\mathbf{h}_j^{'}$, denoted by $\mathbf{f}^j$. In our model we use the scores $\mathbf{f}^j$ to compute the attention. Therefore, $\mathbf{h}_j^{'}$ should be considered in the graph model. Inspired by the query-focused graph-based extractive summarization model (Wan et al., 2007), we realize this by applying the idea of topic-sensitive PageRank (Haveliwala, 2002), which is to rank the sentences with the concern of their relevance to the topic. We treat the current decoding state $\mathbf{h}_j^{'}$ as the topic and add it into the graph as the 0-th pseudo-sentence. Given a topic T, the topic-sensitive PageRank is similar to Eq. 3 except that $\mathbf{y}$ becomes:

$$\mathbf{y}_T = \begin{cases} \frac{1}{|T|} & i \in T \\ 0 & i \notin T \end{cases} \quad (4)$$

Therefore $\mathbf{y}_T$ is always a one hot vector and only $y_0 = 1$, indicating the 0-th sentence is $s_j^{'}$. Denote W^j as the new adjacent matrix added with $\mathbf{h}_j^{'}$, and D^j as the new diagonal matrix corresponding to W^j. Then the convergence score vector $\mathbf{f}^j$ contains the importance scores for all the

input sentences when generating sentence $s_j^{'}$, as:

$$\mathbf{f}^j = (1-\lambda)(I - \lambda W^j D^{j^{-1}})^{-1}\mathbf{y}_T \quad (5)$$

The new scores $\mathbf{f}^j$ can be used to compute the graph-based attention when decoding $\mathbf{h}_j^{'}$, to find the sentences which are both globally important and relevant to current decoding state $\mathbf{h}_j^{'}$. Inspired by (Chen et al., 2016) we adopt a distraction mechanism to compute the final attention value α_i^j, which subtracts the rank scores of the previous step, to penalize the model from attending to previously attended sentences, and also help to normalize the ranked scores $\mathbf{f}^j$. The graph-based attention is finally computed as:

$$\alpha_i^j = \frac{\max(f_i^j - f_i^{j-1}, 0)}{\sum_l \left(\max(f_l^j - f_l^{j-1}, 0)\right)} \quad (6)$$

where $\mathbf{f}^0$ is initialized with all elements equal to $1/n$. The graph-based attention will only focus on those sentences ranked higher over the previous decoding step, so that it concentrates more on the sentences which are both salient and novel. Both Eq. 5 and Eq. 6 are differentiable; thus we can use the graph-based attention function Eq. 6 to replace the traditional attention function Eq. 1, and the neural model using the graph-based attention can also be trained using traditional gradient-based methods.

3.5 Model Training

The loss function $\mathcal{L}$ of the model is the negative log likelihood of generating summaries over the training set $\mathcal{D}$:

$$\mathcal{L} = \sum_{(Y,X)\in\mathcal{D}} -\log p(Y|X;\theta) \quad (7)$$

where $X = \{x_1, \ldots, x_{|X|}\}$ and $Y = \{y_1, \ldots, y_{|Y|}\}$ denote the word sequences of a document and its summary respectively, including the "<eos>" and "<eod>" tokens for structure information. Then

$$\log p(Y|X;\theta) = \sum_{\tau=1}^{|Y|} \log p\left(y_\tau | \{y_1, \ldots, y_{\tau-1}\}, \mathbf{c}; \theta\right) \quad (8)$$

and $\log p\left(y_\tau | \{y_1, \ldots, y_{\tau-1}\}, \mathbf{c}; \theta\right)$ is modeled by the LSTM encoder and decoder. We use the Adamax (Kingma and Ba, 2014) gradient-based optimization method to optimize the model parameters θ.

3.6 Decoding Algorithm

We find there are several problems during the generation of summary, including out-of-vocabulary (OOV) words, information incorrectness, error accumulation and repetition. These problems make the generated abstractive summaries far from satisfactory. In this work, we propose a hierarchical decoding algorithm with a reference mechanism to tackle these difficulties, which effectively improves the quality of generated summaries.

As OOV words frequently occur in name entities, we can first identify the entities of a document using NLP toolkit like Stanford CoreNLP[1]. Then we prefix every entity with an "@entity" token and a number indicating how many words the entity has. We hope the entity prefixes can help better deal with entities which have more than one word, and help improve the accuracy of recovering OOV words in entities. After decoding we recover the OOV words by matching entities in the original document according to the contexts.

For the hierarchical decoder, a major challenge is that same sentences or phrases are often repeated in the output. A beam search strategy may help to alleviate the repetition in a sentence, but the repetition in the whole generated summary is remained a problem. The word-level beam search is not easy to be extended to the sentence level. The reason is that the K-best sentences generated by a word decoder will mostly be similar to each other, which is also noticed by Li et al. (2016).

In this paper we propose a hierarchical beam search algorithm with a reference mechanism. The hierarchical algorithm comprises K-best word-level beam search and N-best sentence-level beam search. At the word level, the only difference to vanilla beam search is that we add an additional term to the score $\tilde{p}(y_\tau)$ of generating word y_τ, and now $score(y_\tau) = \tilde{p}(y_\tau) + \gamma\left(ref(Y_{\tau-1} + y_\tau, s_*) - ref(Y_{\tau-1}, s_*)\right)$, where $Y_{\tau-1} = \{y_1, \ldots, y_{\tau-1}\}$ and $\tilde{p}(y_\tau) = \log p\left(y_\tau | Y_{\tau-1}, \mathbf{c}; \theta\right)$. s_* is an original sentence to refer to. ref is a function which calculates the ratio of bigram overlap between two texts. The added term aims to favor the generated word y_τ with improving the bigram overlap between current generated summary $Y_{\tau-1}$ and the target orig-

[1] http://stanfordnlp.github.io/CoreNLP/

Dataset	Train	Valid	Test	D.L.	S. L.
CNN	83568	1220	1093	29.8	3.54
DailyMail	196557	12147	10396	26.0	3.84

Table 1: The statistics of the two datasets. D.L. and S.L. indicate the average number of sentences in the document and summary, respectively.

inal sentence s_*. At the word decoder level, the reference mechanism helps to both improve the information correctness and avoid redundancy. Because the reference score is based on the bigram overlap improvement to the whole generated summary $Y_{\tau-1}$, the awareness of previously generated sentences also helps alleviate sentence-level redundancy. A factor γ is introduced to control the influence of the reference mechanism. Note that because of the non-optimal search, the generated sentence will still be different to the original sentence even with an extremely large γ.

At the sentence level, N-best sentence beam is to keep the N generated sentences by referring to N different original sentences, which have the highest attention scores and have not been used as a reference. With referring to N different sentences, the N candidate sentences are guaranteed diverse. Sentence-level beam search is realized by maximizing the accumulated score of all the sentences generated.

4 Experiments

4.1 Dataset

We conduct experiments on two large-scale corpora of CNN and DailyMail, which have been widely used in neural document summarization tasks. The corpora are originally constructed in (Hermann et al., 2015) by collecting human generated abstractive highlights from the news stories in the CNN and DailyMail website. The statistics and split of the two datasets are listed in Table 1.

4.2 Implementation

We use the corpora which are already provided with labeled entities (Nallapati et al., 2016). The documents and summaries are first lowercased and tokenized, and all digit characters are replaced with the "#" symbol, similar to (Nallapati et al., 2016, 2017). We keep the 40,000 most frequently occurring words and other words are replaced with the "<OOV>" token.

We use Theano[2] for implementation. For the word encoder and decoder we use three layers of LSTM, and for the sentence encoder and decoder we use one layer of LSTM. The dimension of hidden vectors are all 512. We use pre-trained GloVe (Pennington et al., 2014) vectors[3] for the initialization of word vectors, which will be further trained in the model. The dimension of word vectors is 100. λ is set to 0.9. The parameters of Adamax are set to those provided in (Kingma and Ba, 2014). The batch size is set to 8 documents, and an epoch is set containing 10,000 randomly sampled documents. Convergence is reached within 200 epochs on the DailyMail dataset and 120 epochs on the CNN dataset. It takes about one day for every 30 epochs on a GTX-1080 GPU card. γ is tuned on the validation set and the best choice is 300. The beam sizes for word decoder and sentence decoder are 15 and 2, respectively.

4.3 Evaluation

We adopt the widely used ROUGE (Lin, 2004) toolkit for evaluation. We first compare with the reported results in (Chen et al., 2016) including various traditional extractive methods and a state-of-the-art abstractive model (Distraction-M3) on the CNN dataset, as shown in Table 2. Uni-GRU is a non-hierarchical seq2seq baseline model. In Table 3 we compare our method with the results of state-of-the-art neural summarization methods reported in recent papers. Extractive models include NN-SE (Cheng and Lapata, 2016) and SummaRuNNer (Nallapati et al., 2017), while SummaRuNNer-abs is also an extractive model similar to SummaRuNNer but is trained directly on the abstractive summaries. Moreover, we include several baselines for comparison, including the baselines reported in (Cheng and Lapata, 2016) although they are tested on 500 samples of the test set. LREG is a feature based method using linear regression. NN-ABS is a neural abstractive baseline which is a simple hierarchical extension of (Rush et al., 2015). NN-WE is the abstractive model which restricts the generation of words from the original document. Lead-3 is a strong extractive baseline that uses the lead three sentences as the summary.

In Table 4 we compare our model with the abstractive attentional encoder-decoder models in

[2]https://github.com/Theano/Theano
[3]http://nlp.stanford.edu/projects/glove

Method	Rouge-1	Rouge-2	Rouge-L
Lead-3	26.1	9.6	17.8
Luhn	23.2	7.2	15.5
Edmundson	24.5	8.2	16.7
LSA	21.2	6.2	14.0
LexRank	26.1	9.6	17.7
TextRank	23.3	7.7	15.8
Sum-basic	22.9	5.5	14.8
KL-sum	20.7	5.9	13.7
Uni-GRU	18.4	4.8	14.3
Distraction-M3	27.1	8.2	18.7
Our Method	**30.3**	**9.8**	**20.0**

Table 2: Comparison results on the **CNN test set** using the **full-length F1** variants of Rouge.

Method	Rouge-1	Rouge-2	Rouge-L
LREG(500)	18.5	6.9	10.2
NN-ABS(500)	7.8	1.7	7.1
NN-WE(500)	15.7	6.4	9.8
Lead-3	21.9	7.2	11.6
NN-SE	22.7	8.5	12.5
SummaRuNNer-abs	23.8	9.6	13.3
SummaRuNNer	26.2	10.8	14.4
Our Method	**27.4**	**11.3**	**15.1**

Table 3: Comparison results on the **DailyMail test set** using Rouge **recall** at **75 bytes**.

(Nallapati et al., 2016), which leverage several effective techniques and achieve state-of-the-art performance on sentence abstractive summarization tasks. The words-lvt2k and words-lvt2k-ptr are flat models and words-lvt2k-hieratt is a hierarchical extension.

Results in Table 2 show our abstractive method is able to outperform traditional extractive methods and the distraction-based abstractive model. The results in Tables 3 and 4 show that our method has considerable improvement over neural abstractive baselines, and is able to outperform state-of-the-art neural extractive methods. An interesting observation is the results of the hierarchical model in Table 4 are lower than the flat models, which may demonstrate the difficulty for a traditional attention model to identify the important information in a document.

We also conducted human evaluation on 20 random samples from the DailyMail test set and compared the summaries generated by our method with the outputs of Lead-3, NN-SE (Cheng and Lapata, 2016) and Distraction (Chen et al., 2016). The output summaries of NN-SE are provided by the authors, and the output summaries of Distraction are achieved by running the code provided by the authors on the DailyMail dataset. Three participants were asked to compare the generated summaries with the human summaries, and assess each summary from four independent perspectives: (1) How informative the summary is? (2) How concise the summary is? (3) How coherent (between sentences) the summary is? (4) How fluent, grammatical the sentences of a summary are? Each property is assessed with a score from 1 (worst) to 5 (best). The average results are presented in Table 5.

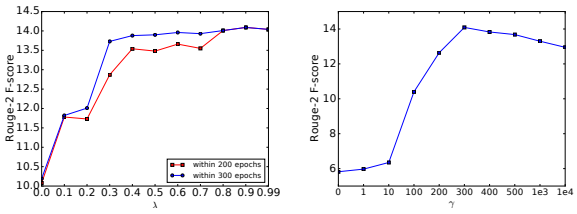
Method	Rouge-1	Rouge-2	Rouge-L
words-lvt2k	32.5	11.8	29.5
words-lvt2k-ptr	32.1	11.7	29.2
words-lvt2k-hieratt	31.8	11.6	28.7
Our Method	**38.1**	**13.9**	**34.0**

Table 4: Comparison results on the **merged CNN/DailyMail test set** using **full-length F1** metric.

Method	Informative	Concise	Coherent	Fluent
Lead-3	3.60	3.75	4.16	3.85
NN-SE	3.85	3.70	3.48	3.78
Distraction	3.03	3.25	2.93	3.65
Our Method	3.93	3.82	3.53	3.80

Table 5: Human evaluation results.

As shown in Table 5, our method consistently outperforms the previous state-of-the-art abstractive method Distraction. Compared with extractive methods, our method is able to generate more informative and concise summaries, which shows the advantage of abstractive methods. The Distraction method in fact usually produces the shortest summaries, but the conciseness score is low mainly because sometimes it generates repeated sentences. The repetition also causes Distraction to achieve a low coherence score. Concerning coherence and fluency, our abstractive method achieves slightly better scores than NN-SE, while not surprisingly Lead-3 gets the best scores. The fluency scores show the good ability of the abstractive model to generate fluent and grammatical sentences.



(a) Rouge-2 F1 score vs. λ. (b) Rouge-2 F1 score vs. γ.

Figure 2: Results of different setting of hyperparameters tested on 500 samples from the DailyMail test set.

Framework	Rouge-1	Rouge-2	Rouge-L
Our Method	**30.3**	**9.8**	**20.0**
w/o GraphAtt	29.2	9.0	19.0
w/o SentenceBeam	29.6	9.3	19.1
w/o BeamSearch	25.1	6.7	17.9

Table 6: Results of removing different components of our method on the **CNN test set** using the **full-length F1** variants of Rouge. Two-tailed t-tests demonstrate the difference between **Our Method** and other frameworks are all statistically significant ($p < 0.01$).

4.4 Model Validation

We conduct experiments to see how the model's performance is affected by the choice of the hyperparameters. For efficiency we test on 500 random samples from the DailyMail test set. Figure 2a shows the maximum average Rouge-2 F1-score achieved when the model is trained using different λ values within 200 and 300 epochs. When using a larger λ, the performance is better and the convergence is faster. When $\lambda = 1.0$ the model fails to train because of running into a singular matrix.

Figure 2b shows the results achieved when using different γ values in the hierarchical decoding algorithm. $\gamma = 0$ is the baseline of the traditional decoding algorithm which does not refer to the original document. The poor results indicate that even the model is able to learn to identify the salient information in the original document, the performance is limited by the model's ability of generating a long output sequence. That may be a reason why simple extensions of seq2seq models fail on the abstractive document summarization task. The performance is significantly improved using a reasonable γ, and the optimal γ value is consistent with the one chosen on the validation set. When using an extremely large γ, the permanence begins to decrease, because the model will copy too much from the original document, and at this time the generated text also becomes less fluent. Results show that introducing the reference mechanism in the hierarchical beam search is very effective. The γ factor significantly affects the results, but the optimal value is easy to be decided on a validation set.

We also conduct ablation experiments on the CNN dataset to verify the effectiveness of the proposed model. Results on the CNN test set are shown in Table 6. "*w/o GraphAtt*" is to replace the graph-based attention by a traditional attention function. "*w/o SentenceBeam*" is to remove the sentence-level beam search. "*w/o BeamSearch*" is to remove both the sentence-level and word-level beam search, and use a greedy decoding algorithm with the reference mechanism. As seen from Table 6, the graph-based attention mechanism is significantly better than traditional attention mechanism for the document summarization task. Beam search helps significantly improve the generated summaries. Our proposed decoding algorithm enables a sentence-level beam search, which helps improve the generated summaries with multiple sentences.

4.5 Case Study

We show the case study of a sample[4] from the DailyMail test set in Figure 3. We show the "@entity" and number here although they are removed in the evaluation. We compare our result with the output by a model using traditional attention as *Baseline Attention*. We also show the output generated by a *Baseline Decoder*, which sets $\gamma = 0$ and does not use the sentence-level beam search, to study the difficulty for a traditional decoder to generate multiple sentences. Many observations can be found in Figure 3. The lead three sentences mainly focus on the money information and are not sufficient. As for the *Baseline Decoder*, first it usually ends the generation too early. The "<eod>" token indicates where the original output stops. When we force the decoder not to end here, the model shows the ability to continue producing the important information. However, two flaws are presented. First is the repetition of "## - year -

[4]The original story and highlights can be found at http://www.dailymail.co.uk/news/article-3041766/Benefits-cheat-pocketed-17-000-taxpayers-money.html

Gold Summary:
@entity 2 mary day , ## , claimed over £ ##,### in benefits despite not being eligible .
she had £ ##,### savings in the bank which meant she was not entitled .
day used taxpayers ' money to go on luxury holidays to @entity 1 indian resort of @entity 1 goa .
pleaded guilty to dishonestly claiming benefits and has paid back money .

Lead3:
a benefits cheat who pocketed almost £ ##,### of taxpayers ' money and spent it on a string of luxury holidays despite having £ ##,### in the bank has avoided jail .
@entity 2 mary day , ## , of @entity 1 swanage in @entity 1 dorset , used taxpayers ' money to go on luxury holidays to the @entity 1 indian resort of @entity 1 goa for up to a month each time .
day fraudulently claimed £ ##,### of income support and disability allowance despite having £ ##,### of her own savings in the bank .

Baseline Decoder:
- month - old @entity 2 mary day , ## , was given £ ##,### in money .
the ## - year - old claimed £ ##,### in disability allowance . <*eod*>
the ## - year - old was given a six - month prison sentence .
- year - old pleaded guilty to two counts of fraud .

Baseline Attention:
@entity 2 mary day , ## , used taxpayers ' money to go on luxury holidays .
claimed £ ##,### of income support and disability allowance despite having savings in the bank . <*eod*>
benefits of taxpayers £ ##,### in disability handouts .

Our Method:
@entity 2 mary day , ## , used taxpayers ' money to go on luxury holidays to the @entity 1 indian resort of @entity 1 goa .
despite having £ ##,### of her own savings in the bank , she claimed £ ##,### of income support and disability allowance .
she pleaded guilty and had given the sentence for three months in prison , but suspended the sentence for ## months .

Figure 3: Examples of generated summaries.

old". Because the word decoder is unaware of the history generated sentences, it repeats generating the sequence as the subject all the time. Second, more importantly, is the information incorrectness. The "## - month - old" is not appropriate to describe the heroine, and the "six - month prison sentence" is in fact "three months". Information incorrectness occurs because, for a decoder, it aims at generating a fluent sentence according to the input representation. However, no favor of consistent with the original input is concerned. The proposed hierarchical decoding algorithm helps to alleviate the two problems. The awareness of all the generated sentences helps prevent from always generating some important information. The favor of bigram overlapping with the original sentences helps generate more correct sentences. For example the model is able to correctly distinguish between the "three-month sentence" and the "##-month suspend". In conclusion, our method is able to identify the most important information in the original document, and the decoding algorithm we propose is able to generate a more discourse-fluent and information-correct abstractive summary.

The visualization of the graph-based attention when our method generates the presented example

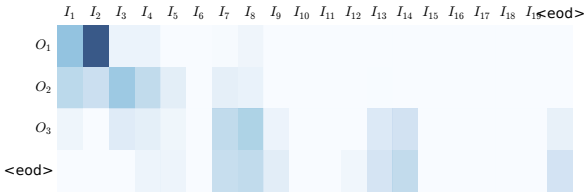


Figure 4: Attention heatmap when generating the example summary. I_i and O_i indicate the i-th sentence of the input and output, respectively.

is shown in Figure 4. It seems that the graph-based attention mechanism is able to find the important sentences in the input document, and the distraction mechanism makes the decoder focus on different sentences during decoding. Gradually the decoder attends to "<eod>" until it stops.

5 Conclusion and Future Work

In this paper we tackle the challenging task of abstractive document summarization, which is still less investigated to date. We study the difficulty of the abstractive document summarization task, and address the need of finding salient content from the original document, which is overlooked by previous studies. We propose a novel graph-based attention mechanism in a hierarchical encoder-decoder framework, and propose a hierarchical beam search algorithm to generate multi-sentence summary. Extensive experiments verify the effectiveness of the proposed method. Experimental results on two large-scale datasets demonstrate our method achieves state-of-the-art abstractive document summarization performance. It is also able to achieve competitive results with state-of-the-art neural extractive summarization models.

There is lots of future work we can do. An appealing direction is to investigate the neural abstractive method on the multi-document summarization task, which is more challenging and lacks training data. Further endeavor may be needed.

Acknowledgments

This work was supported by 863 Program of China (2015AA015403), NSFC (61331011), and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for helpful comments and Xinjie Zhou, Jianmin Zhang for doing human evaluation. Xiaojun Wan is the corresponding author.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015a. Ranking with recursive neural networks and its application to multi-document summarization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*. pages 2153–2159.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. 2015b. Learning summary prior representation for extractive summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 829–833. https://doi.org/10.3115/v1/P15-2136.

Jaime G. Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*. pages 335–336.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*. pages 2754–2760.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 484–494. https://doi.org/10.18653/v1/P16-1046.

Kit Jackie Chi Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 775–786. https://doi.org/10.3115/v1/D14-1085.

Sumit Chopra, Michael Auli, and M. Alexander Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 93–98. https://doi.org/10.18653/v1/N16-1012.

Anirban Dasgupta, Ravi Kumar, and Sujith Ravi. 2013. Summarization through submodularity and dispersion. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1014–1022. http://aclweb.org/anthology/P13-1100.

Harold P Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM (JACM)* 16(2):264–285.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)* 22:457–479.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the Eleventh International World Wide Web Conference, WWW 2002, May 7-11, 2002, Honolulu, Hawaii*. pages 517–526.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 1693–1701.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Hayato Kobayashi, Masaki Noguchi, and Taichi Yatsuka. 2015. Summarization based on embedding distributions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1984–1989. https://doi.org/10.18653/v1/D15-1232.

Chen Li, Xian Qian, and Yang Liu. 2013. Using supervised bigram-based ilp for extractive summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1004–1013. http://aclweb.org/anthology/P13-1099.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1106–1115. https://doi.org/10.3115/v1/P15-1107.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, volume 8.

Ryan T. McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Advances in Information Retrieval, 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007, Proceedings*. pages 557–564.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. pages 404–411.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 3349–3358. http://aclweb.org/anthology/C16-1316.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*. pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 280–290. https://doi.org/10.18653/v1/K16-1028.

Hitoshi Nishikawa, Kazuho Arita, Katsumi Tanaka, Tsutomu Hirao, Toshiro Makino, and Yoshihiro Matsuo. 2014. Learning to generate coherent summary with discriminative hidden semi-markov model. In *Proceedings of COLING 2014*. Dublin City University and Association for Computational Linguistics, pages 1648–1659.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1532–1543. https://doi.org/10.3115/v1/D14-1162.

Daniele Pighin, Marco Cornolti, Enrique Alfonseca, and Katja Filippova. 2014. Modelling events through memory-based, open-ie patterns for abstractive summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 892–901. https://doi.org/10.3115/v1/P14-1084.

M. Alexander Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 379–389. https://doi.org/10.18653/v1/D15-1044.

Rui Sun, Yue Zhang, Meishan Zhang, and Donghong Ji. 2015. Event-driven headline generation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 462–472. https://doi.org/10.3115/v1/P15-1045.

Kapil Thadani and Kathleen McKeown. 2013. Supervised sentence fusion with single-stage inference. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, pages 1410–1418.

Xiaojun Wan. 2010. Towards a unified approach to simultaneous single-document and multi-document summarizations. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, pages 1137–1145.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Manifold-ranking based topic-focused multi-document summarization. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*. pages 2903–2908.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 233–243.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2017. Recent advances in document summarization. *Knowledge and Information Systems* .

Wenpeng Yin and Yulong Pei. 2015. Optimizing sentence modeling and selection for document summarization. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. pages 1383–1389.

Probabilistic Typology: Deep Generative Models of Vowel Inventories

Ryan Cotterell and **Jason Eisner**
Department of Computer Science
Johns Hopkins University
{ryan.cotterell,eisner}@jhu.edu

Abstract

Linguistic typology studies the range of structures present in human language. The main goal of the field is to discover which sets of possible phenomena are universal, and which are merely frequent. For example, all languages have vowels, while most—but not all—languages have an [u] sound. In this paper we present the first probabilistic treatment of a basic question in phonological typology: What makes a natural vowel inventory? We introduce a series of deep stochastic point processes, and contrast them with previous computational, simulation-based approaches. We provide a comprehensive suite of experiments on over 200 distinct languages.

1 Introduction

Human languages exhibit a wide range of phenomena, within some limits. However, some structures seem to occur or co-occur more frequently than others. Linguistic typology attempts to describe the range of natural variation and seeks to organize and quantify linguistic universals, such as patterns of co-occurrence. Perhaps one of the simplest typological questions comes from phonology: which vowels tend to occur and co-occur within the phoneme inventories of different languages? Drawing inspiration from the linguistic literature, we propose models of the probability distribution from which the attested vowel inventories have been drawn.

It is a typological universal that every language contains both vowels and consonants (Velupillai, 2012). But which vowels a language contains is guided by softer constraints, in that certain configurations are more widely attested than others. For instance, in a typical phoneme inventory, there tend to be far fewer vowels than consonants. Likewise, all languages contrast vowels based on height, although which contrast is made is language-dependent (Ladefoged and Maddieson, 1996). Moreover, while over 600 unique vowel phonemes have been attested cross-linguistically (Moran et al., 2014), certain regions of acoustic space are used much more often than others, e.g., the regions conventionally transcribed as [a], [i], and [u]. Human language also seems to prefer inventories where phonologically distinct vowels are spread out in acoustic space ("dispersion") so that they can be easily distinguished by a listener. We depict the acoustic space for English in Figure 2.

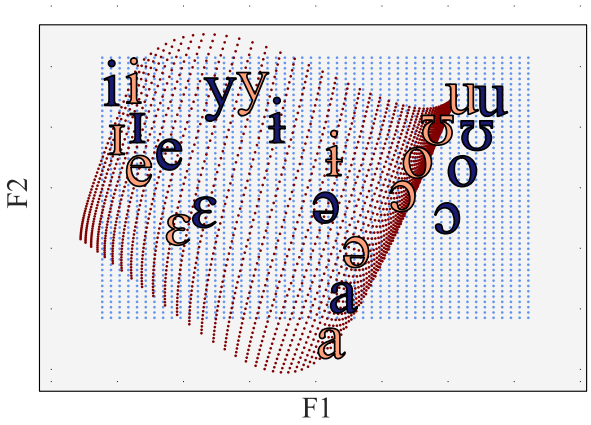


Figure 1: The transformed vowel space that is constructed within one of our deep generative models (see §7.1). A deep network nonlinearly maps the blue grid ("formant space") to the red grid ("metric space"), with individual vowels mapped from blue to red position as shown. Vowel pairs such as [ə]–[ɔ] that are brought close together are anti-correlated in the point process. Other pairs such as [y]–[i] are driven apart. For purposes of the visualization, we have transformed the red coordinate system to place red vowels near their blue positions—while preserving distances up to a constant factor (a "Procrustes transformation").

In this work, we regard the proper goal of linguistic typology as the construction of a universal prior distribution from which linguistic systems are drawn. For vowel system typology, we propose three formal probability models based on stochastic point processes. We estimate the parameters of the model on one set of languages and evaluate performance on a held-out set. We explore three questions: (i) How well do the properties of our proposed probability models line up experimentally with linguistic theory? (ii) How well can our models predict held-out vowel systems? (iii) Do our models benefit from a "deep" transformation from formant space to metric space?

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1182–1192
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1109

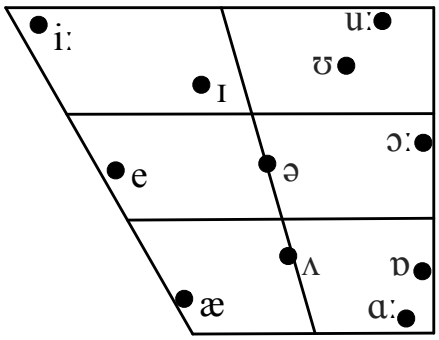


Figure 2: The standard vowel table in IPA for the RP accent of English. The x-axis indicates the front-back spectrum and the y-axis indicates the high-low distinction.

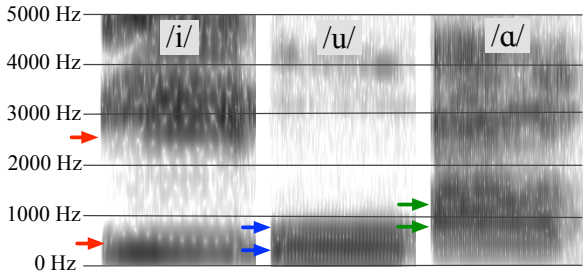


Figure 3: Example spectrogram of the three English vowels: [i], [u] and [ɑ]. The x-axis is time and y-axis is frequency. The first two formants F_1 and F_2 are marked in with colored arrows for each vowel. We used the Praat toolkit to generate the spectrogram and find the formants (Boersma et al., 2002).

2 Vowel Inventories and their Typology

Vowel inventories are a simple entry point into the study of linguistic typology. Every spoken language chooses a discrete set of vowels, and the number of vowel phonemes ranges from 3 to 46, with a mean of 8.7 (Gordon, 2016). Nevertheless, the empirical distribution over vowel inventories is remarkably peaked. The majority of languages have 5–7 vowels, and there are only a handful of distinct 4-vowel systems attested despite many possibilities. Reigning linguistic theory (Becker-Kristal, 2010) has proposed that vowel inventories are shaped by the principles discussed below.

2.1 Acoustic Phonetics

One way to describe the sound of a vowel is through its acoustic energy at different frequencies. A spectrogram (Figure 3) is a visualization of the energy at various frequencies over time. Consider the "peak" frequencies $F_0 < F_1 < F_2 < \ldots$ that have a greater energy than their neighboring frequencies. F_0 is called the fundamental frequency or pitch. The other qualities of the vowel are largely determined by $F_1, F_2, \ldots$, which are known as formants (Ladefoged and Johnson, 2014). In many languages, the first two formants F_1 and F_2 contain enough information to identify a vowel: Figure 3 shows how these differ across three English vowels. We consider each vowel listed in the International Phonetic Alphabet (IPA) to be *cross-linguistically* characterized by some (F_1, F_2) pair.

2.2 Dispersion

The dispersion criterion (Liljencrants and Lindblom, 1972; Lindblom, 1986) states that the phonemes of a language must be "spread out" so that they are easily discriminated by a listener. A language seeks phonemes that are sufficiently "distant" from one another to avoid confusion. Distances between phonemes are defined in some latent "metric space." We use this term rather than "perceptual space" because the confusability of two vowels may reflect not just their perceptual similarity, but also their common distortions by imprecise articulation or background noise.[1]

2.3 Focalization

The dispersion criterion alone does not seem to capture the whole story. Certain vowels are simply more popular cross-linguistically. A commonly accepted explanation is the *quantal theory of speech* (Stevens, 1972, 1989). The quantal theory states that certain sounds are easier to articulate and to perceive than others. These vowels may be characterized as those where F_1 and F_2 have frequencies that are close to one another. On the production side, these vowels are easier to pronounce since they allow for greater articulatory imprecision. On the perception side, they are more salient since the two spectral peaks aggregate and act as one, larger peak to a certain degree. In general, languages will prefer these vowels.

2.4 Dispersion-Focalization Theory

The dispersion-focalization theory (DFT) combines both of the above notions. A good vowel system now consists of vowels that contrast with each other *and* are individually desirable (Schwartz et al., 1997). This paper provides the first probabilistic treatment of DFT, and new evaluation metrics for future probabilistic and non-probabilistic treatments of vowel inventory typology.

[1] We assume in this paper that the metric space is universal—although it would not be unreasonable to suppose that each language's vowel system has adapted to avoid confusion in the *specific* communicative environment of its speakers.

3 Point Process Models

Given a base set $\mathcal{V}$, a point process is a distribution over its subsets.[2] In this paper, we take $\mathcal{V}$ to be the set of all IPA symbols corresponding to vowels. Thus a draw from a point process is a vowel inventory $V \subseteq \mathcal{V}$, and the point process itself is a distribution over such inventories. We will consider three basic point process models for vowel systems: the Bernoulli Point Process, the Markov Point Process and the Determinantal Point Process. In this section, we review the relevant theory of point processes, highlighting aspects related to §2.

3.1 Bernoulli Point Processes

Taking $\mathcal{V} = \{v_1, \ldots, v_N\}$, a Bernoulli point process (BPP) makes an *independent* decision about whether to include each vowel in the subset. The probability of a vowel system $V \subseteq \mathcal{V}$ is thus

$$p(V) \propto \prod_{v_i \in V} \phi(v_i), \tag{1}$$

where ϕ is a unary potential function, i.e., $\phi(v_i) \geq 0$. Qualitatively, this means that $\phi(v_i)$ should be large if the i^{th} vowel is good in the sense of §2.3. Marginal inference in a BPP is computationally trivial. The probability that the inventory V contains v_i is $\phi(v_i)/(1 + \phi(v_i))$, independent of the other vowels in V. Since a BPP predicts each vowel independently, it only models focalization. Thus, the model provides an appropriate baseline that will let us measure the importance of the dispersion principle—how far can we get with just focalization? A BPP may still tend to generate well-dispersed sets if it defines ϕ to be large only on certain vowels in $\mathcal{V}$ and these are well-dispersed (e.g., [i], [u], [a]). More precisely, it can define ϕ so that $\phi(v_i)\phi(v_j)$ is small whenever v_i, v_j are similar.[3] But it cannot actively encourage dispersion: including v_i does not lower the probability of also including v_j.

3.2 Markov Point Processes

A Markov Point Process (MPP) (Van Lieshout, 2000)—also known as a Boltzmann machine (Ackley et al., 1985; Hinton and Sejnowski, 1986)—generalizes the BPP by adding pairwise interactions between vowels. The probability of a vowel system $V \subseteq \mathcal{V}$ is now

$$p(V) \propto \prod_{v_i \in V} \phi(v_i) \prod_{v_i, v_j \in V} \psi(v_i, v_j), \tag{2}$$

where each $\phi(v_i) \geq 0$ is, again, a unary potential that scores the quality of the i^{th} vowel, and each $\psi(v_i, v_j) \geq 0$ is a binary potential that scores the combination of the i^{th} and j^{th} vowels. Roughly speaking, the potential $\psi(v_i, v_j)$ should be large if the i^{th} and j^{th} vowel often co-occur. Recall that under the principle of dispersion, the vowels that often co-occur are easily distinguishable. Thus, confusable vowel pairs should tend to have potential $\psi(v_i, v_j) < 1$.

Unlike the BPP, the MPP can capture both focalization *and* dispersion. In this work, we will consider a fully connected MPP, i.e., there is a potential function for each pair of vowels in $\mathcal{V}$. MPPs closely resemble Ising models (Ising, 1925), but with the difference that Ising models are typically lattice-structured, rather than fully connected.

Inference in MPPs. Inference in fully connected MPPs, just as in general Markov Random Fields (MRFs), is intractable (Cooper, 1990) and we must rely on approximation. In this work, we estimate any needed properties of the MPP distribution by (approximately) drawing vowel inventories from it via Gibbs sampling (Geman and Geman, 1984; Robert and Casella, 2005). Gibbs sampling simulates a discrete-time Markov chain whose stationary distribution is the desired MPP distribution. At each time step, for some random $v_i \in \mathcal{V}$, it stochastically decides whether to replace the current inventory V with $\bar{V}$, where $\bar{V}$ is a copy of V with v_i added (if $v_i \notin V$) or removed (if $v_i \in V$). The probability of replacement is $\frac{p(\bar{V})}{p(V)+p(\bar{V})}$.

3.3 Determinantal Point Processes

A determinantal point process (DPP) (Macchi, 1975) provides an elegant alternative to an MPP, and one that is directly suited to modeling both focalization and dispersion. Inference requires only

[2] A point process is a specific kind of stochastic process, which is the technical term for a distribution over *functions*. Under this view, drawing some subset of $\mathcal{V}$ from the point process is regarded as drawing some *indicator function* on $\mathcal{V}$.

[3] We point out that such a scheme would break down if we extended our work to cover fine-grained phonetic modeling of the vowel inventory. In that setting, we ask not just whether the inventory includes /i/ but exactly which pronunciation of /i/ it contains. In the limit, ϕ becomes a function over a *continuous* vowel space $\mathcal{V} = \mathbb{R}^2$, turning the BPP into an inhomogeneous spatial Poisson process. A continuous ϕ function implies that the model places similar probability on similar vowels. Then if most vowel inventories contain some version of /i/, then many of them will contain several closely related variants of /i/ (independently chosen). By contrast, the other methods in this paper do extend nicely to fine-grained phonetic modeling.

a few matrix computations and runs tractably in $O(|\mathcal{V}|^3)$ time, even though the model may encode a rich set of multi-way interactions. We focus on the L-ensemble parameterization of the DPP, due to Borodin and Rains (2005).[4] This type of DPP defines the probability of an inventory $V \subseteq \mathcal{V}$ as

$$p(V) \propto \det L_V, \tag{3}$$

where $L \in \mathbb{R}^{N \times N}$ (for $N = |\mathcal{V}|$) is a symmetric positive semidefinite matrix, and L_V refers to the submatrix of L with only those rows and columns corresponding to those elements in the subset V.

Although MAP inference remains NP-hard in DPPs (just as in MPPs), marginal inference becomes tractable. We may compute the normalizing constant in closed form as follows:

$$\sum_{V \in 2^{\mathcal{V}}} \det L_V = \det (L + I). \tag{4}$$

How does a DPP ensure focalization and dispersion? L is positive semidefinite iff it can be written as $E^\top E$ for some matrix $E \in \mathbb{R}^{N \times N}$. It is possible to express $p(V)$ in terms of the column vectors of E, which we call $\mathbf{e}_1, \ldots, \mathbf{e}_N$:

- For inventories of size 2, $p(\{v_i, v_j\}) \propto (\phi(v_i)\phi(v_j)\sin\theta)^2$, where $\phi(v_i), \phi(v_j)$ represent the *quality* of vowels v_i, v_j (as in the BPP) while $\sin\theta \in [0, 1]$ represents their *dissimilarity*. More precisely, $\phi(v_i), \phi(v_j)$ are the lengths of vectors $\mathbf{e}_i, \mathbf{e}_j$ while θ is the angle between them. Thus, we should choose the columns of E so that focal vowels get *long* vectors and similar vowels get vectors *of similar direction*.
- Generalizing beyond inventories of size 2, $p(V)$ is proportional to the square of the volume of the parallelepiped whose sides are given by $\{\mathbf{e}_i : v_i \in V\}$. This volume can be regarded as $\prod_{v_i \in V} \phi(v_i)$ times a term that ranges from 1 for an orthogonal set of vowels to 0 for a linearly dependent set of vowels.
- The events $v_i \in V$ and $v_j \in V$ are anti-correlated (when not independent). That is, while both vowels may individually have high probabilities (focalization), having either one in the inventory lowers the probability of the other (dispersion).

[4] Most DPPs are L-ensembles (Kulesza and Taskar, 2012).

4 Dataset

At this point it is helpful to introduce the empirical dataset we will model. For each of 223 languages,[5] Becker-Kristal (2010) provides the vowel inventory as a set of IPA symbols, listing the first 5 formants for each vowel (or fewer when not available in the original source). Some corpus statistics are shown in Figs. 4 and 5.[6] For the present paper, we take $\mathcal{V}$ to be the set of all 53 IPA symbols that appear in the corpus. We treat these IPA labels as meaningful, in that we consider two vowels in different languages to be the *same* vowel in $\mathcal{V}$ if (for example) they are both annotated as [ɔ]. We characterize that vowel by its *average* formant vector across all languages in the corpus that contain the vowel: e.g., $(F_1, F_2, \ldots) = (500, 700, \ldots)$ for [ɔ]. In future work, we plan to relax this idealization (see footnote 3), allowing us to investigate natural questions such as whether [u] is pronounced higher (smaller F_1) in languages that also contain [o] (to achieve better dispersion).

5 Model Parameterization

The BPP, MPP, and DPP models (§3) require us to specify parameters for each vowel in $\mathcal{V}$. In §5.1, we will accomplish this by deriving the parameters for each vowel v_i from a possibly high-dimensional embedding of that vowel, $\boldsymbol{e}(v_i) \in \mathbb{R}^r$.

In §5.2, $\boldsymbol{e}(v_i) \in \mathbb{R}^r$ will in turn be defined as some learned function of $\boldsymbol{f}(v_i) \in \mathbb{R}^k$, where $\boldsymbol{f} : \mathcal{V} \mapsto \mathbb{R}^k$ is the function that maps a vowel to a k-vector of its measurable acoustic properties. This approach allows us to determine reasonable parameters even for rare vowels, based on their measurable properties. It will even enable us in

[5] Becker-Kristal lists some languages multiple times with different measurements. When a language had multiple listings, we selected one randomly for our experiments.

[6] Caveat: The corpus is a curation of information from various phonetics papers into a common electronic format. No standard procedure was followed across all languages: it was up to individual phoneticists to determine the size of each vowel inventory, the choice of IPA symbols to describe it, and the procedure for measuring the formants. Moreover, it is an idealization to provide a single vector of formants for each vowel *type* in the language. In real speech, different *tokens* of the same vowel are pronounced differently, because of coarticulation with the vowel context, allophony, interspeaker variation, and stochastic intraspeaker variation. Even within a token, the formants change during the duration of the vowel. Thus, one might do better to represent a vowel's pronunciation not by a formant vector, but by a conditional probability distribution over its formant trajectories given its context, or by a parameter vector that characterizes such a conditional distribution. This setting would require richer data than we present here.

future to generalize to vowels that were unseen in the training set, letting us scale to very large or infinite $\mathcal{V}$ (footnote 3).

5.1 Deep Point Processes

We consider deep versions of all three processes.

Deep Bernoulli Point Process. We define

$$\phi(v_i) = ||\boldsymbol{e}(v_i)|| \geq 0 \tag{5}$$

Deep Markov Point Process. The MPP employs the same unary potential as the BPP, as well as the binary potential

$$\psi(v_i, v_j) = \exp -\frac{1}{T \cdot ||\boldsymbol{e}(v_i) - \boldsymbol{e}(v_j)||^2} < 1 \tag{6}$$

where the learned temperature $T > 0$ controls the relative strength of the unary and binary potentials.

This formula is inspired by Coulomb's law for describing the repulsion of static electrically charged particles. Just as the repulsive force between two particles approaches ∞ as they approach each other, the probability of finding two vowels in the same inventory approaches $\exp -\infty = 0$ as they approach each other. The formula is also reminiscent of Shepard (1987)'s "universal law of generalization," which says here that the probability of responding to v_i as if it were v_j should fall off exponentially with their distance in some "psychological space" (here, embedding space).

Deep Determinantal Point Process. For the DPP, we simply define the vector $\mathbf{e}_i$ to be $\boldsymbol{e}(v_i)$, and proceed as before.

Summary. In the deep BPP, the probability of a set of vowels is proportional to the product of the lengths of their embedding vectors. The deep MPP modifies this by multiplying in *pairwise* repulsion terms in $(0, 1)$ that increase as the vectors' endpoints move apart in Euclidean space (or as $T \to \infty$). The deep DPP instead modifies it by multiplying in a single *setwise* repulsion term in $(0, 1)$ that increases as the embedding vectors become more mutually orthogonal. In the limit, then, the MPP and DPP both approach the BPP.

5.2 Embeddings

Throughout this work, we simply have $\boldsymbol{f}$ extract the first $k = 2$ formants, since our dataset does not provide higher formants for all languages.[7] For example, we have $\boldsymbol{f}([\text{ɔ}]) = (500, 700)$. We now describe three possible methods for mapping $\boldsymbol{f}(v_i)$ to an embedding $\boldsymbol{e}(v_i)$. Each of these maps has learnable parameters.

Neural Embedding. We first consider directly embedding each vowel v_i into a vector space $\mathbb{R}^r$. We achieve this through a feed-forward neural net

$$\boldsymbol{e}(v_i) = W_1 \tanh\left(W_0 \boldsymbol{f}(v_i) + \mathbf{b}_0\right) + \mathbf{b}_1, \tag{7}$$

Equation (7) gives an architecture with 1 layer of nonlinearity; in general we consider stacking $d \geq 0$ layers. Here $W_0 \in \mathbb{R}^{r \times k}, W_1 \in \mathbb{R}^{r \times r}, \ldots W_d \in \mathbb{R}^{r \times r}$ are weight matrices, $\mathbf{b}_0, \ldots \mathbf{b}_d \in \mathbb{R}^r$ are bias vectors, and $\tanh$ could be replaced by any pointwise nonlinearity. We treat both the depth d and the embedding size r as hyperparameters, and select the optimal values on a development set.

Interpretable Neural Embedding. We are interested in the special case of neural embeddings when $r = k$ since then (for any d) the mapping $\boldsymbol{f}(v_i) \mapsto \boldsymbol{e}(v_i)$ is a diffeomorphism:[8] a smooth invertible function of $\mathbb{R}^k$. An example of such a diffeomorphism is shown in Figure 1.

There is a long history in cognitive psychology of mapping stimuli into some psychological space. The distances in this psychological space may be predictive of generalization (Shepard, 1987) or of perception. Due to the anatomy of the ear, the mapping of vowels from acoustic space to perceptual space is often presumed to be nonlinear (Rosner and Pickering, 1994; Nearey and Kiefte, 2003), and there are many perceptually-oriented phonetic scales, e.g., Bark and Mel, that carry out such nonlinear transformations while preserving the dimensionality k, as we do here. As discussed in §2.2, vowel system typology is similarly believed to be influenced by distances between the vowels in a latent metric space. We are interested in whether a constrained k-dimensional model of these distances can do well in our experiments.

Prototype-Based Embedding. Unfortunately, our interpretable neural embedding is unfortunately incompatible with the DPP. The DPP assigns probability 0 to any vowel inventory V whose $\boldsymbol{e}$ vectors are linearly dependent. If the vectors are in $\mathbb{R}^k$, then this means that $p(V) = 0$ whenever $|V| > k$. In our setting, this would limit vowel inventories to size 2.

[7] In lieu of higher formants, we could have extended the vector $\boldsymbol{f}(v_i)$ to encode the binary distinctive features of the IPA vowel v_i: round, tense, long, nasal, creaky, etc.

[8] Provided that our nonlinearity in (7) is a differentiable invertible function like $\tanh$ rather than relu.

Our solution to this problem is to still construct our interpretable metric space $\mathbb{R}^k$, but then map that nonlinearly to $\mathbb{R}^r$ for some large r. This latter map is constrained. Specifically, we choose "prototype" points $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_r \in \mathbb{R}^k$. These prototype points are parameters of the model: their coordinates are learned and do not necessarily correspond to any actual vowel. We then construct $\boldsymbol{e}(v_i) \in \mathbb{R}^r$ as a "response vector" of similarities of our vowel v_i to these prototypes. Crucially, the responses depend on distances measured in the interpretable metric space $\mathbb{R}^k$. We use a Gaussian-density response function, where $\mathbf{x}(v_i)$ denotes the representation of our vowel v_i in the interpretable space:

$$\begin{aligned} \boldsymbol{e}(v_i)_\ell &= w_\ell \; p(\mathbf{x}(v_i); \boldsymbol{\mu}_\ell, \sigma^2 I) \\ &= w_\ell \; (2\pi\sigma^2)^{-\left(\frac{k}{2}\right)} \exp\left(\frac{-||\mathbf{x} - \boldsymbol{\mu}_\ell||^2}{2\sigma^2}\right). \end{aligned} \tag{8}$$

for $\ell = 1, 2, \ldots, r$. We additionally impose the constraints that each $w_\ell \geq 0$ and $\sum_{\ell=1}^{r} w_\ell = 1$.

Notice that the sum $\sum_{\ell=1}^{r} \boldsymbol{e}(v_i)$ may be viewed as the density at $\mathbf{x}(v_i)$ under a Gaussian mixture model. We use this fact to construct a prototype-based MPP as well: we redefine $\phi(v_i)$ to equal this positive density, while still defining ψ via equation (6). The idea is that dispersion is measured in the interpretable space $\mathbb{R}^k$, and focalization is defined by certain "good" regions in that space that are centered at the r prototypes.

6 Evaluation Metrics

Fundamentally, we are interested in whether our model has abstracted the core principles of what makes a *good* vowel system. Our choice of a probabilistic model provides a natural test: how surprised is our model by held-out languages? In other words, how likely does our model think unobserved, but attested vowel systems are? While this is a natural evaluation paradigm in NLP, it has not—to the best of our knowledge—been applied to a quantitative investigation of linguistic typology.

As a second evaluation, we introduce a *vowel system cloze task* that could also be used to evaluate non-probabilistic models. This task is defined by analogy to the traditional semantic cloze task (Taylor, 1953), where the reader is asked to fill in a missing word in the sentence from the context. In our vowel system cloze task, we present a learner with a subset of the vowels in a held-out vowel system and ask them to predict the remaining vowels. Consider, as a concrete example, the general American English vowel system (excluding long vowels) {[i], [ɪ], [u], [ʊ], [ɛ], [æ], [ɔ], [ɑ], [ə]}. One potential cloze task would be to predict {[i], [u]} given {[ɪ], [ʊ], [ɛ], [æ], [ɔ], [ɑ], [ə]} and the fact that two vowels are missing from the inventory. Within the cloze task, we report accuracy, i.e., did we guess the missing vowel right? We consider three versions of the cloze tasks. First, we predict *one* missing vowel in a setting where exactly one vowel was deleted. Second, we predict *up to one* missing vowel where a vowel *may* have been deleted. Third, we predict *up to two* missing vowels, where one or two vowels may be deleted.

7 Experiments

We evaluate our models using 10-fold cross-validation over the 223 languages. We report the mean performance over the 10 folds. The performance on each fold ("test") was obtained by training many models on 8 of the other 9 folds ("train"), selecting the model that obtained the best task-specific performance on the remaining fold ("development"), and assessing it on the test fold. Minimization of the parameters is performed with the L-BFGS algorithm (Liu and Nocedal, 1989). As a preprocessing step, the first two formants values F_1 and F_2 are centered around zero and scaled down by a factor of 1000 since the formant values themselves may be quite large.

Specifically, we use the development fold to select among the following combinations of hyperparameters. For neural embeddings, we tried $r \in \{2, 10, 50, 100, 150, 200\}$. For prototype embeddings, we took the number of components $r \in \{20, 30, 40, 50\}$. We tried network depths $d \in \{0, 1, 2, 3\}$. We sweep the coefficient for an L_2 regularizer on the neural network parameters.

7.1 Results and Discussion

Figure 1 visualizes the diffeomorphism from formant space to metric space for one of our DPP models (depth $d = 3$ with $r = 20$ prototypes). Similar figures can be generated for all of the interpretable models.

We report results for cross-entropy and the cloze evaluation in Table 1.[9] Under both metrics, we see that the DPP is slightly better than the MPP; both are better than the BPP. This ranking holds for

[9] Computing cross-entropy exactly is intractable with the MPP, so we resort to an unbiased importance sampling scheme where we draw samples from the BPP and reweight according to the MPP (Liu et al., 2015).

	BPP	uBPP	uMPP	uDPP	iBPP	iMPP	iDPP	pBPP	pMPP	pDPP
x-ent	8.24	8.28	8.08	8.00	13.01	11.50	✗	12.83	10.95	10.29
cloze-1	69.55%	69.55%	72.05%	73.18%	64.13%	67.02%	✗	65.13%	68.18%	68.18%
cloze-01	60.00%	60.00%	61.01%	62.27%	61.78%	61.04%	✗	61.02%	63.04%	63.63%
cloze-012	53.18%	53.18%	57.92%	58.18%	39.04%	43.02%	✗	40.56%	45.01%	45.46%

Table 1: Cross-entropy in nats (lower is better) and cloze prediction accuracy (higher is better). "BPP" is a simple BPP with one parameter for each of the 53 vowels in $\mathcal{V}$. This model does artificially well by modeling an "accidental" feature of our data: it is able to learn not only which vowels are popular among languages, but also which IPA symbols are popular or conventional among the descriptive phoneticists who created our dataset (see footnote 6), something that would become irrelevant if we upgraded our task to predict actual formant vectors rather than IPA symbols (see footnote 3). Our point processes, by contrast, are appropriately allowed to consider a vowel only through its formant vector. The "*u*-" versions of the models use the uninterpretable neural embedding of the formant vector into $\mathbb{R}^r$: by taking r to be large, they are still able to learn special treatment for each vowel in $\mathcal{V}$ (which is why uBPP performs identically to BPP, before being beaten by uMPP and uDPP). The "*i*-" versions limit themselves to an interpretable neural embedding into $\mathbb{R}^k$, giving a more realistic description that does not perform as well. The "*p*-"versions lift that $\mathbb{R}^k$ embedding into $\mathbb{R}^r$ by measuring similarities to r prototypes; they thereby improve on the corresponding *i*- versions. For each result shown, the depth d of our neural network was tuned on a development set (typically $d = 2$). r was also tuned when applicable (typically $r > 100$ dimensions for the *u*- models and $r \approx 30$ prototypes for the *p*- models).

each of the 3 embedding schemes. The embedding schemes themselves are compared in the caption.

Within each embedding scheme, the BPP performs several points worse on the cloze tasks, confirming that dispersion is needed to model vowel inventories well. Still, the BPP's respectable performance shows that much of the structure can be capture by focalization. As §3 noted, the BPP may generate well-dispersed sets, as the common vowels tend to be dispersed already (see Figure 4). In this capacity, however, the BPP is not explanatory as it cannot actually tell us why *these* vowels should be frequent.

We mention that depth in the neural network is helpful, with deeper embedding networks performing slightly better than depth $d = 0$.

Finally, we identified each model's favorite complete vowel system of size n (Table 2). For the BPP, this is simply the n most probable vowels. Decoding the DPP and MPP is NP-hard, but we found the best system by brute force (for small n). The dispersion in these models predicts different systems than the BPP.

8 Discussion: Probabilistic Typology

Typology as Density Estimation? Our goal is to define a *universal* distribution over all possible vowel inventories. Is this appropriate? We regard this as a natural approach to typology, because it directly describes which kinds of linguistic systems are more or less common. Traditional implicational universals ("all languages with v_i have v_j") are softened, in our approach, into conditional probabilities such as "$p(v_j \in V \mid v_i \in V) \approx 0.9$." Here the 0.9 is not merely an empirical ratio, but a smoothed probability derived from the complete estimated distribution. It is meant to make predictions about unseen languages.

Whether human language learners exploit any properties of this distribution[10] is a separate question that goes beyond typology. Jakobson (1941) did find that children acquired phoneme inventories in an order that reflected principles similar to dispersion ("maximum contrast") and focalization.

At any rate, we estimate the distribution given some set of attested systems that are assumed to have been drawn IID from it. One might object that this IID assumption ignores evolutionary relationships among the attested systems, causing our estimated distribution to favor systems that are coincidentally frequent among current human languages, rather than being natural in some timeless sense. We reply that our approach is then appropriate when the goal of typology is to estimate the distribution of *actual* human languages—a distribution that can be utilized in principle (and also in practice, as we show) to predict properties of *actual* languages from outside the training set.

A different possible goal of typology is a theory of *natural* human languages. This goal would require a more complex approach. One should not imagine that natural languages are drawn in a vacuum from some single, stationary distribution. Rather, each language is drawn *conditionally* on its parent language. Thus, one should estimate a stochastic model of the evolution of linguistic systems through time, and identify "naturalness" with

[10] This could happen because learners have evolved to expect the languages (the Baldwin effect), or because the languages have evolved to be easily learned (universal grammar).

	BPP			MPP			DPP		
		changes from $n-1$			changes from $n-1$			changes from $n-1$	
n	MAP inventory	additions	deletions	MAP inventory	additions	deletions	MAP inventory	additions	deletions
1	i	i		ə	ə		ə	ə	
2	i, u	u		i, u	i, u	ə	i, u	i, u	ə
3	i, u, a	a		i, u, a	a		i, u, a	a	
4	i, u, a, o	o		i, u, a, e	e		i, u, a, o	o	
5	i, u, a, o, e	e		i, u, a, e, ə	ə		i, u, a, o, ə	o	

Table 2: Highest-probability inventory of each size according to our three models (prototype-based embeddings and $d = 3$). The MAP configuration is computed by brute-force enumeration for small n.

the directions in which this system tends to evolve.

Energy Minimization Approaches. The traditional energy-based approach (Liljencrants and Lindblom, 1972) to vowel simulation minimizes the following objective (written in our notation):

$$\mathcal{E}(m) = \sum_{1 \leq i < j \leq m} \frac{1}{||\boldsymbol{e}(v_i) - \boldsymbol{e}(v_j)||^2}, \quad (9)$$

where the vectors $\boldsymbol{e}(v_i) \in \mathbb{R}^r$ are not spit out of a deep network, as in our case, but rather directly optimized. Liljencrants and Lindblom (1972) propose a coordinate descent algorithm to optimize $\mathcal{E}(m)$. While this is not in itself a probabilistic model, they generate diverse vowel systems through random restarts that find different local optima (a kind of *deterministic* evolutionary mechanism). We note that equation (9) assumes that the number of vowels m is given, and only encodes a notion of dispersion. Roark (2001) subsequently extended equation (9) to include the notion of focalization.

Vowel Inventory Size. A fatal flaw of the traditional energy minimization paradigm is that it has no clear way to compare vowel inventories of different sizes. The problem is quite crippling since, in general, inventories with fewer vowels will have lower energy. This does not match reality—the empirical distribution over inventory sizes (shown in Figure 5) shows that the mode is actually 5 and small inventories are uncommon: no 1-vowel inventory is attested and only one 2-vowel inventory is known. A probabilistic model over *all* vowel systems must implicitly model the size of the system. Indeed, our models pit all potential inventories against each other, bestowing the extra burden to match the empirical distribution over size.

Frequency of Inventories. Another problem is the inability to model frequency. While for inventories of a modest size (3-5 vowels) there are very few unique attested systems, there is a plethora of attested larger vowel systems. The energy minimization paradigm has no principled manner to tell the scientist how likely a novel system may be. Appealing again to the empirical distribution over attested vowel systems, we consider the relative diversity of systems of each size. We graph this in Figure 5. Consider all vowel systems of size 7. There are $\binom{|\mathcal{V}|}{7}$ potential inventories, yet the empirical distribution is remarkably peaked. Our probabilistic models have the advantage in this context as well, as they naturally quantify the likelihood of an individual inventory.

Typology is a Small-Data Problem. In contrast to many common problems in applied NLP, e.g., part-of-speech tagging, parsing and machine translation, the modeling of linguistic typology is fundamentally a "small-data" problem. Out of the 7105 languages on earth, we only have linguistic annotation for 2600 of them (Comrie et al., 2013). Moreover, we only have *phonetic and phonological annotation* for a much smaller set of languages—between 300-500 (Maddieson, 2013). Given the paucity of data, overfitting on only those attested languages is a dangerous possibility—just because a certain inventory has never been attested, it is probably wrong to conclude that it is impossible—or even improbable—on that basis alone. By analogy to language modeling, almost all sentences observed in practice are novel with respect to the training data, but we still must employ a principled manner to discriminate high-probability sentences (which are syntactically and semantically coherent) from low-probability ones. Probabilistic modeling provides a natural paradigm for this sort of investigation—machine learning has developed well-understood smoothing techniques, e.g., regularization with tuning on a held-out dev set, to avoid overfitting in a small-data scenario.

Related Work in NLP. Various point processes have been previously applied to potpourri of tasks

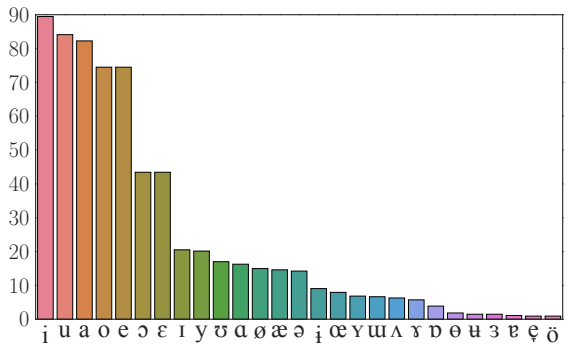


Figure 4: Percentage of the vowel inventories (y-axis) in the Becker-Kristal corpus (Becker-Kristal, 2010) that have a given vowel (shown in IPA along the x-axis).

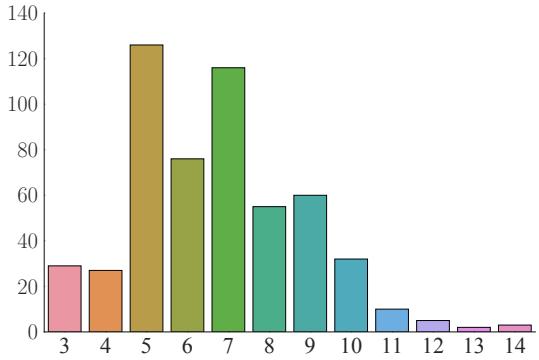


Figure 5: Histogram of the sizes of different vowel inventories in the corpus. The x-axis is the size of the vowel inventory and the y-axis is the number of inventories with that size.

in NLP. Determinantal point processes have found a home in the literature in tasks that require diversity. E.g., DPPs have achieved state-of-the-art results on multi-document document summarization (Kulesza and Taskar, 2011), news article selection (Affandi et al., 2012) recommender systems (Gartrell et al., 2017), joint clustering of verbal lexical semantic properties (Reichart and Korhonen, 2013), *inter alia*. Poisson point processes have also been applied to NLP problems: Yee et al. (2015) model the emerging topic on social media using a homogeneous point process and Lukasik et al. (2015) apply a log-Gaussian point process, a variant of the Poisson point process, to rumor detection in Twitter. We are unaware of previous attempts to probabilistically model vowel inventory typology.

Future Work. This work lends itself to several technical extensions. One could expand the function $\boldsymbol{f}$ to more completely characterize each vowel's acoustic properties, perceptual properties, or distinctive features (footnote 7). One could generalize our point process models to sample finite subsets from the *continuous* space of vowels (footnote 3). One could consider augmenting the MPP with a new factor that explicitly controls the size of the vowel inventory. Richer families of point processes might also be worth exploring. For example, perhaps the vowel inventory is generated by some temporal mechanism with latent intermediate steps, such as sequential selection of the vowels or evolutionary drift of the inventory. Another possibility is that vowel systems tend to reuse distinctive features or even follow factorial designs, so that an inventory with creaky front vowels also tends to have creaky back vowels.

9 Conclusions

We have presented a series of point process models for the modeling of vowel system inventory typology with the goal of a mathematical grounding for research in phonological typology. All models were additionally given a deep parameterization to learn representations similar to perceptual space in cognitive science. Also, we motivated our preference for *probabilistic* modeling in linguistic typology over previously proposed computational approaches and argued it is a more natural research paradigm. Additionally, we have introduced several novel evaluation metrics for research in vowel-system typology, which we hope will spark further interest in the area. Their performance was empirically validated on the Becker-Kristal corpus, which includes data from over 200 languages.

Acknowledgments

The first author was funded by an NDSEG graduate fellowship, and the second author by NSF grant IIS-1423276. We would like to thank Tim Vieira and Huda Khayrallah for helpful initial feedback.

References

David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. 1985. A learning algorithm for Boltzmann machines. *Cognitive Science* 9(1):147–169.

Raja Hafiz Affandi, Alex Kulesza, and Emily B. Fox. 2012. Markov determinantal point processes. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*. pages 26–35.

Roy Becker-Kristal. 2010. *Acoustic Typology of Vowel Inventories and Dispersion Theory: Insights from a Large Cross-Linguistic Corpus*. Ph.D. thesis, UCLA.

Paulus Petrus Gerardus Boersma et al. 2002. Praat, a system for doing phonetics by computer. *Glot International* 5.

Alexei Borodin and Eric M. Rains. 2005. Eynard-Mehta theorem, Schur process, and their Pfaffian analogs. *Journal of Statistical Physics* 121(3-4):291–317.

Bernard Comrie, Matthew S. Dryer, David Gil, and Martin Haspelmath. 2013. Introduction. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*, Max Planck Institute for Evolutionary Anthropology, Leipzig. http://wals.info/chapter/s1.

Gregory F. Cooper. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* 42(2-3):393–405.

Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. 2017. Low-rank factorization of determinantal point processes pages 1912–1918.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (6):721–741.

Matthew K. Gordon. 2016. *Phonological Typology*. Oxford.

Geoffrey E. Hinton and Terry J. Sejnowski. 1986. Learning and relearning in Boltzmann machines. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing*, MIT Press, volume 2, chapter 7, pages 282–317.

Ernst Ising. 1925. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei* 31(1):253–258.

Roman Jakobson. 1941. *Kindersprache, Aphasie und allgemeine Lautgesetze*. Suhrkamp Frankfurt aM.

Alex Kulesza and Ben Taskar. 2011. Learning determinantal point processes. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*. pages 419–427.

Alex Kulesza and Ben Taskar. 2012. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning* 5(2–3):123–286.

Peter Ladefoged and Keith Johnson. 2014. *A Course in Phonetics*. Centage.

Peter Ladefoged and Ian Maddieson. 1996. *The Sounds of the World's Languages*. Oxford.

Johan Liljencrants and Björn Lindblom. 1972. Numerical simulation of vowel quality systems: The role of perceptual contrast. *Language* pages 839–862.

Björn Lindblom. 1986. Phonetic universals in vowel systems. *Experimental Phonology* pages 13–44.

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

Qiang Liu, Jian Peng, Alexander T. Ihler, and John W. Fisher III. 2015. Estimating the partition function by discriminance sampling. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*. pages 514–522.

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Point process modelling of rumour dynamics in social media. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 518–523.

Odile Macchi. 1975. The coincidence approach to stochastic point processes. *Advances in Applied Probability* pages 83–122.

Ian Maddieson. 2013. Vowel quality inventories. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*, Max Planck Institute for Evolutionary Anthropology, Leipzig. http://wals.info/chapter/2.

Steven Moran, Daniel McCloy, and Richard Wright. 2014. PHOIBLE online. *Leipzig: Max Planck Institute for Evolutionary Anthropology* .

Terrance M. Nearey and Michael Kiefte. 2003. Comparison of several proposed perceptual representations of vowel spectra. *Proceedings of the XVth International Congress of Phonetic Sciences* 1:1005–1008.

Roi Reichart and Anna Korhonen. 2013. Improved lexical acquisition through DPP-based verb clustering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 862–872.

Brian Roark. 2001. Explaining vowel inventory tendencies via simulation: Finding a role for quantal locations and formant normalization. In *North East Linguistic Society*. volume 31, pages 419–434.

Christian P. Robert and George Casella. 2005. *Monte Carlo Statistical Methods*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Burton S. Rosner and John B. Pickering. 1994. *Vowel Perception and Production*. Oxford University Press.

Jean-Luc Schwartz, Louis-Jean Boë, Nathalie Vallée, and Christian Abry. 1997. The dispersion-focalization theory of vowel systems. *Journal of Phonetics* 25(3):255–286.

Roger N. Shepard. 1987. Toward a universal law of generalization for psychological science. *Science* 237(4820):1317–1323.

Kenneth N. Stevens. 1972. The quantal nature of speech: Evidence from articulatory-acoustic data. In E. E. David and P. B. Denes, editors, *Human Communication: A Unified View*, McGraw-Hill, pages 51–56.

Kenneth N Stevens. 1989. On the quantal nature of speech. *Journal of Phonetics* 17:3–45.

Wilson L. Taylor. 1953. Cloze procedure: a new tool for measuring readability. *Journalism and Mass Communication Quarterly* 30(4):415.

M. N. M. Van Lieshout. 2000. *Markov Point Processes and Their Applications*. Imperial College Press, London.

Viveka Velupillai. 2012. *An Introduction to Linguistic Typology*. John Benjamins Publishing Company.

Connie Yee, Nathan Keane, and Liang Zhou. 2015. Modeling and characterizing social media topics using the gamma distribution. In *EVENTS*. pages 117–122.

Adversarial Multi-Criteria Learning for Chinese Word Segmentation

Xinchi Chen, Zhan Shi, Xipeng Qiu*, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{xinchichen13,zshi16,xpqiu,xjhuang}@fudan.edu.cn

Abstract

Different linguistic perspectives causes many diverse segmentation criteria for Chinese word segmentation (CWS). Most existing methods focus on improve the performance for each single criterion. However, it is interesting to exploit these different criteria and mining their common underlying knowledge. In this paper, we propose adversarial multi-criteria learning for CWS by integrating shared knowledge from multiple heterogeneous segmentation criteria. Experiments on eight corpora with heterogeneous segmentation criteria show that the performance of each corpus obtains a significant improvement, compared to single-criterion learning. Source codes of this paper are available on Github[1].

1 Introduction

Chinese word segmentation (CWS) is a preliminary and important task for Chinese natural language processing (NLP). Currently, the state-of-the-art methods are based on statistical supervised learning algorithms, and rely on a large-scale annotated corpus whose cost is extremely expensive. Although there have been great achievements in building CWS corpora, they are somewhat incompatible due to different segmentation criteria. As shown in Table 1, given a sentence "姚明进入总决赛 (YaoMing reaches the final)", the two commonly-used corpora, PKU's People's Daily (PKU) (Yu et al., 2001) and Penn Chinese Treebank (CTB) (Fei, 2000), use different segmentation criteria. In a sense, it is a waste of resources if we fail to fully exploit these corpora.

Corpora	Yao	Ming	reaches	the final	
CTB	姚明		进入	总决赛	
PKU	姚	明	进入	总	决赛

Table 1: Illustration of the different segmentation criteria.

Recently, some efforts have been made to exploit heterogeneous annotation data for Chinese word segmentation or part-of-speech tagging (Jiang et al., 2009; Sun and Wan, 2012; Qiu et al., 2013; Li et al., 2015, 2016). These methods adopted stacking or multi-task architectures and showed that heterogeneous corpora can help each other. However, most of these model adopt the shallow linear classifier with discrete features, which makes it difficult to design the shared feature spaces, usually resulting in a complex model. Fortunately, recent deep neural models provide a convenient way to share information among multiple tasks (Collobert and Weston, 2008; Luong et al., 2015; Chen et al., 2016).

In this paper, we propose an adversarial multi-criteria learning for CWS by integrating shared knowledge from multiple segmentation criteria. Specifically, we regard each segmentation criterion as a single task and propose three different shared-private models under the framework of multi-task learning (Caruana, 1997; Ben-David and Schuller, 2003), where a shared layer is used to extract the criteria-invariant features, and a private layer is used to extract the criteria-specific features. Inspired by the success of adversarial strategy on domain adaption (Ajakan et al., 2014; Ganin et al., 2016; Bousmalis et al., 2016), we further utilize adversarial strategy to make sure the shared layer can extract the common underlying and criteria-invariant features, which are suitable for all the criteria. Finally, we exploit the eight segmentation criteria on the five simplified Chi-

*Corresponding author.
[1] https://github.com/FudanNLP

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1193–1203
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1110

nese and three traditional Chinese corpora. Experiments show that our models are effective to improve the performance for CWS. We also observe that traditional Chinese could benefit from incorporating knowledge from simplified Chinese.

The contributions of this paper could be summarized as follows.

- Multi-criteria learning is first introduced for CWS, in which we propose three shared-private models to integrate multiple segmentation criteria.
- An adversarial strategy is used to force the shared layer to learn criteria-invariant features, in which an new objective function is also proposed instead of the original cross-entropy loss.
- We conduct extensive experiments on eight CWS corpora with different segmentation criteria, which is by far the largest number of datasets used simultaneously.

2 General Neural Model for Chinese Word Segmentation

Chinese word segmentation task is usually regarded as a character based sequence labeling problem. Specifically, each character in a sentence is labeled as one of $\mathcal{L} = \{B, M, E, S\}$, indicating the begin, middle, end of a word, or a word with single character. There are lots of prevalent methods to solve sequence labeling problem such as maximum entropy Markov model (MEMM), conditional random fields (CRF), etc. Recently, neural networks are widely applied to Chinese word segmentation task for their ability to minimize the effort in feature engineering (Zheng et al., 2013; Pei et al., 2014; Chen et al., 2015a,b).

Specifically, given a sequence with n characters $X = \{x_1, \ldots, x_n\}$, the aim of CWS task is to figure out the ground truth of labels $Y^* = \{y_1^*, \ldots, y_n^*\}$:

$$Y^* = \arg\max_{Y \in \mathcal{L}^n} p(Y|X), \quad (1)$$

where $\mathcal{L} = \{B, M, E, S\}$.

The general architecture of neural CWS could be characterized by three components: (1) a character embedding layer; (2) feature layers consisting of several classical neural networks and (3) a tag inference layer. The role of feature layers is to extract features, which could be either convolution neural network or recurrent neural network. In this paper, we adopt the bi-directional long short-term memory neural networks followed by CRF as the tag inference layer. Figure 1 illustrates the general architecture of CWS.

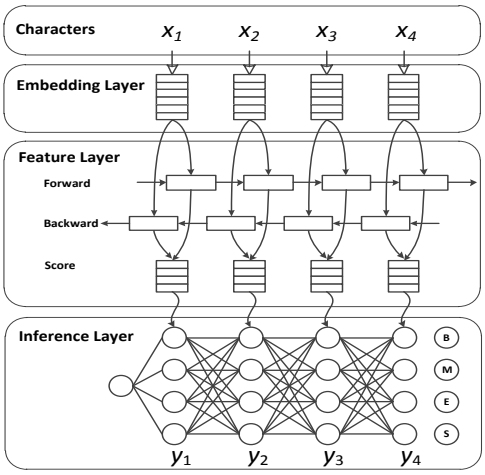


Figure 1: General neural architecture for Chinese word segmentation.

2.1 Embedding layer

In neural models, the first step usually is to map discrete language symbols to distributed embedding vectors. Formally, we lookup embedding vector from embedding matrix for each character x_i as $\mathbf{e}_{x_i} \in \mathbb{R}^{d_e}$, where d_e is a hyper-parameter indicating the size of character embedding.

2.2 Feature layers

We adopt bi-directional long short-term memory (Bi-LSTM) as feature layers. While there are numerous LSTM variants, here we use the LSTM architecture used by (Jozefowicz et al., 2015), which is similar to the architecture of (Graves, 2013) but without peep-hole connections.

LSTM LSTM introduces gate mechanism and memory cell to maintain long dependency information and avoid gradient vanishing. Formally, LSTM, with input gate $\mathbf{i}$, output gate $\mathbf{o}$, forget gate $\mathbf{f}$ and memory cell $\mathbf{c}$, could be expressed as:

$$\begin{bmatrix} \mathbf{i}_i \\ \mathbf{o}_i \\ \mathbf{f}_i \\ \tilde{\mathbf{c}}_i \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \phi \end{bmatrix} \left(\mathbf{W}_g{}^{\mathsf{T}} \begin{bmatrix} \mathbf{e}_{x_i} \\ \mathbf{h}_{i-1} \end{bmatrix} + \mathbf{b}_g \right), \quad (2)$$

$$\mathbf{c}_i = \mathbf{c}_{i-1} \odot \mathbf{f}_i + \tilde{\mathbf{c}}_i \odot \mathbf{i}_i, \quad (3)$$

$$\mathbf{h}_i = \mathbf{o}_i \odot \phi(\mathbf{c}_i), \quad (4)$$

where $\mathbf{W}_g \in \mathbb{R}^{(d_e+d_h)\times 4d_h}$ and $\mathbf{b}_g \in \mathbb{R}^{4d_h}$ are trainable parameters. d_h is a hyper-parameter, in-

dicating the hidden state size. Function $\sigma(\cdot)$ and $\phi(\cdot)$ are sigmoid and tanh functions respectively.

Bi-LSTM In order to incorporate information from both sides of sequence, we use bi-directional LSTM (Bi-LSTM) with forward and backward directions. The update of each Bi-LSTM unit can be written precisely as follows:

$$\mathbf{h}_i = \overrightarrow{\mathbf{h}}_i \oplus \overleftarrow{\mathbf{h}}_i, \quad (5)$$

$$= \text{Bi-LSTM}(\mathbf{e}_{x_i}, \overrightarrow{\mathbf{h}}_{i-1}, \overleftarrow{\mathbf{h}}_{i+1}, \theta), \quad (6)$$

where $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ are the hidden states at position i of the forward and backward LSTMs respectively; $\oplus$ is concatenation operation; θ denotes all parameters in Bi-LSTM model.

2.3 Inference Layer

After extracting features, we employ conditional random fields (CRF) (Lafferty et al., 2001) layer to inference tags. In CRF layer, $p(Y|X)$ in Eq (1) could be formalized as:

$$p(Y|X) = \frac{\Psi(Y|X)}{\sum_{Y' \in \mathcal{L}^n} \Psi(Y'|X)}. \quad (7)$$

Here, $\Psi(Y|X)$ is the potential function, and we only consider interactions between two successive labels (first order linear chain CRFs):

$$\Psi(Y|X) = \prod_{i=2}^{n} \psi(X, i, y_{i-1}, y_i), \quad (8)$$

$$\psi(\mathbf{x}, i, y', y) = \exp(s(X, i)_y + \mathbf{b}_{y'y}), \quad (9)$$

where $\mathbf{b}_{y'y} \in \mathbf{R}$ is trainable parameters respective to label pair (y', y). Score function $s(X, i) \in \mathbb{R}^{|\mathcal{L}|}$ assigns score for each label on tagging the i-th character:

$$s(X, i) = \mathbf{W}_s^\top \mathbf{h}_i + \mathbf{b}_s, \quad (10)$$

where $\mathbf{h}_i$ is the hidden state of Bi-LSTM at position i; $\mathbf{W}_s \in \mathbb{R}^{d_h \times |\mathcal{L}|}$ and $\mathbf{b}_s \in \mathbb{R}^{|\mathcal{L}|}$ are trainable parameters.

3 Multi-Criteria Learning for Chinese Word Segmentation

Although neural models are widely used on CWS, most of them cannot deal with incompatible criteria with heterogonous segmentation criteria simultaneously.

Inspired by the success of multi-task learning (Caruana, 1997; Ben-David and Schuller, 2003; Liu et al., 2016a,b), we regard the heterogenous criteria as multiple "related" tasks, which could improve the performance of each other simultaneously with shared information.

Formally, assume that there are M corpora with heterogeneous segmentation criteria. We refer $\mathcal{D}_m$ as corpus m with N_m samples:

$$\mathcal{D}_m = \{(X_i^{(m)}, Y_i^{(m)})\}_{i=1}^{N_m}, \quad (11)$$

where X_i^m and Y_i^m denote the i-th sentence and the corresponding label in corpus m.

To exploit the shared information between these different criteria, we propose three sharing models for CWS task as shown in Figure 2. The feature layers of these three models consist of a private (criterion-specific) layer and a shared (criterion-invariant) layer. The difference between three models is the information flow between the task layer and the shared layer. Besides, all of these three models also share the embedding layer.

3.1 Model-I: Parallel Shared-Private Model

In the feature layer of Model-I, we regard the private layer and shared layer as two parallel layers. For corpus m, the hidden states of shared layer and private layer are:

$$\mathbf{h}_i^{(s)} = \text{Bi-LSTM}(\mathbf{e}_{x_i}, \overrightarrow{\mathbf{h}}_{i-1}^{(s)}, \overleftarrow{\mathbf{h}}_{i+1}^{(s)}, \theta_s), \quad (12)$$

$$\mathbf{h}_i^{(m)} = \text{Bi-LSTM}(\mathbf{e}_{x_i}, \overrightarrow{\mathbf{h}}_{i-1}^{(m)}, \overleftarrow{\mathbf{h}}_{i+1}^{(m)}, \theta_m), \quad (13)$$

and the score function in the CRF layer is computed as:

$$s^{(m)}(X, i) = {\mathbf{W}_s^{(m)}}^\top \begin{bmatrix} \mathbf{h}_i^{(s)} \\ \mathbf{h}_i^{(m)} \end{bmatrix} + \mathbf{b}_s^{(m)}, \quad (14)$$

where $\mathbf{W}_s^{(m)} \in \mathbb{R}^{2d_h \times |\mathcal{L}|}$ and $\mathbf{b}_s^{(m)} \in \mathbb{R}^{|\mathcal{L}|}$ are criterion-specific parameters for corpus m.

3.2 Model-II: Stacked Shared-Private Model

In the feature layer of Model-II, we arrange the shared layer and private layer in stacked manner. The private layer takes output of shared layer as input. For corpus m, the hidden states of shared layer and private layer are:

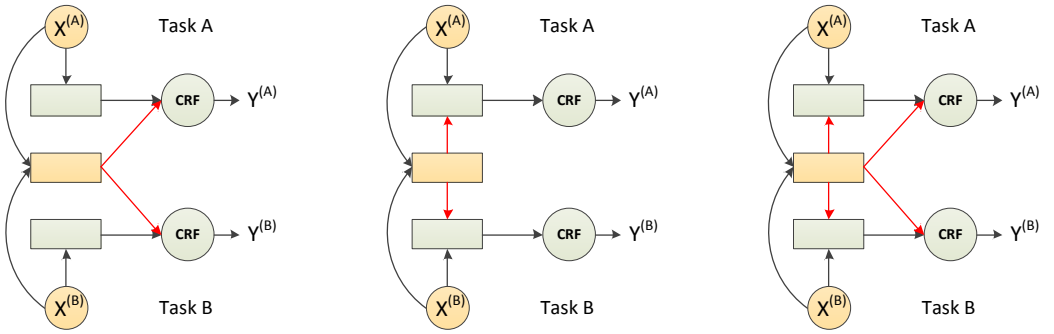
$$\mathbf{h}_i^{(s)} = \text{Bi-LSTM}(\mathbf{e}_{x_i}, \overrightarrow{\mathbf{h}}_{i-1}^{(s)}, \overleftarrow{\mathbf{h}}_{i+1}^{(s)}, \theta_s), \quad (15)$$

$$\mathbf{h}_i^{(m)} = \text{Bi-LSTM}(\begin{bmatrix} \mathbf{e}_{x_i} \\ \mathbf{h}_i^{(s)} \end{bmatrix}, \overrightarrow{\mathbf{h}}_{i-1}^{(m)}, \overleftarrow{\mathbf{h}}_{i+1}^{(m)}, \theta_m) \quad (16)$$

and the score function in the CRF layer is computed as:

$$s^{(m)}(X, i) = {\mathbf{W}_s^{(m)}}^\top \mathbf{h}_i^{(m)} + \mathbf{b}_s^{(m)}, \quad (17)$$

where $\mathbf{W}_s^{(m)} \in \mathbb{R}^{2d_h \times |\mathcal{L}|}$ and $\mathbf{b}_s^{(m)} \in \mathbb{R}^{|\mathcal{L}|}$ are criterion-specific parameters for corpus m.



(a) Model-I (b) Model-II (c) Model-III

Figure 2: Three shared-private models for multi-criteria learning. The yellow blocks are the shared Bi-LSTM layer, while the gray block are the private Bi-LSTM layer. The yellow circles denote the shared embedding layer. The red information flow indicates the difference between three models.

3.3 Model-III: Skip-Layer Shared-Private Model

In the feature layer of Model-III, the shared layer and private layer are in stacked manner as Model-II. Additionally, we send the outputs of shared layer to CRF layer directly.

The Model III can be regarded as a combination of Model-I and Model-II. For corpus m, the hidden states of shared layer and private layer are the same with Eq (15) and (16), and the score function in CRF layer is computed as the same as Eq (14).

3.4 Objective function

The parameters of the network are trained to maximize the log conditional likelihood of true labels on all the corpora. The objective function $\mathcal{J}_{seg}$ can be computed as:

$$\mathcal{J}_{seg}(\Theta^m, \Theta^s) = \sum_{m=1}^{M} \sum_{i=1}^{N_m} \log p(Y_i^{(m)} | X_i^{(m)}; \Theta^m, \Theta^s), \tag{18}$$

where Θ^m and Θ^s denote all the parameters in private and shared layers respectively.

4 Incorporating Adversarial Training for Shared Layer

Although the shared-private model separates the feature space into shared and private spaces, there is no guarantee that sharable features do not exist in private feature space, or vice versa. Inspired by the work on domain adaptation (Ajakan et al., 2014; Ganin et al., 2016; Bousmalis et al., 2016), we hope that the features extracted by shared layer is invariant across the heterogonous segmentation criteria. Therefore, we jointly optimize the shared layer via adversarial training (Goodfellow et al., 2014).

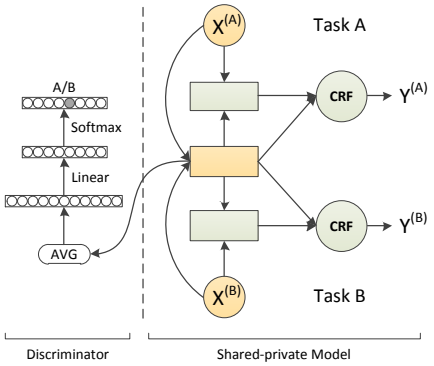


Figure 3: Architecture of Model-III with adversarial training strategy for shared layer. The discriminator firstly averages the hidden states of shared layer, then derives probability over all possible criteria by applying softmax operation after a linear transformation.

Therefore, besides the task loss for CWS, we additionally introduce an adversarial loss to prevent criterion-specific feature from creeping into shared space as shown in Figure 3. We use a criterion discriminator which aims to recognize which criterion the sentence is annotated by using the shared features.

Specifically, given a sentence X with length n, we refer to $\mathbf{h}_X^{(s)}$ as shared features for X in one of the sharing models. Here, we compute $\mathbf{h}_X^{(s)}$ by simply averaging the hidden states of shared layer $\mathbf{h}_X^{(s)} = \frac{1}{n}\sum_i^n \mathbf{h}_{x_i}^{(s)}$. The criterion discriminator computes the probability $p(\cdot|X)$ over all criteria as:

$$p(\cdot|X; \Theta^d, \Theta^s) = \text{softmax}(\mathbf{W}_d^\top \mathbf{h}_X^{(s)} + \mathbf{b}_d), \tag{19}$$

where Θ^d indicates the parameters of criterion discriminator $\mathbf{W}_d \in \mathbb{R}^{d_h \times M}$ and $\mathbf{b}_d \in \mathbb{R}^M$; Θ^s denotes the parameters of shared layers.

4.1 Adversarial loss function

The criterion discriminator maximizes the cross entropy of predicted criterion distribution $p(\cdot|X)$ and true criterion.

$$\max_{\Theta^d} \mathcal{J}^1_{adv}(\Theta^d) = \sum_{m=1}^{M} \sum_{i=1}^{N_m} \log p(m|X_i^{(m)}; \Theta^d, \Theta^s). \quad (20)$$

An adversarial loss aims to produce shared features, such that a criterion discriminator cannot reliably predict the criterion by using these shared features. Therefore, we maximize the entropy of predicted criterion distribution when training shared parameters.

$$\max_{\Theta^s} \mathcal{J}^2_{adv}(\Theta^s) = \sum_{m=1}^{M} \sum_{i=1}^{N_m} \mathrm{H}\left(p(m|X_i^{(m)}; \Theta^d, \Theta^s)\right), \quad (21)$$

where $\mathrm{H}(p) = -\sum_i p_i \log p_i$ is an entropy of distribution p.

Unlike (Ganin et al., 2016), we use entropy term instead of negative cross-entropy.

5 Training

Finally, we combine the task and adversarial objective functions.

$$\mathcal{J}(\Theta; \mathcal{D}) = \mathcal{J}_{seg}(\Theta^m, \Theta^s) + \mathcal{J}^1_{adv}(\Theta^d) + \lambda \mathcal{J}^2_{adv}(\Theta^s), \quad (22)$$

where λ is the weight that controls the interaction of the loss terms and $\mathcal{D}$ is the training corpora.

The training procedure is to optimize two discriminative classifiers alternately as shown in Algorithm 1. We use Adam (Kingma and Ba, 2014) with minibatchs to maximize the objectives.

Notably, when using adversarial strategy, we firstly train 2400 epochs (each epoch only trains on eight batches from different corpora), then we only optimize $\mathcal{J}_{seg}(\Theta^m, \Theta^s)$ with Θ^s fixed until convergence (early stop strategy).

Algorithm 1 Adversarial multi-criteria learning for CWS task.

for $i = 1; i <= n\_epoch; i++$ **do**
  # Train tag predictor for CWS
  for $m = 1; m <= M; m++$ **do**
    # Randomly pick data from corpus m
    $\mathcal{B} = \{X, Y\}_1^{b_m} \in \mathcal{D}^m$
    $\Theta^s \mathrel{+}= \alpha \nabla_{\Theta^s} \mathcal{J}(\Theta; \mathcal{B})$
    $\Theta^m \mathrel{+}= \alpha \nabla_{\Theta^m} \mathcal{J}(\Theta; \mathcal{B})$
  end for
  # Train criterion discriminator
  for $m = 1; m <= M; m++$ **do**
    $\mathcal{B} = \{X, Y\}_1^{b_m} \in \mathcal{D}^m$
    $\Theta^d \mathrel{+}= \alpha \nabla_{\Theta^d} \mathcal{J}(\Theta; \mathcal{B})$
  end for
end for

6 Experiments

6.1 Datasets

To evaluate our proposed architecture, we experiment on eight prevalent CWS datasets from SIGHAN2005 (Emerson, 2005) and SIGHAN2008 (Jin and Chen, 2008). Table 2 gives the details of the eight datasets. Among these datasets, AS, CITYU and CKIP are traditional Chinese, while the remains, MSRA, PKU, CTB, NCC and SXU, are simplified Chinese. We use 10% data of shuffled train set as development set for all datasets.

6.2 Experimental Configurations

For hyper-parameter configurations, we set both the character embedding size d_e and the dimensionality of LSTM hidden states d_h to 100. The initial learning rate α is set to 0.01. The loss weight coefficient λ is set to 0.05. Since the scale of each dataset varies, we use different training batch sizes for datasets. Specifically, we set batch sizes of AS and MSR datasets as 512 and 256 respectively, and 128 for remains. We employ dropout strategy on embedding layer, keeping 80% inputs (20% dropout rate).

For initialization, we randomize all parameters following uniform distribution at $(-0.05, 0.05)$. We simply map traditional Chinese characters to simplified Chinese, and optimize on the same character embedding matrix across datasets, which is pre-trained on Chinese Wikipedia corpus, using word2vec toolkit (Mikolov et al., 2013). Following previous work (Chen et al., 2015b; Pei et al., 2014), all experiments including baseline results are using pre-trained character embedding with bigram feature.

6.3 Overall Results

Table 3 shows the experiment results of the proposed models on test sets of eight CWS datasets, which has three blocks.

(1) In the first block, we can see that the performance is boosted by using Bi-LSTM, and the

Datasets			Words	Chars	Word Types	Char Types	Sents	OOV Rate
Sighan05	MSRA	Train	2.4M	4.1M	88.1K	5.2K	86.9K	-
		Test	0.1M	0.2M	12.9K	2.8K	4.0K	2.60%
	AS	Train	5.4M	8.4M	141.3K	6.1K	709.0K	-
		Test	0.1M	0.2M	18.8K	3.7K	14.4K	4.30%
Sighan08	PKU	Train	1.1M	1.8M	55.2K	4.7K	47.3K	-
		Test	0.2M	0.3M	17.6K	3.4K	6.4K	-
	CTB	Train	0.6M	1.1M	42.2K	4.2K	23.4K	-
		Test	0.1M	0.1M	9.8K	2.6K	2.1K	5.55%
	CKIP	Train	0.7M	1.1M	48.1K	4.7K	94.2K	-
		Test	0.1M	0.1M	15.3K	3.5K	10.9K	7.41%
	CITYU	Train	1.1M	1.8M	43.6K	4.4K	36.2K	-
		Test	0.2M	0.3M	17.8K	3.4K	6.7K	8.23%
	NCC	Train	0.5M	0.8M	45.2K	5.0K	18.9K	-
		Test	0.1M	0.2M	17.5K	3.6K	3.6K	4.74%
	SXU	Train	0.5M	0.9M	32.5K	4.2K	17.1K	-
		Test	0.1M	0.2M	12.4K	2.8K	3.7K	5.12%

Table 2: Details of the eight datasets.

performance of Bi-LSTM cannot be improved by merely increasing the depth of networks. In addition, although the F value of LSTM model in (Chen et al., 2015b) is 97.4%, they additionally incorporate an external idiom dictionary.

(2) In the second block, our proposed three models based on multi-criteria learning boost performance. Model-I gains 0.75% improvement on averaging F-measure score compared with Bi-LSTM result (94.14%). Only the performance on MSRA drops slightly. Compared to the baseline results (Bi-LSTM and stacked Bi-LSTM), the proposed models boost the performance with the help of exploiting information across these heterogeneous segmentation criteria. Although various criteria have different segmentation granularities, there are still some underlying information shared. For instance, MSRA and CTB treat family name and last name as one token "宁泽涛 (NingZeTao)", whereas some other datasets, like PKU, regard them as two tokens, "宁 (Ning)" and "泽涛 (ZeTao)". The partial boundaries (before "宁 (Ning)" or after "涛 (Tao)") can be shared.

(3) In the third block, we introduce adversarial training. By introducing adversarial training, the performances are further boosted, and Model-I is slightly better than Model-II and Model-III. The adversarial training tries to make shared layer keep criteria-invariant features. For instance, as shown in Table 3, when we use shared information, the performance on MSRA drops (worse than baseline result). The reason may be that the shared parameters bias to other segmentation criteria and introduce noisy features into shared parameters. When we additionally incorporate the adversarial strategy, we observe that the performance on MSRA is improved and outperforms the baseline results. We could also observe the improvements on other datasets. However, the boost from the adversarial strategy is not significant. The main reason might be that the proposed three sharing models implicitly attempt to keep invariant features by shared parameters and learn discrepancies by the task layer.

6.4 Speed

To further explore the convergence speed, we plot the results on development sets through epochs. Figure 4 shows the learning curve of Model-I without incorporating adversarial strategy. As shown in Figure 4, the proposed model makes progress gradually on all datasets. After about 1000 epochs, the performance becomes stable and convergent.

We also test the decoding speed, and our models process 441.38 sentences per second averagely. As the proposed models and the baseline models (Bi-LSTM and stacked Bi-LSTM) are nearly in the same complexity, all models are nearly the same efficient. However, the time consumption of training process varies from model to model. For the models without adversarial training, it costs about 10 hours for training (the same for stacked Bi-LSTM to train eight datasets), whereas it takes about 16 hours for the models with adversarial training. All the experiments are conducted on the hardware with Intel(R) Xeon(R) CPU E5-2643 v3 @ 3.40GHz and NVIDIA GeForce GTX TITAN X.

Models		MSRA	AS	PKU	CTB	CKIP	CITYU	NCC	SXU	Avg.
LSTM	P	95.13	93.66	93.96	95.36	91.85	94.01	91.45	95.02	93.81
	R	95.55	94.71	92.65	85.52	93.34	94.00	92.22	95.05	92.88
	F	95.34	94.18	93.30	**95.44**	92.59	94.00	91.83	95.04	93.97
	OOV	63.60	69.83	66.34	76.34	68.67	65.48	56.28	69.46	67.00
Bi-LSTM	P	95.70	93.64	93.67	95.19	92.44	94.00	91.86	95.11	93.95
	R	95.99	94.77	92.93	95.42	93.69	94.15	92.47	95.23	94.33
	F	**95.84**	94.20	93.30	95.30	**93.06**	**94.07**	92.17	95.17	**94.14**
	OOV	66.28	70.07	66.09	76.47	72.12	65.79	59.11	71.27	68.40
Stacked Bi-LSTM	P	95.69	93.89	94.10	95.20	92.40	94.13	91.81	94.99	94.03
	R	95.81	94.54	92.66	95.40	93.39	93.99	92.62	95.37	94.22
	F	95.75	**94.22**	**93.37**	95.30	92.89	94.06	**92.21**	**95.18**	94.12
	OOV	65.55	71.50	67.92	75.44	70.50	66.35	57.39	69.69	68.04
Multi-Criteria Learning										
Model-I	P	95.67	94.44	94.93	95.95	93.99	95.10	92.54	96.07	94.84
	R	95.82	95.09	93.73	96.00	94.52	95.60	92.69	96.08	94.94
	F	95.74	94.76	**94.33**	95.97	**94.26**	95.35	**92.61**	96.07	**94.89**
	OOV	69.89	74.13	72.96	81.12	77.58	80.00	64.14	77.05	74.61
Model-II	P	95.74	94.60	94.82	95.90	93.51	95.30	92.26	96.17	94.79
	R	95.74	95.20	93.76	95.94	94.56	95.50	92.84	95.95	94.94
	F	95.74	**94.90**	94.28	95.92	94.03	95.40	92.55	96.06	94.86
	OOV	69.67	74.87	72.28	79.94	76.67	81.05	61.51	77.96	74.24
Model-III	P	95.76	93.99	94.95	95.85	93.50	95.56	92.17	96.10	94.74
	R	95.89	95.07	93.48	96.11	94.58	95.62	92.96	96.13	94.98
	F	**95.82**	94.53	94.21	**95.98**	94.04	**95.59**	92.57	**96.12**	94.86
	OOV	70.72	72.59	73.12	81.21	76.56	82.14	60.83	77.56	74.34
Adversarial Multi-Criteria Learning										
Model-I+ADV	P	95.95	94.17	94.86	96.02	93.82	95.39	92.46	96.07	94.84
	R	96.14	95.11	93.78	96.33	94.70	95.70	93.19	96.01	95.12
	F	**96.04**	94.64	**94.32**	**96.18**	**94.26**	**95.55**	**92.83**	96.04	**94.98**
	OOV	71.60	73.50	72.67	82.48	77.59	81.40	63.31	77.10	74.96
Model-II+ADV	P	96.02	94.52	94.65	96.09	93.80	95.37	92.42	95.85	94.84
	R	95.86	94.98	93.61	95.90	94.69	95.63	93.20	96.07	94.99
	F	95.94	**94.75**	94.13	96.00	94.24	95.50	92.81	95.96	94.92
	OOV	72.76	75.37	73.13	82.19	77.71	81.05	62.16	76.88	75.16
Model-III+ADV	P	95.92	94.25	94.68	95.86	93.67	95.24	92.47	96.24	94.79
	R	95.83	95.11	93.82	96.10	94.48	95.60	92.73	96.04	94.96
	F	95.87	94.68	94.25	95.98	94.07	95.42	92.60	**96.14**	94.88
	OOV	70.86	72.89	72.20	81.65	76.13	80.71	63.22	77.88	74.44

Table 3: Results of proposed models on test sets of eight CWS datasets. There are three blocks. The first block consists of two baseline models: Bi-LSTM and stacked Bi-LSTM. The second block consists of our proposed three models without adversarial training. The third block consists of our proposed three models with adversarial training. Here, P, R, F, OOV indicate the precision, recall, F value and OOV recall rate respectively. The maximum F values in each block are highlighted for each dataset.

6.5 Error Analysis

We further investigate the benefits of the proposed models by comparing the error distributions between the single-criterion learning (baseline model Bi-LSTM) and multi-criteria learning (Model-I and Model-I with adversarial training) as shown in Figure 5. According to the results, we could observe that a large proportion of points lie above diagonal lines in Figure 5a and Figure 5b, which implies that performance benefit from integrating knowledge and complementary information from other corpora. As shown in Table 3, on the test set of CITYU, the performance of Model-I and its adversarial version (Model-I+ADV) boost from 92.17% to 95.59% and 95.42% respectively.

In addition, we observe that adversarial strategy is effective to prevent criterion specific features from creeping into shared space. For instance, the segmentation granularity of personal name is often different according to heterogenous criteria. With the help of adversarial strategy, our models could correct a large proportion of mistakes on personal name. Table 4 lists the examples from 2333-th and 89-th sentences in test sets of PKU and MSRA datasets respectively.

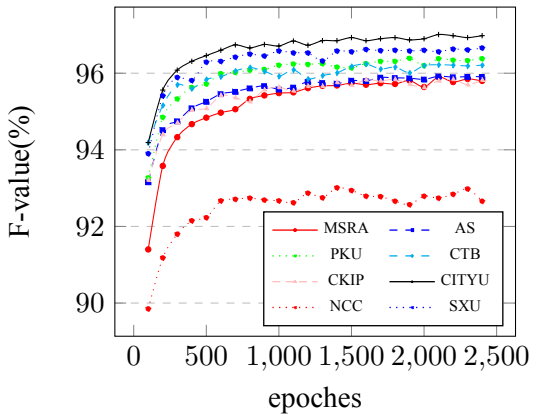
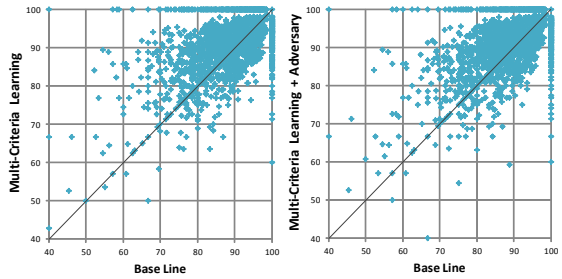


Figure 4: Convergence speed of Model-I without adversarial training on development sets of eight datasets.



(a) (b)

Figure 5: F-measure scores on test set of CITYU dataset. Each point denotes a sentence, with the (x, y) values of each point denoting the F-measure scores of the two models, respectively. (a) is comparison between Bi-LSTM and Model-I. (b) is comparison between Bi-LSTM and Model-I with adversarial training.

7 Knowledge Transfer

We also conduct experiments of whether the shared layers can be transferred to the other related tasks or domains. In this section, we investigate the ability of knowledge transfer on two experiments: (1) simplified Chinese to traditional Chinese and (2) formal texts to informal texts.

7.1 Simplified Chinese to Traditional Chinese

Traditional Chinese and simplified Chinese are two similar languages with slightly difference on character forms (e.g. multiple traditional characters might map to one simplified character). We investigate that if datasets in traditional Chinese and simplified Chinese could help each other. Table 5 gives the results of Model-I on 3 traditional Chinese datasets under the help of 5 simplified Chinese datasets. Specifically, we firstly train the model on simplified Chinese datasets, then we train traditional Chinese datasets independently with shared parameters fixed.

Models	PKU-2333		MSRA-89	
Golds	Roh 卢	Moo-hyun 武铉	Mu Ling Ying 穆玲英	
Base Line	卢武铉		穆	玲英
Model-I	卢武铉		穆	玲英
Modell-I+ADV	卢	武铉	穆玲英	

Table 4: Segmentation cases of personal names.

Models	AS	CKIP	CITYU	Avg.
Baseline(Bi-LSTM)	**94.20**	93.06	94.07	93.78
Model-I*	94.12	**93.24**	**95.20**	**94.19**

Table 5: Performance on 3 traditional Chinese datasets. Model-I* means that the shared parameters are trained on 5 simplified Chinese datasets and are fixed for traditional Chinese datasets. Here, we conduct Model-I without incorporating adversarial training strategy.

As we can see, the average performance is boosted by 0.41% on F-measure score (from 93.78% to 94.19%), which indicates that shared features learned from simplified Chinese segmentation criteria can help to improve performance on traditional Chinese. Like MSRA, as AS dataset is relatively large (train set of 5.4M tokens), the features learned by shared parameters might bias to other datasets and thus hurt performance on such large dataset AS.

7.2 Formal Texts to Informal Texts

7.2.1 Dataset

We use the NLPCC 2016 dataset[2] (Qiu et al., 2016) to evaluate our model on micro-blog texts. The NLPCC 2016 data are provided by the shared task in the 5th CCF Conference on Natural Language Processing & Chinese Computing (NLPCC 2016): Chinese Word Segmentation and POS Tagging for micro-blog Text. Unlike the popular used newswire dataset, the NLPCC 2016 dataset is collected from Sina Weibo[3], which consists of the informal texts from micro-blog with the various topics, such as finance, sports, entertainment, and so on. The information of the dataset is shown in Table 6.

[2] https://github.com/FudanNLP/NLPCC-WordSeg-Weibo

[3] http://www.weibo.com/

Dataset	Words	Chars	Word Types	Char Types	Sents	OOV Rate
Train	421,166	688,743	43,331	4,502	20,135	-
Dev	43,697	73,246	11,187	2,879	2,052	6.82%
Test	187,877	315,865	27,804	3,911	8,592	6.98%

Table 6: Statistical information of NLPCC 2016 dataset.

Models	P	R	F	OOV
Baseline(Bi-LSTM)	93.56	94.33	93.94	70.75
Model-I*	**93.65**	**94.83**	**94.24**	**74.72**

Table 7: Performances on the test set of NLPCC 2016 dataset. Model-I* means that the shared parameters are trained on 8 Chinese datasets (Table 2) and are fixed for NLPCC dataset. Here, we conduct Model-I without incorporating adversarial training strategy.

7.2.2 Results

Formal documents (like the eight datasets in Table 2) and micro-blog texts are dissimilar in many aspects. Thus, we further investigate that if the formal texts could help to improve the performance of micro-blog texts. Table 7 gives the results of Model-I on the NLPCC 2016 dataset under the help of the eight datasets in Table 2. Specifically, we firstly train the model on the eight datasets, then we train on the NLPCC 2016 dataset alone with shared parameters fixed. The baseline model is Bi-LSTM which is trained on the NLPCC 2016 dataset alone.

As we can see, the performance is boosted by 0.30% on F-measure score (from 93.94% to 94.24%), and we could also observe that the OOV recall rate is boosted by 3.97%. It shows that the shared features learned from formal texts can help to improve the performance on of micro-blog texts.

8 Related Works

There are many works on exploiting heterogeneous annotation data to improve various NLP tasks. Jiang et al. (2009) proposed a stacking-based model which could train a model for one specific desired annotation criterion by utilizing knowledge from corpora with other heterogeneous annotations. Sun and Wan (2012) proposed a structure-based stacking model to reduce the approximation error, which makes use of structured features such as sub-words. These models are unidirectional aid and also suffer from error propagation problem.

Qiu et al. (2013) used multi-tasks learning framework to improve the performance of POS tagging on two heterogeneous datasets. Li et al. (2015) proposed a coupled sequence labeling model which could directly learn and infer two heterogeneous annotations. Chao et al. (2015) also utilize multiple corpora using coupled sequence labeling model. These methods adopt the shallow classifiers, therefore suffering from the problem of defining shared features.

Our proposed models use deep neural networks, which can easily share information with hidden shared layers. Chen et al. (2016) also adopted neural network models for exploiting heterogeneous annotations based on neural multi-view model, which can be regarded as a simplified version of our proposed models by removing private hidden layers.

Unlike the above models, we design three sharing-private architectures and keep shared layer to extract criterion-invariance features by introducing adversarial training. Moreover, we fully exploit eight corpora with heterogeneous segmentation criteria to model the underlying shared information.

9 Conclusions & Future Works

In this paper, we propose adversarial multi-criteria learning for CWS by fully exploiting the underlying shared knowledge across multiple heterogeneous criteria. Experiments show that our proposed three shared-private models are effective to extract the shared information, and achieve significant improvements over the single-criterion methods.

Acknowledgments

We appreciate the contribution from Jingjing Gong and Jiacheng Xu. Besides, we would like to thank the anonymous reviewers for their valuable comments. This work is partially funded by National Natural Science Foundation of China (No. 61532011 and 61672162), Shanghai Municipal Science and Technology Commission on (No. 16JC1420401).

References

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. 2014. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446* .

S. Ben-David and R. Schuller. 2003. Exploiting task relatedness for multiple task learning. *Learning Theory and Kernel Machines* pages 567–580.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*. pages 343–351.

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

Jiayuan Chao, Zhenghua Li, Wenliang Chen, and Min Zhang. 2015. Exploiting heterogeneous annotations for weibo word segmentation and pos tagging. In *National CCF Conference on Natural Language Processing and Chinese Computing*. Springer, pages 495–506.

Hongshen Chen, Yue Zhang, and Qun Liu. 2016. Neural network for heterogeneous annotations. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* .

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics.*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*. pages 1197–1206.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*. volume 133.

XIA Fei. 2000. The part-of-speech tagging guidelines for the penn chinese treebank (3.0). *URL: http://www. cis. upenn. edu/˜ chinese/segguide. 3rd. ch. pdf* .

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research* 17(59):1–35.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. pages 2672–2680.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

W. Jiang, L. Huang, and Q. Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and POS tagging: a case study. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing*. pages 522–530.

G. Jin and X. Chen. 2008. The fourth international chinese language processing bakeoff: Chinese word segmentation, named entity recognition and chinese pos tagging. In *Sixth SIGHAN Workshop on Chinese Language Processing*. page 69.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of The 32nd International Conference on Machine Learning*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Zhenghua Li, Jiayuan Chao, Min Zhang, and Wenliang Chen. 2015. Coupled sequence labeling on heterogeneous annotations: Pos tagging as a case study. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Zhenghua Li, Jiayuan Chao, Min Zhang, and Jiwen Yang. 2016. Fast coupled sequence labeling on heterogeneous annotations via context-aware pruning. In *Proceedings of EMNLP*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016a. Deep multi-task learning with shared memory. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016b. Recurrent neural network for text classification with multi-task learning. In *Proceedings of International Joint Conference on Artificial Intelligence*.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Wenzhe Pei, Tao Ge, and Chang Baobao. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of ACL*.

Xipeng Qiu, Peng Qian, and Zhan Shi. 2016. Overview of the NLPCC-ICCPOL 2016 shared task: Chinese word segmentation for micro-blog texts. In *International Conference on Computer Processing of Oriental Languages*. Springer, pages 901–906.

Xipeng Qiu, Jiayi Zhao, and Xuanjing Huang. 2013. Joint chinese word segmentation and pos tagging on heterogeneous annotated corpora with multiple task learning. In *EMNLP*. pages 658–668.

Weiwei Sun and Xiaojun Wan. 2012. Reducing approximation and estimation errors for chinese lexical processing with heterogeneous annotations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. pages 232–241.

S. Yu, J. Lu, X. Zhu, H. Duan, S. Kang, H. Sun, H. Wang, Q. Zhao, and W. Zhan. 2001. Processing norms of modern Chinese corpus. Technical report, Technical report.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*. pages 647–657.

Neural Joint Model for Transition-based Chinese Syntactic Analysis

Shuhei Kurita **Daisuke Kawahara** **Sadao Kurohashi**
Graduate School of Informatics, Kyoto University
{kurita, dk, kuro}@nlp.ist.i.kyoto-u.ac.jp

Abstract

We present neural network-based joint models for Chinese word segmentation, POS tagging and dependency parsing. Our models are the first neural approaches for fully joint Chinese analysis that is known to prevent the error propagation problem of pipeline models. Although word embeddings play a key role in dependency parsing, they cannot be applied directly to the joint task in the previous work. To address this problem, we propose embeddings of character strings, in addition to words. Experiments show that our models outperform existing systems in Chinese word segmentation and POS tagging, and perform preferable accuracies in dependency parsing. We also explore bi-LSTM models with fewer features.

1 Introduction

Dependency parsers have been enhanced by the use of neural networks and embedding vectors (Chen and Manning, 2014; Weiss et al., 2015; Zhou et al., 2015; Alberti et al., 2015; Andor et al., 2016; Dyer et al., 2015). When these dependency parsers process sentences in English and other languages that use symbols for word separations, they can be very accurate. However, for languages that do not contain word separation symbols, dependency parsers are used in pipeline processes with word segmentation and POS tagging models, and encounter serious problems because of error propagations. In particular, Chinese word segmentation is notoriously difficult because sentences are written without word dividers and Chinese words are not clearly defined. Hence, the pipeline of word segmentation, POS tagging and dependency parsing always suffers from word segmentation errors. Once words have been wrongly-segmented, word embeddings and traditional one-hot word features, used in dependency parsers, will mistake the precise meanings of the original sentences. As a result, pipeline models achieve dependency scores of around 80% for Chinese.

A traditional solution to this error propagation problem is to use joint models. Many Chinese words play multiple grammatical roles with only one grammatical form. Therefore, determining the word boundaries and the subsequent tagging and dependency parsing are closely correlated. Transition-based joint models for Chinese word segmentation, POS tagging and dependency parsing are proposed by Hatori et al. (2012) and Zhang et al. (2014). Hatori et al. (2012) state that dependency information improves the performances of word segmentation and POS tagging, and develop the first transition-based joint word segmentation, POS tagging and dependency parsing model. Zhang et al. (2014) expand this and find that both the inter-word dependencies and intra-word dependencies are helpful in word segmentation and POS tagging.

Although the models of Hatori et al. (2012) and Zhang et al. (2014) perform better than pipeline models, they rely on the one-hot representation of characters and words, and do not assume the similarities among characters and words. In addition, not only words and characters but also many incomplete tokens appear in the transition-based joint parsing process. Such incomplete or unknown words (UNK) could become important cues for parsing, but they are not listed in dictionaries or pre-trained word embeddings. Some recent studies show that character-based embeddings are effective in neural parsing (Ballesteros et al., 2015; Zheng et al., 2015), but their models could not be directly applied to joint models because they use given word segmentations. To solve

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1204–1214
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1111

these problems, we propose neural network-based joint models for word segmentation, POS tagging and dependency parsing. We use both character and word embeddings for known tokens and apply character string embeddings for unknown tokens.

Another problem in the models of Hatori et al. (2012) and Zhang et al. (2014) is that they rely on detailed feature engineering. Recently, bidirectional LSTM (bi-LSTM) based neural network models with very few feature extraction are proposed (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016). In their models, the bi-LSTM is used to represent the tokens including their context. Indeed, such neural networks can observe whole sentence through the bi-LSTM. This bi-LSTM is similar to that of neural machine translation models of Bahdanau et al. (2014). As a result, Kiperwasser and Goldberg (2016) achieve competitive scores with the previous state-of-the-art models. We also develop joint models with n-gram character string bi-LSTM.

In the experiments, we obtain state-of-the-art Chinese word segmentation and POS tagging scores, and the pipeline of the dependency model achieves the better dependency scores than the previous joint models. To the best of our knowledge, this is the first model to use embeddings and neural networks for Chinese full joint parsing.

Our contributions are summarized as follows: (1) we propose the first embedding-based fully joint parsing model, (2) we use character string embeddings for UNK and incomplete tokens. (3) we also explore bi-LSTM models to avoid the detailed feature engineering in previous approaches. (4) in experiments using Chinese corpus, we achieve state-of-the-art scores in word segmentation, POS tagging and dependency parsing.

2 Model

All full joint parsing models we present in this paper use the transition-based algorithm in Section 2.1 and the embeddings of character strings in Section 2.2. We present two neural networks: the feed-forward neural network models in Section 2.3 and the bi-LSTM models in Section 2.4.

2.1 Transition-based Algorithm for Joint Segmentation, POS Tagging, and Dependency Parsing

Based on Hatori et al. (2012), we use a modified arc-standard algorithm for character transitions (Figure 1). The model consists of one buffer and one stack. The buffer contains characters in the input sentence, and the stack contains words shifted from the buffer. The stack words may have their child nodes. The words in the stack are formed by the following transition operations.

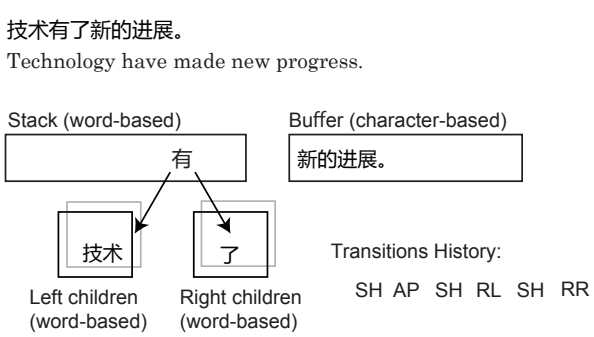


Figure 1: Transition-based Chinese joint model for word segmentation, POS tagging and dependency parsing.

- `SH(t)` (*shift*): Shift the first character of the buffer to the top of the stack as a new word.
- `AP` (*append*): Append the first character of the buffer to the end of the top word of the stack.
- `RR` (*reduce-right*): Reduce the right word of the top two words of the stack, and make the right child node of the left word.
- `RL` (*reduce-left*): Reduce the left word of the top two words of the stack, and make the left child node of the right word.

The `RR` and `RL` operations are the same as those of the arc-standard algorithm (Nivre, 2004a). `SH` makes a new word whereas `AP` makes the current word longer by adding one character. The POS tags are attached with the `SH(t)` transition.

In this paper, we explore both greedy models and beam decoding models. This parsing algorithm works in both types. We also develop a joint model of word segmentation and POS tagging, along with a dependency parsing model. The joint model of word segmentation and POS tagging does not have `RR` and `RL` transitions.

2.2 Embeddings of Character Strings

First, we explain the embeddings used in the neural networks. Later, we explain details of the neural networks in Section 2.3 and 2.4.

Both meaningful words and incomplete tokens appear during transition-based joint parsing. Although embeddings of incomplete tokens are not used in previous work, they could become useful features in several cases. For example, "南京东路" (Nanjing East Road, the famous shopping street of Shanghai) is treated as a single Chinese word in the Penn Chinese Treebank (CTB) corpus. There are other named entities of this form in CTB, e.g, "北京西路" (Beijing West Road) and "湘西路" (Hunan West Road). In these cases, "南京" (Nanjing) and "北京" (Beijing) are location words, while "东路" (East Road) and "西路" (West Road) are sub-words. "东路" and "西路" are similar in terms of their character composition and usage, which is not sufficiently considered in the previous work. Moreover, representations of incomplete tokens are helpful for compensating the segmentation ambiguity. Suppose that the parser makes over-segmentation errors and segments "南京东路" to "南京" and "东路". In this case, "东路" becomes UNK. However, the models could infer that "东路" is also a location, from its character composition and neighboring words. This could give models robustness of segmentation errors. In our models, we prepare the word and character embeddings in the pre-training. We also use the embeddings of character strings for sub-words and UNK which are not in the pre-trained embeddings.

The characters and words are embedded in the same vector space during pre-training. We prepare the same training corpus with the segmented word files and the segmented character files. Both files are concatenated and learned by word2vec (Mikolov et al., 2013). We use the embeddings of 1M frequent words and characters. Words and characters that are in the training set and do not have pre-trained embeddings are given randomly initialized embeddings. The development set and the test set have out-of-vocabulary (OOV) tokens for these embeddings.

The embeddings of the unknown character strings are generated in the neural computation graph when they are required. Consider a character string $c_1 c_2 \cdots c_n$ consisting of characters c_i. When this character string is not in the pre-trained embeddings, the model obtains the embeddings $\mathbf{v}(c_1 c_2 \cdots c_n)$ by the mean of each character embeddings $\sum_{i=1}^{n} \mathbf{v}(c_i)$. Embeddings of words, characters and character strings have the same dimension and are chosen in the neural computation graph. We avoid using the "UNK" vector as far as possible, because this degenerates the information about unknown tokens. However, models use the "UNK" vector if the parser encounters characters that are not in the pre-trained embeddings, though this is quite uncommon.

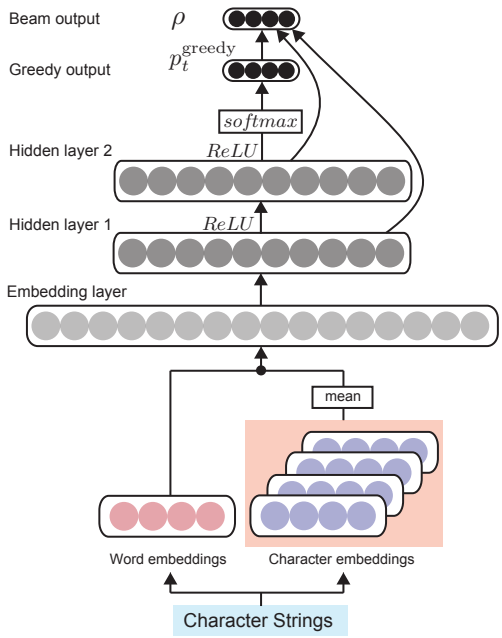


Figure 2: The feed-forward neural network model. The greedy output is obtained at the second top layer, while the beam decoding output is obtained at the top layer. The input character strings are translated into word embeddings if the embeddings of the character strings are available. Otherwise, the embeddings of the character strings are used.

2.3 Feed-forward Neural Network

2.3.1 Neural Network

We present a feed-forward neural network model in Figure 2. The neural network for greedy training is based on the neural networks of Chen and Manning (2014) and Weiss et al. (2015). We add the dynamic generation of the embeddings of character strings for unknown tokens, as described in Section 2.2. This neural network has two hidden layers with 8,000 dimensions. This is larger than Chen and Manning (2014) (200 dimensions) or Weiss et al. (2015) (1,024 or 2,048 dimensions). We use the ReLU for the activation function of the hidden layers (Nair and Hinton, 2010) and the softmax function for the output layer of the greedy

Type	Value
Size of $\mathbf{h_1}$,$\mathbf{h_2}$	8,000
Initial learning rate	0.01
Initial learning rate of beam decoding	0.001
Embedding vocabulary size	1M
Embedding vector size	200
Small embedding vector size	20
Minibatch size	200

Table 1: Parameters for neural network structure and training.

neural network. There are three randomly initialized weight matrices between the embedding layers and the softmax function. The loss function $L(\theta)$ for the greedy training is

$$L(\theta) = -\sum_{s,t} \log p_{s,t}^{\text{greedy}} + \frac{\lambda}{2}||\theta||^2,$$

$$p_{s,t}^{\text{greedy}}(\boldsymbol{\beta}) \propto \exp\left(\sum_j w_{tj}\beta_j + b_t\right),$$

where t denotes one transition among the transition set $\mathcal{T}$ ($t \in \mathcal{T}$). s denotes one element of the single mini-batch. $\boldsymbol{\beta}$ denotes the output of the previous layer. $\mathbf{w}$ and $\mathbf{b}$ denote the weight matrix and the bias term. θ contains all parameters. We use the $L2$ penalty term and the Dropout. The backprop is performed including the word and character embeddings. We use Adagrad (Duchi et al., 2010) to optimize learning rate. We also consider Adam (Kingma and Ba, 2015) and SGD, but find that Adagrad performs better in this model. The other learning parameters are summarized in Table 1.

In our model implementation, we divide all sentences into training batches. Sentences in the same training batches are simultaneously processed by the neural mini-batches. By doing so, the model can parse all sentences of the training batch in the number of transitions required to parse the longest sentence in the batch. This allows the model to parse more sentences at once, as long as the neural mini-batch can be allocated to the GPU memory. This can be applied to beam decoding.

2.3.2 Features

The features of this neural network are listed in Table 2. We use three kinds of features: (1) features obtained from Hatori et al. (2012) by removing combinations of features, (2) features obtained from Chen and Manning (2014), (3) original features related to character strings. In particular, the original features include sub-words, character strings across the buffer and the stack, and character strings in the buffer. Character strings across the buffer and stack could capture the currently-segmented word. To avoid using character strings that are too long, we restrict the length of character string to a maximum of four characters. Unlike Hatori et al. (2012), we use sequential characters of sentences for features, and avoid hand-engineered combinations among one-hot features, because such combinations could be automatically generated in the neural hidden layers as distributed representations (Hinton et al., 1986).

Type	Features
Stack word and tags	s0w, s1w, s2w
	s0p, s1p, s2p
Stack 1 children and tags	s0l0w, s0r0w, s0l1w, s0r1w
	s0l0p, s0r0p, s0l1p, s0r1p
Stack 2 children	s1l0w, s1r0w, s1l1w, s1r1w
Children of children	s0l0lw, s0r0rw, s1l0lw, s1r0rw
Buffer characters	b0c, b1c, b2c, b3c
Previously shifted words	q0w, q1w
Previously shifted tags	q0p, q1p
Character of q0	q0e
Parts of q0 word	q0f1, q0f2, q0f3
Strings across q0 and buf.	q0b1, q0b2, q0b3
Strings of buffer characters	b0-2, b0-3, b0-4
	b1-3, b1-4, b1-5
	b2-4, b2-5, b2-6
	b3-5, b3-6
	b4-6
Length of q0	lenq0

Table 2: Features for the joint model. "q0" denotes the last shifted word and "q1" denotes the word shifted before "q0". In "part of q0 word", "f1", "f2" and "f3" denote sub-words of "q0", which are 1, 2 and 3 sequential characters including the last character of "q0" respectively. In "strings across q0 and buf.", "q0bX" denotes "q0" and X sequential characters of the buffer. This feature could capture words that boundaries have not determined yet. In "strings of buffer characters", "bX-Y" denotes sequential characters from the X-th to Y-th character of the buffer. The suffix "e" denotes the end character of the word. The dimension of the embedding of "length of q0" is 20.

In the later section, we evaluate a joint model for word segmentation and POS tagging. This model does not use the children and children-of-children of stack words as features.

2.3.3 Beam Search

Structured learning plays an important role in previous joint parsing models for Chinese.[1] In this paper, we use the structured learning model proposed by Weiss et al. (2015) and Andor et al. (2016).

In Figure 2, the output layer for the beam decoding is at the top of the network. There are a perceptron layer which has inputs from the two hidden layers and the greedy output layer: $[\mathbf{h}_1, \mathbf{h}_2, \mathbf{p}^{\text{greedy}}(\mathbf{y})]$. This layer is learned by the following cost function (Andor et al., 2016):

$$L(d^*_{1:j};\theta) = -\sum_{i=1}^{j}\rho(d^*_{1:i-1}, d^*_i;\theta) + \ln \sum_{d'_{1:j}\in\mathcal{B}_{1:j}} \exp \sum_{i=1}^{j}\rho(d'_{1:i-1}, d'_i;\theta),$$

where $d_{1:j}$ denotes the transition path and $d^*_{1:j}$ denotes the gold transition path. $\mathcal{B}_{1:j}$ is the set of transition paths from 1 to j step in beam. ρ is the value of the top layer in Figure 2. This training can be applied throughout the network. However, we separately train the last beam layer and the previous greedy network in practice, as in Andor et al. (2016). First, we train the last perceptron layer using the beam cost function freezing the previous greedy-trained layers. After the last layer has been well trained, backprop is performed including the previous layers. We notice that training the embedding layer at this stage could make the results worse, and thus we exclude it. Note that this whole network backprop requires considerable GPU memory. Hence, we exclude particularly large batches from the training, because they cannot be on GPU memory. We use multiple beam sizes for training because models can be trained faster with small beam sizes. After the small beam size training, we use larger beam sizes. The test of this fully joint model takes place with a beam size of 16.

Hatori et al. (2012) use special alignment steps in beam decoding. The `AP` transition has size-2 steps, whereas the other transitions have a size-1 step. Using this alignment, the total number of steps for an N-character sentence is guaranteed to be $2N-1$ (excluding the root arc) for any transition path. This can be interpreted as the `AP` transition doing two things: appending characters and resolving intra-word dependencies. This alignment stepping assumes that the intra-word dependencies of characters to the right of the characters exist in each Chinese word.

[1] Hatori et al. (2012) report that structured learning with a beam size of 64 is optimal.

2.4 Bi-LSTM Model

In Section 2.3, we describe a neural network model with feature extraction. Unfortunately, although this model is fast and very accurate, it has two problems: (1) the neural network cannot see the whole sentence information. (2) it relies on feature engineering. To solve these problems, Kiperwasser and Goldberg (2016) propose a bi-LSTM neural network parsing model. Surprisingly, their model uses very few features, and bi-LSTM is applied to represent the context of the features. Their neural network consists of three parts: bi-LSTM, a feature extraction function and a multilayer perceptron (MLP). First, all tokens in the sentences are converted to embeddings. Second, the bi-LSTM reads all embeddings of the sentence. Third, the feature function extracts the feature representations of tokens from the bi-LSTM layer. Finally, an MLP with one hidden layer outputs the transition scores of the transition-based parser.

In this paper, we propose a Chinese joint parsing model with simple and global features using n-gram bi-LSTM and a simple feature extraction function. The model is described in Figure 3. We consider that Chinese sentences consist of tokens, including words, UNKs and incomplete tokens, which can have some meanings and are useful for parsing. Such tokens appear in many parts of the sentence and have arbitrary lengths. To capture them, we propose the n-gram bi-LSTM. The n-gram bi-LSTM read through characters $c_i \cdots c_{i+n-1}$ of the sentence (c_i is the i-th character). For example, the 1-gram bi-LSTM reads each character, and the 2-gram bi-LSTM reads two consecutive characters $c_i c_{i+1}$. After the n-gram forward LSTM reads character string $c_i \cdots c_{i+n-1}$, it next reads $c_{i+1} \cdots c_{i+n}$. The backward LSTM reads from $c_{i+1} \cdots c_{i+n}$ toward $c_i \cdots c_{i+n-1}$. This allows models to capture any n-gram character strings in the input sentence.[2] All n-gram inputs to bi-LSTM are given by the embeddings of words and characters or the dynamically generated embeddings of character strings, as described in

[2] At the end of the sentence of length N, character strings $c_i \cdots c_N (N < i+n-1)$, which are shorter than n characters, are used.

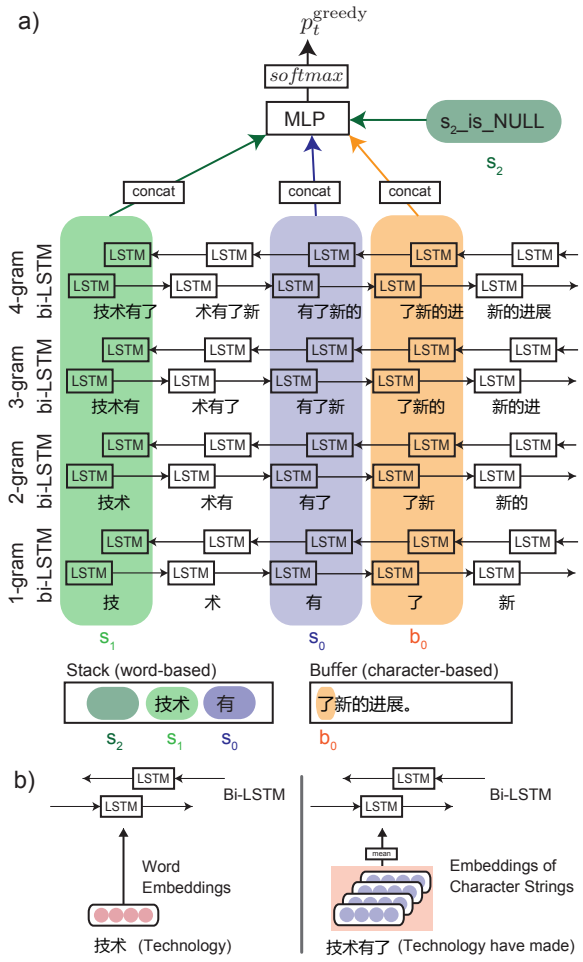


Figure 3: The bi-LSTM model. (a): The Chinese sentence "技术有了新的进展。" has been processed. (b): Similar to the feed-forward neural network model, the embeddings of words, characters and character strings are used. In this figure, a word "技术"(technology) has its embedding, while a token "技术有了"(technology have made) does not.

Section 2.2. Although these arbitrary n-gram tokens produce UNKs, character string embeddings can capture similarities among them. Following the bi-LSTM layer, the feature function extracts the corresponding outputs of the bi-LSTM layer. We summarize the features in Table 3. Finally, MLP and the softmax function outputs the transition probability. We use an MLP with three hidden layers as for the model in Section 2.3. We train this neural network with the loss function for the greedy training.

Model	Features
4 features	s0w, s1w, s2w, b0c
8 features	s0w, s1w, s2w, b0c s0r0w, s0l0w, s1r0w, s1l0w

Table 3: Features for the bi-LSTM models. All features are words and characters. We experiment both four and eight features models.

		#snt	#oov
CTB-5	Train	18k	-
	Dev.	350	553
	Test	348	278
CTB-7	Train	31k	-
	Dev.	10k	13k
	Test	10k	13k

Table 4: Summary of datasets.

3 Experiments

3.1 Experimental Settings

We use the Penn Chinese Treebank 5.1 (CTB-5) and 7 (CTB-7) datasets to evaluate our models, following the splitting of Jiang et al. (2008) for CTB-5 and Wang et al. (2011) for CTB-7. The statistics of datasets are presented in Table 4. We use the Chinese Gigaword Corpus for embedding pre-training. Our model is developed for unlabeled dependencies. The development set is used for parameter tuning. Following Hatori et al. (2012) and Zhang et al. (2014), we use the standard word-level evaluation with F1-measure. The POS tags and dependencies cannot be correct unless the corresponding words are correctly segmented.

We trained three models: SegTag, SegTagDep and Dep. SegTag is the joint word segmentation and POS tagging model. SegTagDep is the full joint segmentation, tagging and dependency parsing model. Dep is the dependency parsing model which is similar to Weiss et al. (2015) and Andor et al. (2016), but uses the embeddings of character strings. Dep compensates for UNKs and segmentation errors caused by previous word segmentation using embeddings of character strings. We will examine this effect later.

Most experiments are conducted on GPUs, but some of beam decoding processes are performed on CPUs because of the large mini-batch size. The neural network is implemented with Theano.

Model	Seg	POS
Hatori+12 SegTag	97.66	93.61
Hatori+12 SegTag(d)	98.18	94.08
Hatori+12 SegTagDep	97.73	94.46
Hatori+12 SegTagDep(d)	98.26	94.64
M. Zhang+14 EAG	97.76	94.36
Y. Zhang+15	98.04	94.47
SegTag(g)	98.41	**94.84**
SegTag	**98.60**	94.76

Table 5: Joint segmentation and POS tagging scores. Both scores are in F-measure. In Hatori et al. (2012), (d) denotes the use of dictionaries. (g) denotes greedy trained models. All scores for previous models are taken from Hatori et al. (2012), Zhang et al. (2014) and Zhang et al. (2015).

3.2 Results

3.2.1 Joint Segmentation and POS Tagging

First, we evaluate the joint segmentation and POS tagging model (SegTag). Table 5 compares the performance of segmentation and POS tagging using the CTB-5 dataset. We train two modles: a greedy-trained model and a model trained with beams of size 4. We compare our model to three previous approaches: Hatori et al. (2012), Zhang et al. (2014) and Zhang et al. (2015). Our SegTag joint model is superior to these previous models, including Hatori et al. (2012)'s model with rich dictionary information, in terms of both segmentation and POS tagging accuracy.

3.2.2 Joint Segmentation, POS Tagging and Dependency Parsing

Table 6 presents the results of our full joint model. We employ the greedy trained full joint model SegTagDep(g) and the beam decoding model SegTagDep. All scores for the existing models in this table are taken from Zhang et al. (2014). Though our model surpasses the previous best end-to-end joint models in terms of segmentation and POS tagging, the dependency score is slightly lower than the previous models. The greedy model SegTagDep(g) achieves slightly lower scores than beam models, although this model works considerably fast because it does not use beam decoding.

Model	Seg	POS	Dep
Hatori+12	97.75	94.33	81.56
M. Zhang+14 EAG	97.76	94.36	**81.70**
SegTagDep(g)	98.24	94.49	80.15
SegTagDep	**98.37**	**94.83**	81.42

Table 6: Joint Segmentation, POS Tagging and Dependency Parsing. Hatori et al. (2012)'s CTB-5 scores are reported in Zhang et al. (2014). EAG in Zhang et al. (2014) denotes the arc-eager model. (g) denotes greedy trained models.

Model	Seg	POS	Dep
Hatori+12	97.75	94.33	81.56
M. Zhang+14 STD	97.67	94.28	81.63
M. Zhang+14 EAG	97.76	94.36	81.70
Y. Zhang+15	98.04	94.47	82.01
SegTagDep(g)	98.24	94.49	80.15
SegTagDep	98.37	**94.83**‡	81.42‡
SegTag+Dep	**98.60**‡	94.76‡	**82.60**‡

Table 7: The SegTag+Dep model. Note that the model of Zhang et al. (2015) requires other base parsers. ‡ denotes that the improvement is statistically siginificant at $p < 0.01$ compared with SegTagDep(g) using paired t-test.

3.2.3 Pipeline of Our Joint SegTag and Dep Model

We use our joint SegTag model for the pipeline input of the Dep model (SegTag+Dep). Both SegTag and Dep models are trained and tested by the beam cost function with beams of size 4. Table 7 presents the results. Our SegTag+Dep model performs best in terms of the dependency and word segmentation. The SegTag+Dep model is better than the full joint model. This is because most segmentation errors of these models occur around named entities. Hatori et al. (2012)'s alignment step assumes the intra-word dependencies in words, while named entities do not always have them. For example, SegTag+Dep model treats named entity "海赛克", a company name, as one word, while the SegTagDep model divides this to "海" (sea) and "赛克", where "赛克" could be used for foreigner's name. For such words, SegTagDep prefers `SH` because `AP` has size-2 step of the character appending and intra-word dependency resolution, which does not exist for named entities. This problem could be solved by adding a special transition `AP_named_entity` which is similar to `AP` but with size-1 step and used

Model	Dep
Dep(g)-cs	80.51
Dep(g)	80.98

Table 8: SegTag+Dep(g) model with and without character strings (cs) representations. Note that we compare these models with greedy training for simplicity's sake.

Model	Seg	POS	Dep
SegTagDep(g) -q0bX	97.81	93.79	79.16
SegTagDep(g)	98.24	94.49	80.15

Table 9: SegTagDep model with and without (-q0bX) features across the buffer and stack. We compare these models with greedy training (g).

Model	Seg	POS	Dep
Hatori+12	95.42	90.62	73.58
M. Zhang+14 STD	95.53	90.75	**75.63**
SegTagDep(g)	96.06	90.28	73.98
SegTagDep	95.86	90.91‡	74.04
SegTag+Dep	**96.23**‡	**91.25**‡	75.28‡

Table 10: Results from SegTag+Dep and SegTagDep applied to the CTB-7 corpus. (g) denotes greedy trained models. ‡ denotes that the improvement is statistically siginificant at $p < 0.01$ compared with SegTagDep(g) using paired t-test.

only for named entities. Additionally, Zhang et al. (2014)'s STD (arc-standard) model works slightly better than Hatori et al. (2012)'s fully joint model in terms of the dependency score. Zhang et al. (2014)'s STD model is similar to our SegTag+Dep because they combine a word segmentator and a dependency parser using "deque" of words.

3.2.4 Effect of Character String Embeddings

Finally, we compare the two pipeline models of SegTag+Dep to show the effectiveness of using character string representations instead of "UNK" embeddings. We use two dependency models with greedy training: Dep(g) for dependency model and Dep(g)-cs for dependency model without the character string embeddings . In the Dep(g)-cs model, we use the "UNK" embedding when the embeddings of the input features are unavailable, whereas we use the character string embeddings in model Dep(g). The results are presented in Table 8. When the models encounter unknown tokens, using the embeddings of character strings is better than using the "UNK" embedding.

3.2.5 Effect of Features across the Buffer and Stack

We test the effect of special features: q0bX in Table 2. The q0bX features capture the tokens across the buffer and stack. Joint transition-based parsing models by Hatori et al. (2012) and Chen and Manning (2014) decide POS tags of words before corresponding word segmentations are determined. In our model, the q0bX features capture words even if their segmentations are not determined. We examine the effectiveness of these features by training greedy full joint models with and without them. The results are shown in Table 9. The q0bX features boost not only POS tagging scores but also word segmentation scores.

3.2.6 CTB-7 Experiments

We also test the SegTagDep and SegTag+Dep models on CTB-7. In these experiments, we notice that the MLP with four hidden layers performs better than the MLP with three hidden layers, but we could not find definite differences in the experiments in CTB-5. We speculate that this is caused by the difference in the training set size. We present the final results with four hidden layers in Table 10.

3.2.7 Bi-LSTM Model

We experiment the n-gram bi-LSTMs models with four and eight features listed in Table 3. We summarize the result in Table 11. The greedy bi-LSTM models perform slightly worse than the previous models, but they do not rely on feature engineering.

4 Related Work

Zhang and Clark (2008) propose an incremental joint word segmentation and POS tagging model driven by a single perceptron. Zhang and Clark (2010) improve this model by using both character and word-based decoding. Hatori et al. (2011) propose a transition-based joint POS tagging and dependency parsing model. Zhang et al. (2013) propose a joint model using character structures of words for constituency parsing. Wang et al. (2013) also propose a lattice-based joint model for constituency parsing. Zhang et al. (2015) propose joint segmentation, POS tagging and dependency re-ranking system. This system requires

Model	Seg	POS	Dep
Hatori+12	97.75	94.33	81.56
M. Zhang+14 EAG	97.76	94.36	81.70
SegTagDep (g)	98.24	94.49	80.15
Bi-LSTM 4feat.(g)	97.72	93.12	79.03
Bi-LSTM 8feat.(g)	97.70	93.37	79.38

Table 11: Bi-LSTM feature extraction model. "4feat." and "8feat." denote the use of four and eight features.

base parsers. In neural joint models, Zheng et al. (2013) propose a neural network-based Chinese word segmentation model based on tag inferences. They extend their models for joint segmentation and POS tagging. Zhu et al. (2015) propose the re-ranking system of parsing results with recursive convolutional neural network.

5 Conclusion

We propose the joint parsing models by the feed-forward and bi-LSTM neural networks. Both of them use the character string embeddings. The character string embeddings help to capture the similarities of incomplete tokens. We also explore the neural network with few features using n-gram bi-LSTMs. Our SegTagDep joint model achieves better scores of Chinese word segmentation and POS tagging than previous joint models, and our SegTag and Dep pipeline model achieves state-of-the-art score of dependency parsing. The bi-LSTM models reduce the cost of feature engineering.

References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1354–1359. http://aclweb.org/anthology/D15-1159.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2442–2452. http://www.aclweb.org/anthology/P16-1231.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 349–359. http://aclweb.org/anthology/D15-1041.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 740–750. http://www.aclweb.org/anthology/D14-1082.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 32–37. http://anthology.aclweb.org/P16-2006.

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. UCB/EECS-2010-24.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 334–343. http://www.aclweb.org/anthology/P15-1033.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, pages 1216–1224. http://www.aclweb.org/anthology/I11-1136.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1045–1053. http://www.aclweb.org/anthology/P12-1110.

Geoffrey E. Hinton, J. L. McClelland, and D. E. Rumelhart. 1986. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*. pages Vol.1, p.12.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, pages 897–904.

D. P. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. volume abs/1301.3781. http://arxiv.org/abs/1301.3781.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. pages 807–814.

Joakim Nivre. 2004a. Incrementality in deterministic dependency parsing. In Frank Keller, Stephen Clark, Matthew Crocker, and Mark Steedman, editors, *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics, pages 50–57.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, pages 309–317. http://www.aclweb.org/anthology/I11-1035.

Zhiguo Wang, Chengqing Zong, and Nianwen Xue. 2013. A lattice-based framework for joint chinese word segmentation, pos tagging and parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 623–627. http://www.aclweb.org/anthology/P13-2110.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 323–333. http://www.aclweb.org/anthology/P15-1032.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese parsing exploiting characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 125–134. http://www.aclweb.org/anthology/P13-1013.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1326–1336. http://www.aclweb.org/anthology/P14-1125.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, pos tagging and dependency parsing. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligenc*. Association for Computational Linguistics, pages 42–52. http://www.aclweb.org/anthology/N15-1005.

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

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 843–852. http://www.aclweb.org/anthology/D10-1082.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 647–657. http://www.aclweb.org/anthology/D13-1061.

Xiaoqing Zheng, Haoyuan Peng, Yi Chen, Pengjing Zhang, and Zhang Wenqiang. 2015. Character-based parsing with convolutional neural network. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. page 153.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*

and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Association for Computational Linguistics, pages 1213–1222. http://www.aclweb.org/anthology/P15-1117.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1159–1168. http://www.aclweb.org/anthology/P15-1112.

Robust Incremental Neural Semantic Graph Parsing

Jan Buys[1] **and Phil Blunsom**[1,2]
[1]Department of Computer Science, University of Oxford [2]DeepMind
{jan.buys,phil.blunsom}@cs.ox.ac.uk

Abstract

Parsing sentences to linguistically-expressive semantic representations is a key goal of Natural Language Processing. Yet statistical parsing has focussed almost exclusively on bilexical dependencies or domain-specific logical forms. We propose a neural encoder-decoder transition-based parser which is the first full-coverage semantic graph parser for Minimal Recursion Semantics (MRS). The model architecture uses stack-based embedding features, predicting graphs jointly with unlexicalized predicates and their token alignments. Our parser is more accurate than attention-based baselines on MRS, and on an additional Abstract Meaning Representation (AMR) benchmark, and GPU batch processing makes it an order of magnitude faster than a high-precision grammar-based parser. Further, the 86.69% Smatch score of our MRS parser is higher than the upper-bound on AMR parsing, making MRS an attractive choice as a semantic representation.

1 Introduction

An important goal of Natural Language Understanding (NLU) is to parse sentences to structured, interpretable meaning representations that can be used for query execution, inference and reasoning. Recently end-to-end models have outperformed traditional pipeline approaches, predicting syntactic or semantic structure as intermediate steps, on NLU tasks such as sentiment analysis and semantic relatedness (Le and Mikolov, 2014; Kiros et al., 2015), question answering (Hermann et al., 2015) and textual entailment (Rocktäschel et al., 2015). However the linguistic structure used in applications has predominantly been shallow, restricted to bilexical dependencies or trees.

In this paper we focus on robust parsing into linguistically deep representations. The main representation that we use is Minimal Recursion Semantics (MRS) (Copestake et al., 1995, 2005), which serves as the semantic representation of the English Resource Grammar (ERG) (Flickinger, 2000). Existing parsers for full MRS (as opposed to bilexical semantic graphs derived from, but simplifying MRS) are grammar-based, performing disambiguation with a maximum entropy model (Toutanova et al., 2005; Zhang et al., 2007); this approach has high precision but incomplete coverage.

Our main contribution is to develop a fast and robust parser for full MRS-based semantic graphs. We exploit the power of global conditioning enabled by deep learning to predict linguistically deep graphs incrementally. The model does not have access to the underlying ERG or syntactic structures from which the MRS analyses were originally derived. We develop parsers for two graph-based conversions of MRS, Elementary Dependency Structure (EDS) (Oepen and Lønning, 2006) and Dependency MRS (DMRS) (Copestake, 2009), of which the latter is inter-convertible with MRS.

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a graph-based semantic representation that shares the goals of MRS. Aside from differences in the choice of which linguistic phenomena are annotated, MRS is a compositional representation explicitly coupled with the syntactic structure of the sentence, while AMR does not assume compositionality or alignment with the sentence structure. Recently a number of AMR parsers have been developed (Flanigan et al., 2014; Wang et al., 2015b; Artzi et al., 2015;

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1215–1226
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1112

Damonte et al., 2017), but corpora are still under active development and low inter-annotator agreement places on upper bound of 83% F1 on expected parser performance (Banarescu et al., 2013). We apply our model to AMR parsing by introducing structure that is present explicitly in MRS but not in AMR (Buys and Blunsom, 2017).

Parsers based on RNNs have achieved state-of-the-art performance for dependency parsing (Dyer et al., 2015; Kiperwasser and Goldberg, 2016) and constituency parsing (Vinyals et al., 2015b; Dyer et al., 2016; Cross and Huang, 2016b). One of the main reasons for the prevalence of bilexical dependencies and tree-based representations is that they can be parsed with efficient and well-understood algorithms. However, one of the key advantages of deep learning is the ability to make predictions conditioned on unbounded contexts encoded with RNNs; this enables us to predict more complex structures without increasing algorithmic complexity. In this paper we show how to perform linguistically deep parsing with RNNs.

Our parser is based on a transition system for semantic graphs. However, instead of generating arcs over an ordered, fixed set of nodes (the words in the sentence), we generate the nodes and their alignments jointly with the transition actions. We use a graph-based variant of the arc-eager transition-system. The sentence is encoded with a bidirectional RNN. The transition sequence, seen as a graph linearization, can be predicted with any encoder-decoder model, but we show that using hard attention, predicting the alignments with a pointer network and conditioning explicitly on stack-based features improves performance. In order to deal with data sparsity candidate lemmas are predicted as a pre-processing step, so that the RNN decoder predicts unlexicalized node labels.

We evaluate our parser on DMRS, EDS and AMR graphs. We show that our model architecture improves performance from 79.68% to 84.16% F1 over an attention-based encoder-decoder baseline. Although our parser is less accurate that a high-precision grammar-based parser on a test set of sentences parsable by that grammar, incremental prediction and GPU batch processing enables it to parse 529 tokens per second, against 7 tokens per second for the grammar-based parser. On AMR parsing our model obtains 60.11% Smatch, an improvement of 8% over an existing neural AMR parser.

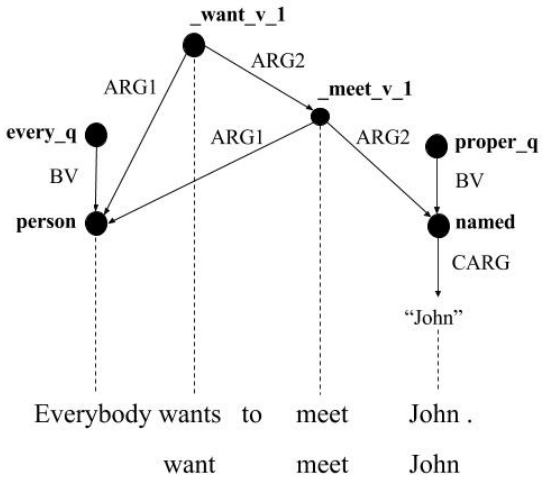


Figure 1: Semantic representation of the sentence "Everybody wants to meet John." The graph is based on the Elementary Dependency Structure (EDS) representation of Minimal Recursion Semantics (MRS). The alignments are given together with the corresponding tokens, and lemmas of surface predicates and constants.

2 Meaning Representations

We define a common framework for semantic graphs in which we can place both MRS-based graph representations (DMRS and EDS) and AMR. Sentence meaning is represented with rooted, labelled, connected, directed graphs (Kuhlmann and Oepen, 2016). An example graph is visualized in Figure 1. representations. Node labels are referred to as *predicates* (*concepts* in AMR) and edge labels as *arguments* (AMR *relations*). In addition *constants*, a special type of node modifiers, are used to denote the string values of named entities and numbers (including date and time expressions). Every node is aligned to a token or a continuous span of tokens in the sentence the graph corresponds to.

Minimal Recursion Semantics (MRS) is a framework for computational semantics that can be used for parsing or generation (Copestake et al., 2005). Instances and eventualities are represented with logical variables. Predicates take arguments with labels from a small, fixed set of roles. Arguments are either logical variables or handles, designated formalism-internal variables. Handle equality constraints support scope underspecification; multiple scope-resolved logical representations can be derived from one MRS structure. A predicate corresponds to its intrinsic argument

and is aligned to a character span of the (untokenized) input sentence. Predicates representing named entities or numbers are parameterized by strings. Quantification is expressed through predicates that bound instance variables, rather than through logical operators such as ∃ or ∀. MRS was designed to be integrated with feature-based grammars such as Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) or Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982). MRS has been implement the English Resource Grammar (ERG) (Flickinger, 2000), a broad-coverage high-precision HPSG grammar.

Oepen and Lønning (2006) proposed Elementary Dependency Structure (EDS), a conversion of MRS to variable-free dependency graphs which drops scope underspecification. Copestake (2009) extended this conversion to avoid information loss, primarily through richer edge labels. The resulting representation, Dependency MRS (DMRS), can be converted back to the original MRS, or used directly in MRS-based applications (Copestake et al., 2016). We are interested in the empirical performance of parsers for both of these representations: while EDS is more interpretable as an independent semantic graph representation, DMRS can be related back to underspecified logical forms. A bilexical simplification of EDS has previously been used for semantic dependency parsing (Oepen et al., 2014, 2015). Figure 1 illustrates an EDS graph.

MRS makes an explicit distinction between surface and abstract predicates (by convention surface predicates are prefixed by an underscore). Surface predicates consist of a lemma followed by a coarse part-of-speech tag and an optional sense label. Predicates absent from the ERG lexicon are represented by their surface forms and POS tags. We convert the character-level predicate spans given by MRS to token-level spans for parsing purposes, but the representation does not require gold tokenization. Surface predicates usually align with the span of the token(s) they represent, while abstract predicates can span longer segments. In full MRS every predicate is annotated with a set of morphosyntactic features, encoding for example tense, aspect and number information; we do not currently model these features.

AMR (Banarescu et al., 2013) graphs can be represented in the same framework, despite a number of linguistic differences with MRS. Some information annotated explicitly in MRS is latent in AMR, including alignments and the distinction between surface (lexical) and abstract concepts. AMR predicates are based on PropBank (Palmer et al., 2005), annotated as lemmas plus sense labels, but they form only a subset of concepts. Other concepts are either English words or special keywords, corresponding to overt lexemes in some cases but not others.

```
:root( <2> _v_1
  :ARG1( <1> person
    :BV-of( <1> every_q ) )
  :ARG2 <4> _v_1
    :ARG1*( <1> person
    :ARG2( <5> named_CARG
      :BV-of ( <5> proper_q ) ) )
```

Figure 2: A top-down linearization of the EDS graph in Figure 1, using unlexicalized predicates.

3 Incremental Graph Parsing

We parse sentences to their meaning representations by incrementally predicting semantic graphs together with their alignments. Let $\mathbf{e} = e_1, e_2, \ldots, e_I$ be a tokenized English sentence, $\mathbf{t} = t_1, t_2, \ldots, t_J$ a sequential representation of its graph derivation and $\mathbf{a} = a_1, a_2, \ldots, a_J$ an alignment sequence consisting of integers in the range $1, \ldots, I$. We model the conditional distribution $p(\mathbf{t}, \mathbf{a}|\mathbf{e})$ which decomposes as

$$\prod_{j=1}^{J} p(a_j|(\mathbf{a}, \mathbf{t})_{1:j-1}, \mathbf{e}) p(t_j|\mathbf{a}_{1:j}, \mathbf{t}_{1:j-1}, \mathbf{e}).$$

We also predict the end-of-span alignments as a seperate sequence $\mathbf{a}^{(\mathbf{e})}$.

3.1 Top-down linearization

We now consider how to linearize the semantic graphs, before defining the neural models to parameterize the parser in section 4. The first approach is to linearize a graph as the pre-order traversal of its spanning tree, starting at a designated root node (see Figure 2). Variants of this approach have been proposed for neural constituency parsing (Vinyals et al., 2015b), logical form prediction (Dong and Lapata, 2016; Jia and Liang, 2016) and AMR parsing (Barzdins and Gosko, 2016; Peng et al., 2017).

In the linearization, labels of edges whose direction are reversed in the spanning tree are marked

by adding `-of`. Edges not included in the spanning tree, referred to as *reentrancies*, are represented with special edges whose dependents are dummy nodes pointing back to the original nodes. Our potentially lossy representation represents these edges by repeating the dependent node labels and alignments, which are recovered heuristically. The alignment does not influence the linearized node ordering.

3.2 Transition-based parsing

Figure 1 shows that the semantic graphs we work with can also be interpreted as dependency graphs, as nodes are aligned to sentence tokens. Transition-based parsing (Nivre, 2008) has been used extensively to predict dependency graphs incrementally. We apply a variant of the arc-eager transition system that has been proposed for graph (as opposed to tree) parsing (Sagae and Tsujii, 2008; Titov et al., 2009; Gómez-Rodríguez and Nivre, 2010) to derive a transition-based parser for deep semantic graphs. In dependency parsing the sentence tokens also act as nodes in the graph, but here we need to generate the nodes incrementally as the transition-system proceeds, conditioning the generation on the given sentence. Damonte et al. (2017) proposed an arc-eager AMR parser, but their transition system is more narrowly restricted to AMR graphs.

The transition system consists of a *stack* of graph nodes being processed and a *buffer*, holding a single node at a time. The main transition actions are *shift*, *reduce*, *left-arc*, *right-arc*. Figure 3 shows an example transition sequence together with the stack and buffer after each step. The shift transition moves the element on the buffer to the top of the stack, and generates a predicate and its alignment as the next node on the buffer. Left-arc and right-arc actions add labeled arcs between the buffer and stack top (for DMRS a transition for undirected arcs is included), but do not change the state of the stack or buffer. Finally, reduce pops the top element from the stack, and predicts its end-of-span alignment (if included in the representation). To predict non-planar arcs, we add another transition, which we call *cross-arc*, which first predicts the stack index of a node which is not on top of the stack, adding an arc between the head of the buffer and that node. Another special transition designates the buffer node as the root.

To derive an oracle for this transition system, it is necessary to determine the order in which the nodes are generated. We consider two approaches. The first ordering is obtained by performing an in-order traversal of the spanning tree, where the node order is determined by the alignment. In the resulting linearization the only non-planar arcs are reentrancies. The second approach lets the ordering be monotone (non-decreasing) with respect to the alignments, while respecting the in-order ordering for nodes with the same alignment. In an arc-eager oracle arcs are added greedily, while a reduce action can either be performed as soon as the stack top node has been connected to all its dependents, or delayed until it has to reduce to allow the correct parse tree to be formed. In our model the oracle delays reduce, where possible, until the end alignment of the stack top node spans the node on the buffer. As the span end alignments often cover phrases that they head (e.g. for quantifiers) this gives a natural interpretation to predicting the span end together with the reduce action.

3.3 Delexicalization and lemma prediction

Each token in MRS annotations is aligned to at most one surface predicate. We decompose surface predicate prediction by predicting candidate lemmas for input tokens, and delexicalized predicates consisting only of sense labels. The full surface predicates are then recovered through the predicted alignments.

We extract a dictionary mapping words to lemmas from the ERG lexicon. Candidate lemmas are predicted using this dictionary, and where no dictionary entry is available with a lemmatizer. The same approach is applied to predict constants, along with additional normalizations such as mapping numbers to digit strings.

We use the Stanford CoreNLP toolkit (Manning et al., 2014) to tokenize and lemmatize sentences, and tag tokens with the Stanford Named Entity Recognizer (Finkel et al., 2005). The tokenization is customized to correspond closely to the ERG tokenization; hyphens are removed pre-processing step. For AMR we use automatic alignments and the graph topology to classify concepts as surface or abstract (Buys and Blunsom, 2017). The lexicon is restricted to Propbank (Palmer et al., 2005) predicates; for other concepts we extract a lexicon from the training data.

Action	Stack	Buffer	Arc added
init(1, person)	[ ]	(1, 1, person)	-
sh(1, every_q)	[(1, 1, person)]	(2, 1, every_q)	-
la(BV)	[(1, 1, person)]	(2, 1, every_q)	(2, BV, 1)
sh(2, _v_1)	[(1, 1, person), (2, 1, every_q)]	(2, 1, _v_1)	-
re	[(1, 1, person)]	(3, 2, _v_1)	-
la(ARG1)	[(1, 1, person)]	(3, 2, _v_1)	(3, ARG1, 1)

Figure 3: Start of the transition sequence for parsing the graph in Figure 1. The transitions are shift (`sh`), reduce (`re`), left arc (`la`) and right arc (`ra`). The action taken at each step is given, along with the state of the stack and buffer after the action is applied, and any arcs added. Shift transitions generate the alignments and predicates of the nodes placed on the buffer. Items on the stack and buffer have the form (*node index, alignment, predicate label*), and arcs are of the form (*head index, argument label, dependent index*).

4 Encoder-Decoder Models

4.1 Sentence encoder

The sentence $\mathbf{e}$ is encoded with a bidirectional RNN. We use a standard LSTM architecture without peephole connections (Jozefowicz et al., 2015). For every token e we embed its word, POS tag and named entity (NE) tag as vectors x_w, x_t and x_n, respectively.

The embeddings are concatenated and passed through a linear transformation

$$g(e) = W^{(x)}[x_w; x_t; x_n] + b^x,$$

such that $g(e)$ has the same dimension as the LSTM. Each input position i is represented by a hidden state h_i, which is the concatenation of its forward and backward LSTM state vectors.

4.2 Hard attention decoder

We model the alignment of graph nodes to sentence tokens, $\mathbf{a}$, as a random variable. For the arc-eager model, a_j corresponds to the alignment of the node of the buffer after action t_j is executed. The distribution of t_j is over all transitions and predicates (corresponding to shift transitions), predicted with a single softmax.

The parser output is predicted by an RNN decoder. Let s_j be the decoder hidden state at output position j. We initialize s_0 with the final state of the backward encoder. The alignment is predicted with a pointer network (Vinyals et al., 2015a).

The logits are computed with an MLP scoring the decoder hidden state against each of the encoder hidden states (for $i = 1, \ldots, I$),

$$u_j^i = w^T \tanh(W^{(1)}h_i + W^{(2)}s_j).$$

The alignment distribution is then estimated by

$$p(a_j = i|\mathbf{a}_{1:j-1}, \mathbf{t}_{1:j-1}, \mathbf{e}) = \text{softmax}(u_j^i).$$

To predict the next transition t_i, the output vector is conditioned on the encoder state vector h_{a_j}, corresponding to the alignment:

$$o_j = W^{(3)}s_j + W^{(4)}h_{a_j}$$
$$v_j = R^{(d)}o_j + b^{(d)},$$

where $R^{(d)}$ and $b^{(d)}$ are the output representation matrix and bias vector, respectively.

The transition distribution is then given by

$$p(t_j|\mathbf{a}_{1:j}, \mathbf{t}_{1:j-1}, \mathbf{e}) = \text{softmax}(v_j).$$

Let $e(t)$ be the embedding of decoder symbol t. The RNN state at the next time-step is computed as

$$d_{j+1} = W^{(5)}e(t_j) + W^{(6)}h_{a_j}$$
$$s_{j+1} = RNN(d_{j+1}, s_j).$$

The end-of-span alignment $a_j^{(e)}$ for MRS-based graphs is predicted with another pointer network. The end alignment of a token is predicted only when a node is reduced from the stack, therefore this alignment is not observed at each time-step; it is also not fed back into the model.

The hard attention approach, based on supervised alignments, can be contrasted to soft attention, which learns to attend over the input without supervision. The attention is computed as with hard attention, as $\alpha_j^i = \text{softmax}(u_j^i)$. However instead of making a hard selection, a weighted average over the encoder vectors is computed as $q_j = \sum_{i=1}^{i=I} \alpha_j^i h_i$. This vector is used instead of h_{a_j} for prediction and feeding to the next time-step.

4.3 Stack-based model

We extend the hard attention model to include features based on the transition system stack. These features are embeddings from the bidirectional RNN encoder, corresponding to the alignments of the nodes on the buffer and on top of the stack. This approach is similar to the features proposed by Kiperwasser and Goldberg (2016) and Cross and Huang (2016a) for dependency parsing, although they do not use RNN decoders.

To implement these features the layer that computes the output vector is extended to

$$o_j = W^{(3)} s_j + W^{(4)} h_{a_j} + W^{(7)} h_{\mathtt{st}_0},$$

where $\mathtt{st}_0$ is the sentence alignment index of the element on top of the stack. The input layer to the next RNN time-step is similarly extended to

$$d_{j+1} = W^{(5)} e(t_j) + W^{(6)} h_{\mathtt{buf}} + W^{(8)} h_{\mathtt{st}_0},$$

where `buf` is the buffer alignment after t_j is executed.

Our implementation of the stack-based model enables batch processing in static computation graphs, similar to Bowman et al. (2016). We maintain a stack of alignment indexes for each element in the batch, which is updated inside the computation graph after each parsing action. This enables minibatch SGD during training as well as efficient batch decoding.

We perform greedy decoding. For the stack-based model we ensure that if the stack is empty, the next transition predicted has to be shift. For the other models we ensure that the output is well-formed during post-processing by robustly skipping over out-of-place symbols or inserting missing ones.

5 Related Work

Prior work for MRS parsing predominantly predicts structures in the context of grammar-based parsing, where sentences are parsed to HPSG derivations consistent with the grammar, in this case the ERG (Flickinger, 2000). The nodes in the derivation trees are feature structures, from which MRS is extracted through unification. This approach fails to parse sentences for which no valid derivation is found. Maximum entropy models are used to score the derivations in order to find the most likely parse (Toutanova et al., 2005). This approach is implemented in the PET (Callmeier, 2000) and ACE[1] parsers.

There have also been some efforts to develop robust MRS parsers. One proposed approach learns a PCFG grammar to approximate the HPSG derivations (Zhang and Krieger, 2011; Zhang et al., 2014). MRS is then extracted with robust unification to compose potentially incompatible feature structures, although that still fails for a small proportion of sentences. The model is trained on a large corpus of Wikipedia text parsed with the grammar-based parser. Ytrestøl (2012) proposed a transition-based approach to HPSG parsing that produces derivations from which both syntactic and semantic (MRS) parses can be extracted. The parser has an option not to be restricted by the ERG. However, neither of these approaches have results available that can be compared directly to our setup, or generally available implementations.

Although AMR parsers produce graphs that are similar in structure to MRS-based graphs, most of them make assumptions that are invalid for MRS, and rely on extensive external AMR-specific resources. Flanigan et al. (2014) proposed a two-stage parser that first predicts concepts or subgraphs corresponding to sentence segments, and then parses these concepts into a graph structure. However MRS has a large proportion of abstract nodes that cannot be predicted from short segments, and interact closely with the graph structure. Wang et al. (2015b,a) proposed a custom transition-system for AMR parsing that converts dependency trees to AMR graphs, relying on assumptions on the relationship between these. Pust et al. (2015) proposed a parser based on syntax-based machine translation (MT), while AMR has also been integrated into CCG Semantic Parsing (Artzi et al., 2015; Misra and Artzi, 2016). Recently Damonte et al. (2017) and Peng et al. (2017) proposed AMR parsers based on neural networks.

6 Experiments

6.1 Data

DeepBank (Flickinger et al., 2012) is an HPSG and MRS annotation of the Penn Treebank Wall Street Journal (WSJ) corpus. It was developed following an approach known as dynamic treebanking (Oepen et al., 2004) that couples treebank annotation with grammar development, in this case

[1] http://sweaglesw.org/linguistics/ace/

of the ERG. This approach has been shown to lead to high inter-annotator agreement: 0.94 against 0.71 for AMR (Bender et al., 2015). Parses are only provided for sentences for which the ERG has an analysis acceptable to the annotator – this means that we cannot evaluate parsing accuracy for sentences which the ERG cannot parse (approximately 15% of the original corpus).

We use Deepbank version 1.1, corresponding to ERG 1214[2], following the suggested split of sections 0 to 19 as training data data, 20 for development and 21 for testing. The gold-annotated training data consists of 35,315 sentences. We use the LOGON environment[3] and the pyDelphin library[4] to extract DMRS and EDS graphs.

For AMR parsing we use LDC2015E86, the dataset released for the SemEval 2016 AMR parsing Shared Task (May, 2016). This data includes newswire, weblog and discussion forum text. The training set has 16,144 sentences. We obtain alignments using the rule-based JAMR aligner (Flanigan et al., 2014).

6.2 Evaluation

Dridan and Oepen (2011) proposed an evaluation metric called Elementary Dependency Matching (EDM) for MRS-based graphs. EDM computes the F1-score of tuples of predicates and arguments. A predicate tuple consists of the label and character span of a predicate, while an argument tuple consists of the character spans of the head and dependent nodes of the relation, together with the argument label. In order to tolerate subtle tokenization differences with respect to punctuation, we allow span pairs whose ends differ by one character to be matched.

The Smatch metric (Cai and Knight, 2013), proposed for evaluating AMR graphs, also measures graph overlap, but does not rely on sentence alignments to determine the correspondences between graph nodes. Smatch is instead computed by performing inference over graph alignments to estimate the maximum F1-score obtainable from a one-to-one matching between the predicted and gold graph nodes.

[2] http://svn.delph-in.net/erg/tags/1214/

[3] http://moin.delph-in.net/LogonTop

[4] https://github.com/delph-in/pydelphin

Model	EDM	EDM$_P$	EDM$_A$
TD lex	81.44	85.20	76.87
TD unlex	81.72	85.59	77.04
AE lex	81.35	85.79	76.02
AE unlex	82.56	86.76	77.54

Table 1: DMRS development set results for attention-based encoder-decoder models with alignments encoded in the linearization, for top-down (TD) and arc-eager (AE) linearizations, and lexicalized and unlexicalized predicate prediction.

6.3 Model setup

Our parser[5] is implemented in TensorFlow (Abadi et al., 2015). For training we use Adam (Kingma and Ba, 2015) with learning rate 0.01 and batch-size 64. Gradients norms are clipped to 5.0 (Pascanu et al., 2013). We use single-layer LSTMs with dropout of 0.3 (tuned on the development set) on input and output connections. We use encoder and decoder embeddings of size 256, and POS and NE tag embeddings of size 32, For DMRS and EDS graphs the hidden units size is set to 256, for AMR it is 128. This configuration, found using grid search and heuristic search within the range of models that fit into a single GPU, gave the best performance on the development set under multiple graph linearizations. Encoder word embeddings are initialized (in the first 100 dimensions) with pre-trained order-sensitive embeddings (Ling et al., 2015). Singletons in the encoder input are replaced with an unknown word symbol with probability 0.5 for each iteration.

6.4 MRS parsing results

We compare different linearizations and model architectures for parsing DMRS on the development data, showing that our approach is more accurate than baseline neural approaches. We report EDM scores, including scores for predicate (EDM$_P$) and argument (EDM$_A$) prediction.

First we report results using standard attention-based encoder-decoders, with the alignments encoded as token strings in the linearization. (Table 1). We compare the top-down (TD) and arc-eager (AE) linearizations, as well as the effect of delexicalizing the predicates (factorizing lemmas out of the linearization and predicting them sepa-

[5] Code and data preparation scripts are available at https://github.com/janmbuys/DeepDeepParser.

Model	EDM	EDM_P	EDM_A
TD soft	81.53	85.32	76.94
TD hard	82.75	86.37	78.37
AE hard	84.65	87.77	80.85
AE stack	85.28	88.38	81.51

Table 2: DMRS development set results of encoder-decoder models with pointer-based alignment prediction, delexicalized predicates and hard or soft attention.

rately.) In both cases constants are predicted with a dictionary lookup based on the predicted spans. A special label is predicted for predicates not in the ERG lexicon – the words and POS tags that make up those predicates are recovered through the alignments during post-processing.

The arc-eager unlexicalized representation gives the best performance, even though the model has to learn to model the transition system stack through the recurrent hidden states without any supervision of the transition semantics. The unlexicalized models are more accurate, mostly due to their ability to generalize to sparse or unseen predicates occurring in the lexicon. For the arc-eager representation, the oracle EDM is 99% for the lexicalized representation and 98.06% for the delexicalized representation. The remaining errors are mostly due to discrepancies between the tokenization used by our system and the ERG tokenization. The unlexicalized models are also faster to train, as the decoder's output vocabulary is much smaller, reducing the expense of computing softmaxes over large vocabularies.

Next we consider models with delexicalized linearizations that predict the alignments with pointer networks, contrasting soft and hard attention models (Table 2). The results show that the arc-eager models performs better than those based on top-down representation. For the arc-eager model we use hard attention, due to the natural interpretation of the alignment prediction corresponding to the transition system. The stack-based architecture gives further improvements.

When comparing the effect of different predicate orderings for the arc-eager model, we find that the monotone ordering performs 0.44 EDM better than the in-order ordering, despite having to parse more non-planar dependencies.

We also trained models that only predict predicates (in monotone order) together with their start spans. The hard attention model obtains 91.36% F1 on predicates together with their start spans with the unlexicalized model, compared to 88.22% for lexicalized predicates and 91.65% for the full parsing model.

Model	TD RNN	AE RNN	ACE
EDM	79.68	84.16	89.64
EDM_P	83.36	87.54	92.08
EDM_A	75.16	80.10	86.77
Start EDM	84.44	87.81	91.91
Start EDM_A	80.93	85.61	89.28
Smatch	85.28	86.69	93.50

Table 3: DMRS parsing test set results, comparing the standard top-down attention-based and arc-eager stack-based RNN models to the grammar-based ACE parser.

Table 3 reports test set results for various evaluation metrics. Start EDM is calculated by requiring only the start of the alignment spans to match, not the ends. We compare the performance of our baseline and stack-based models against ACE, the ERG-based parser.

Despite the promising performance of the model a gap remains between the accuracy of our parser and ACE. One reason for this is that the test set sentences will arguably be easier for ACE to parse as their choice was restricted by the same grammar that ACE uses. EDM metrics excluding end-span prediction (Start EDM) show that our parser has relatively more difficulty in parsing end-span predictions than the grammar-based parser.

We also evaluate the speed of our model compared with ACE. For the unbatched version of our model, the stack-based parser parses 41.63 tokens per second, while the batched implementation parses 529.42 tokens per second using a batch size of 128. In comparison, the setting of ACE for which we report accuracies parses 7.47 tokens per second. By restricting the memory usage of ACE, which restricts its coverage, we see that ACE can parse 11.07 tokens per second at 87.7% coverage, and 15.11 tokens per second at 77.8% coverage.

Finally we report results for parsing EDS (Table 4). The EDS parsing task is slightly simpler than DMRS, due to the absence of rich argument labels and additional graph edges that allow the recovery of full MRS. We see that for ACE the accuracies are very similar, while for our model EDS

Model	AE RNN	ACE
EDM	85.48	89.58
EDM_P	88.14	91.82
EDM_A	82.20	86.92
Smatch	86.50	93.52

Table 4: EDS parsing test set results.

Model	Concept F1	Smatch
TD no pointers	70.16	57.95
TD soft	71.25	59.39
TD soft unlex	72.62	59.88
AE hard unlex	76.83	59.83
AE stack unlex	77.93	61.21

Table 5: Development set results for AMR parsing. All the models except the first predict alignments with pointer networks.

parsing is more accurate on the EDM metrics. We hypothesize that most of the extra information in DMRS can be obtained through the ERG, to which ACE has access but our model doesn't.

An EDS corpus which consists of about 95% of the DeepBank data has also been released[6], with the goal of enabling comparison with other semantic graph parsing formalisms, including CCG dependencies and Prague Semantic Dependencies, on the same data set (Kuhlmann and Oepen, 2016). On this corpus our model obtains 85.87 EDM and 85.49 Smatch.

6.5 AMR parsing

We apply the same approach to AMR parsing. Results on the development set are given in Table 5. The arc-eager-based models again give better performance, mainly due to improved concept prediction accuracy. However, concept prediction remains the most important weakness of the model; Damonte et al. (2017) reports that state-of-the-art AMR parsers score 83% on concept prediction.

We report test set results in Table 6. Our best neural model outperforms the baseline JAMR parser (Flanigan et al., 2014), but still lags behind the performance of state-of-the-art AMR parsers such as CAMR (Wang et al., 2016) and AMR Eager (Damonte et al., 2017). These models make extensive use of external resources, including syntactic parsers and semantic role labellers. Our attention-based encoder-decoder model already outperforms previous sequence-to-sequence

[6] http://sdp.delph-in.net/osdp-12.tgz

Model	Smatch
Flanigan et al. (2014)	56
Wang et al. (2016)	66.54
Damonte et al. (2017)	64
Peng and Gildea (2016)	55
Peng et al. (2017)	52
Barzdins and Gosko (2016)	43.3
TD no pointers	56.56
AE stack delex	60.11

Table 6: AMR parsing test set results (Smatch F1 scores). Published results follow the number of decimals which were reported.

AMR parsers (Barzdins and Gosko, 2016; Peng et al., 2017), and the arc-eager model boosts accuracy further. Our model also outperforms a Synchronous Hyperedge Replacement Grammar model (Peng and Gildea, 2016) which is comparable as it does not make extensive use of external resources.

7 Conclusion

In this paper we advance the state of parsing by employing deep learning techniques to parse sentence to linguistically expressive semantic representations that have not previously been parsed in an end-to-end fashion. We presented a robust, wide-coverage parser for MRS that is faster than existing parsers and amenable to batch processing. We believe that there are many future avenues to explore to further increase the accuracy of such parsers, including different training objectives, more structured architectures and semi-supervised learning.

Acknowledgments

The first author thanks the financial support of the Clarendon Fund and the Skye Foundation. We thank Stephan Oepen for feedback and help with data preperation, and members of the Oxford NLP group for valuable discussions.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore,

Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. http://tensorflow.org/.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1699–1710. http://aclweb.org/anthology/D15-1198.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, pages 178–186. http://www.aclweb.org/anthology/W13-2322.

Guntis Barzdins and Didzis Gosko. 2016. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on AMR parsing accuracy. In *Proceedings of SemEval*.

Emily M Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann Copestake. 2015. Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics*. pages 239–249.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of ACL*. pages 1466–1477. http://www.aclweb.org/anthology/P16-1139.

Jan Buys and Phil Blunsom. 2017. Oxford at SemEval-2017 Task 9: Neural AMR parsing with pointer-augmented attention. In *Proceedings of SemEval*.

Shu Cai and Kevin Knight. 2013. Smatch: An evaluation metric for semantic feature structures. In *Proceedings of ACL (short papers)*.

Ulrich Callmeier. 2000. PET - a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering* 6(1):99–107.

Ann Copestake. 2009. Invited talk: Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of EACL*. pages 1–9. http://www.aclweb.org/anthology/E09-1001.

Ann Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszyska. 2016. Resources for building applications with dependency minimal recursion semantics. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. 1995. Translation using minimal recursion semantics. In *In Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation* 3(2-3):281–332.

James Cross and Liang Huang. 2016a. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of ACL*. page 32.

James Cross and Liang Huang. 2016b. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of EMNLP*. pages 1–11. https://aclweb.org/anthology/D16-1001.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of EACL*. pages 536–546. http://www.aclweb.org/anthology/E17-1051.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of ACL*. pages 33–43. http://www.aclweb.org/anthology/P16-1004.

Rebecca Dridan and Stephan Oepen. 2011. Parser evaluation using elementary dependency matching. In *Proceedings of the 12th International Conference on Parsing Technologies*. Association for Computational Linguistics, pages 225–230.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*. pages 334–343. http://www.aclweb.org/anthology/P15-1033.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of ACL*. pages 363–370. http://dx.doi.org/10.3115/1219840.1219885.

Jeffrey Flanigan, Sam Thomson, Jaime G. Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of ACL*. pages 1426–1436. http://aclweb.org/anthology/P/P14/P14-1134.pdf.

Dan Flickinger. 2000. On building a more effcient grammar by exploiting types. *Natural Language Engineering* 6(01):15–28.

Dan Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank. a dynamically annotated treebank of the wall street journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*. pages 85–96.

Carlos Gómez-Rodríguez and Joakim Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of ACL*. pages 1492–1501. http://www.aclweb.org/anthology/P10-1151.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*. pages 12–22. http://www.aclweb.org/anthology/P16-1002.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of ICML*. pages 2342–2350.

Ronald M Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. *Formal Issues in Lexical-Functional Grammar* pages 29–130.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*. http://arxiv.org/abs/1412.6980.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Marco Kuhlmann and Stephan Oepen. 2016. Towards a catalogue of linguistic graph banks. *Computational Linguistics* 42(4):819–827.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. volume 14, pages 1188–1196.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of NAACL-HLT*. pages 1299–1304. http://www.aclweb.org/anthology/N15-1142.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. In *Proceedings of SemEval*. pages 1063–1073. http://www.aclweb.org/anthology/S16-1166.

Dipendra Kumar Misra and Yoav Artzi. 2016. Neural shift-reduce ccg semantic parsing. In *Proceedings of EMNLP*. Austin, Texas, pages 1775–1786. https://aclweb.org/anthology/D16-1183.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics* 34(4):513–553.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. Lingo redwoods. *Research on Language and Computation* 2(4):575–596. https://doi.org/10.1007/s11168-004-7430-4.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of SemEval*. pages 915–926. http://www.aclweb.org/anthology/S15-2153.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of SemEval*. pages 63–72. http://www.aclweb.org/anthology/S14-2008.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-based MRS banking. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*. pages 1250–1255.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31(1):71–106.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28:1310–1318.

Xiaochang Peng and Daniel Gildea. 2016. Uofr at semeval-2016 task 8: Learning synchronous hyperedge replacement grammar for amr parsing. In *Proceedings of SemEval-2016*. pages 1185–1189. http://www.aclweb.org/anthology/S16-1183.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural amr parsing. In *Proceedings of EACL*. Preprint. http://www.cs.brandeis.edu/ cwang24/files/eacl17.pdf.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into abstract meaning representation using syntax-based machine translation. In *Proceedings of EMNLP*. Association for Computational Linguistics, Lisbon, Portugal, pages 1143–1154. http://aclweb.org/anthology/D15-1136.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664* .

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of Coling 2008*. pages 753–760. http://www.aclweb.org/anthology/C08-1095.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *IJCAI*. pages 1562–1567.

Kristina Toutanova, Christopher D. Manning, Dan Flickinger, and Stephan Oepen. 2005. Stochastic HPSG parse disambiguation using the redwoods corpus. *Research on Language and Computation* 3(1):83–105.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems 28*. pages 2692–2700. http://papers.nips.cc/paper/5866-pointer-networks.pdf.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2755–2763.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. Camr at semeval-2016 task 8: An extended transition-based amr parser. In *Proceedings of SemEval*. pages 1173–1178. http://www.aclweb.org/anthology/S16-1181.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of ACL (2)*. pages 857–862. http://www.aclweb.org/anthology/P15-2141.pdf.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proceedings of NAACL 2015*. pages 366–375. http://aclweb.org/anthology/N/N15/N15-1040.pdf.

Gisle Ytrestøl. 2012. *Transition-Based Parsing for Large-Scale Head-Driven Phrase Structure Grammars*. Ph.D. thesis, University of Oslo.

Yi Zhang and Hans-Ulrich Krieger. 2011. Large-scale corpus-driven PCFG approximation of an HPSG. In *Proceedings of the 12th international conference on parsing technologies*. Association for Computational Linguistics, pages 198–208.

Yi Zhang, Stephan Oepen, and John Carroll. 2007. Efficiency in unification-based n-best parsing. In *Proceedings of IWPT*. pages 48–59. http://www.aclweb.org/anthology/W/W07/W07-2207.

Yi Zhang, Stephan Oepen, Rebecca Dridan, Dan Flickinger, and Hans-Ulrich Krieger. 2014. Robust parsing, meaning composition, and evaluation: Integrating grammar approximation, default unification, and elementary semantic dependencies. Unpublished manuscript.

Joint Extraction of Entities and Relations Based on a Novel Tagging Scheme

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao,Peng Zhou, Bo Xu
Institute of Automation, Chinese Academy of Sciences, 100190, Beijing, P.R. China
{suncong.zheng, feng.wang,hongyun.bao, haoyuexing2014, peng.zhou,xubo}@ia.ac.cn

Abstract

Joint extraction of entities and relations is an important task in information extraction. To tackle this problem, we firstly propose a novel tagging scheme that can convert the joint extraction task to a tagging problem. Then, based on our tagging scheme, we study different end-to-end models to extract entities and their relations directly, without identifying entities and relations separately. We conduct experiments on a public dataset produced by distant supervision method and the experimental results show that the tagging based methods are better than most of the existing pipelined and joint learning methods. What's more, the end-to-end model proposed in this paper, achieves the best results on the public dataset.

1 Introduction

Joint extraction of entities and relations is to detect entity mentions and recognize their semantic relations simultaneously from unstructured text, as Figure 1 shows. Different from open information extraction (Open IE) (Banko et al., 2007) whose relation words are extracted from the given sentence, in this task, relation words are extracted from a predefined relation set which may not appear in the given sentence. It is an important issue in knowledge extraction and automatic construction of knowledge base.

Traditional methods handle this task in a pipelined manner, i.e., extracting the entities (Nadeau and Sekine, 2007) first and then recognizing their relations (Rink, 2010). This separated framework makes the task easy to deal with, and each component can be more flexible. But it neglects the relevance between these two sub-tasks and each subtask is an independent model. The results of entity recognition may affect the performance of relation classification and lead to erroneous delivery (Li and Ji, 2014).

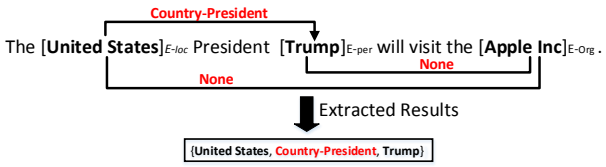


Figure 1: A standard example sentence for the task. "Country-President" is a relation in the predefined relation set.

Different from the pipelined methods, joint learning framework is to extract entities together with relations using a single model. It can effectively integrate the information of entities and relations, and it has been shown to achieve better results in this task. However, most existing joint methods are feature-based structured systems (Li and Ji, 2014; Miwa and Sasaki, 2014; Yu and Lam, 2010; Ren et al., 2017). They need complicated feature engineering and heavily rely on the other NLP toolkits, which might also lead to error propagation. In order to reduce the manual work in feature extraction, recently, (Miwa and Bansal, 2016) presents a neural network-based method for the end-to-end entities and relations extraction. Although the joint models can represent both entities and relations with shared parameters in a single model, they also extract the entities and relations separately and produce redundant information. For instance, the sentence in Figure 1 contains three entities: "United States", "Trump" and "Apple Inc". But only "United States" and "Trump" hold a fix relation "Country-President". Entity "Apple Inc" has no obvious relationship with the other entities in this sen-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1227–1236
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1113

tence. Hence, the extracted result from this sentence is {**United States**$_{e1}$, **Country-President**$_{r}$, **Trump**$_{e2}$}, which called triplet here.

In this paper, we focus on the extraction of triplets that are composed of two entities and one relation between these two entities. Therefore, we can model the triplets directly, rather than extracting the entities and relations separately. Based on the motivations, we propose a tagging scheme accompanied with the end-to-end model to settle this problem. We design a kind of novel tags which contain the information of entities and the relationships they hold. Based on this tagging scheme, the joint extraction of entities and relations can be transformed into a tagging problem. In this way, we can also easily use neural networks to model the task without complicated feature engineering.

Recently, end-to-end models based on LSTM (Hochreiter and Schmidhuber, 1997) have been successfully applied to various tagging tasks: Named Entity Recognition (Lample et al., 2016), CCG Supertagging (Vaswani et al., 2016), Chunking (Zhai et al., 2017) et al. LSTM is capable of learning long-term dependencies, which is beneficial to sequence modeling tasks. Therefore, based on our tagging scheme, we investigate different kinds of LSTM-based end-to-end models to jointly extract the entities and relations. We also modify the decoding method by adding a biased loss to make it more suitable for our special tags.

The method we proposed is a supervised learning algorithm. In reality, however, the process of manually labeling a training set with a large number of entity and relation is too expensive and error-prone. Therefore, we conduct experiments on a public dataset[1] which is produced by distant supervision method (Ren et al., 2017) to validate our approach. The experimental results show that our tagging scheme is effective in this task. In addition, our end-to-end model can achieve the best results on the public dataset.

The major contributions of this paper are: (1) A novel tagging scheme is proposed to jointly extract entities and relations, which can easily transform the extraction problem into a tagging task. (2) Based on our tagging scheme, we study different kinds of end-to-end models to settle the problem. The tagging-based methods are better than most of the existing pipelined and joint learning methods. (3) Furthermore, we also develop an end-to-end model with biased loss function to suit for the novel tags. It can enhance the association between related entities.

[1] https://github.com/shanzhenren/CoType

2 Related Works

Entities and relations extraction is an important step to construct a knowledge base, which can be benefit for many NLP tasks. Two main frameworks have been widely used to solve the problem of extracting entity and their relationships. One is the pipelined method and the other is the joint learning method.

The pipelined method treats this task as two separated tasks, i.e., named entity recognition (NER) (Nadeau and Sekine, 2007) and relation classification (RC) (Rink, 2010). Classical NER models are linear statistical models, such as Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (Passos et al., 2014; Luo et al., 2015). Recently, several neural network architectures (Chiu and Nichols, 2015; Huang et al., 2015; Lample et al., 2016) have been successfully applied to NER, which is regarded as a sequential token tagging task. Existing methods for relation classification can also be divided into handcrafted feature based methods (Rink, 2010; Kambhatla, 2004) and neural network based methods (Xu, 2015a; Zheng et al., 2016; Zeng, 2014; Xu, 2015b; dos Santos, 2015).

While joint models extract entities and relations using a single model. Most of the joint methods are feature-based structured systems (Ren et al., 2017; Yang and Cardie, 2013; Singh et al., 2013; Miwa and Sasaki, 2014; Li and Ji, 2014). Recently, (Miwa and Bansal, 2016) uses a LSTM-based model to extract entities and relations, which can reduce the manual work.

Different from the above methods, the method proposed in this paper is based on a special tagging manner, so that we can easily use end-to-end model to extract results without NER and RC. end-to-end method is to map the input sentence into meaningful vectors and then back to produce a sequence. It is widely used in machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014) and sequence tagging tasks (Lample et al., 2016; Vaswani et al., 2016). Most methods apply bidirectional LSTM to encode the input sentences, but the decoding methods are always different. For examples, (Lample et al., 2016) use a CRF layers to decode the tag sequence, while

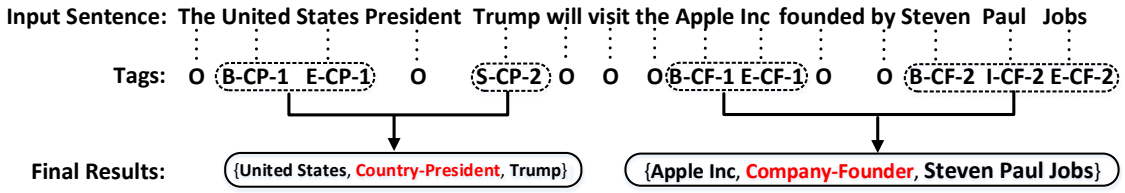


Figure 2: Gold standard annotation for an example sentence based on our tagging scheme, where "CP" is short for "Country-President" and "CF" is short for "Company-Founder".

(Vaswani et al., 2016; Katiyar and Cardie, 2016) apply LSTM layer to produce the tag sequence.

3 Method

We propose a novel tagging scheme and an end-to-end model with biased objective function to jointly extract entities and their relations. In this section, we firstly introduce how to change the extraction problem to a tagging problem based on our tagging method. Then we detail the model we used to extract results.

3.1 The Tagging Scheme

Figure 2 is an example of how the results are tagged. Each word is assigned a label that contributes to extract the results. Tag "O" represents the "Other" tag, which means that the corresponding word is independent of the extracted results. In addition to "O", the other tags consist of three parts: the word position in the entity, the relation type, and the relation role. We use the "BIES" (Begin, Inside, End,Single) signs to represent the position information of a word in the entity. The relation type information is obtained from a predefined set of relations and the relation role information is represented by the numbers "1" and "2". An extracted result is represented by a triplet: $(Entity_1, RelationType, Entity_2)$. "1" means that the word belongs to the first entity in the triplet, while "2" belongs to second entity that behind the relation type. Thus, the total number of tags is $N_t = 2 * 4 * |R| + 1$, where $|R|$ is the size of the predefined relation set.

Figure 2 is an example illustrating our tagging method. The input sentence contains two triplets: {**United States, Country-President, Trump**} and {**Apple Inc, Company-Founder, Steven Paul Jobs**}, where "Country-President" and "Company-Founder" are the predefined relation types. The words "United","States"," Trump","Apple","Inc" ,"Steven", "Paul" and "Jobs" are all related to the final extracted results. Thus they are tagged based on our special tags. For example, the word of "United" is the first word of entity "United States" and is related to the relation "Country-President", so its tag is "B-CP-1". The other entity " Trump", which is corresponding to "United States", is labeled as "S-CP-2". Besides, the other words irrelevant to the final result are labeled as "O".

3.2 From Tag Sequence To Extracted Results

From the tag sequence in Figure 2, we know that " Trump" and "United States" share the same relation type "Country-President", "Apple Inc" and "Steven Paul Jobs" share the same relation type "Company-Founder". We combine entities with the same relation type into a triplet to get the final result. Accordingly, " Trump" and "United States" can be combined into a triplet whose relation type is "Country-President". Because, the relation role of " Trump" is "2" and "United States" is "1", the final result is {**United States, Country-President, Trump**}. The same applies to {**Apple Inc, Company-Founder, Steven Paul Jobs**}.

Besides, if a sentence contains two or more triplets with the same relation type, we combine every two entities into a triplet based on the nearest principle. For example, if the relation type "Country-President" in Figure 2 is "Company-Founder", then there will be four entities in the given sentence with the same relation type. "United States" is closest to entity " Trump" and the "Apple Inc" is closest to "Jobs", so the results will be {**United States, Company-Founder, Trump**} and {**Apple Inc, Company-Founder, Steven Paul Jobs**}.

In this paper, we only consider the situation where an entity belongs to a triplet, and we leave identification of overlapping relations for future work.

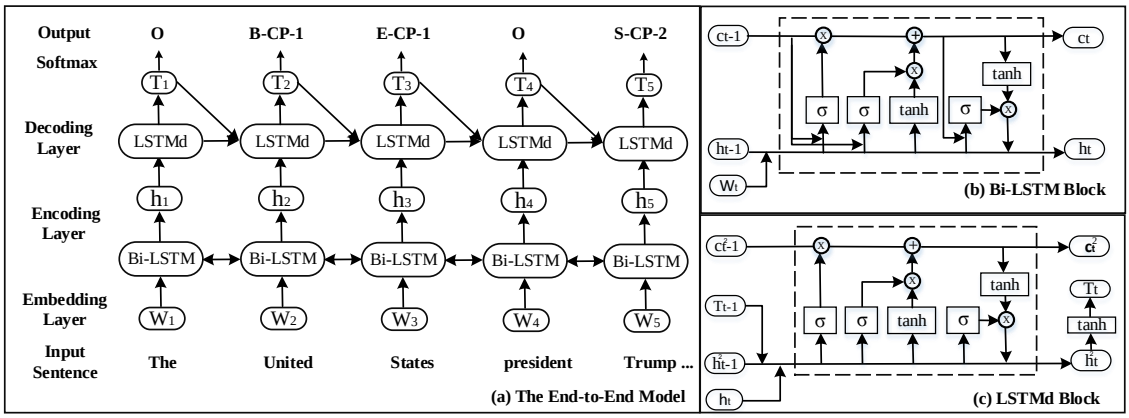


Figure 3: An illustration of our model. (a): The architecture of the end-to-end model, (b): The LSTM memory block in Bi-LSTM encoding layer, (c): The LSTM memory block in LSTM$_d$ decoding layer.

3.3 The End-to-end Model

In recent years, end-to-end model based on neural network is been widely used in sequence tagging task. In this paper, we investigate an end-to-end model to produce the tags sequence as Figure 3 shows. It contains a bi-directional Long Short Term Memory (Bi-LSTM) layer to encode the input sentence and a LSTM-based decoding layer with biased loss. The biased loss can enhance the relevance of entity tags.

The Bi-LSTM Encoding Layer. In sequence tagging problems, the Bi-LSTM encoding layer has been shown the effectiveness to capture the semantic information of each word. It contains forward lstm layer, backward lstm layer and the concatenate layer. The word embedding layer converts the word with 1-hot representation to an embedding vector. Hence, a sequence of words can be represented as $W = \{w_1, ...w_t, w_{t+1}...w_n\}$, where $w_t \in \mathbb{R}^d$ is the d-dimensional word vector corresponding to the t-th word in the sentence and n is the length of the given sentence. After word embedding layer, there are two parallel LSTM layers: forward LSTM layer and backward LSTM layer. The LSTM architecture consists of a set of recurrently connected subnets, known as memory blocks. Each time-step is a LSTM memory block. The LSTM memory block in Bi-LSTM encoding layer is used to compute current hidden vector h_t based on the previous hidden vector h_{t-1}, the previous cell vector c_{t-1} and the current input word embedding w_t. Its structure diagram is shown in Figure 3 (b), and detail operations are defined as follows:

$$i_t = \delta(W_{wi}w_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \quad (1)$$

$$f_t = \delta(W_{wf}w_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \quad (2)$$

$$z_t = tanh(W_{wc}w_t + W_{hc}h_{t-1} + b_c), \quad (3)$$

$$c_t = f_t c_{t-1} + i_t z_t, \quad (4)$$

$$o_t = \delta(W_{wo}w_t + W_{ho}h_{t-1} + W_{co}c_t + b_o), \quad (5)$$

$$h_t = o_t tanh(c_t), \quad (6)$$

where i, f and o are the input gate, forget gate and output gate respectively, b is the bias term, c is the cell memory, and $W_{(.)}$ are the parameters. For each word w_t, the forward LSTM layer will encode w_t by considering the contextual information from word w_1 to w_t, which is marked as $\overrightarrow{h_t}$. In the similar way, the backward LSTM layer will encode w_t based on the contextual information from w_n to w_t, which is marked as $\overleftarrow{h_t}$. Finally, we concatenate $\overleftarrow{h_t}$ and $\overrightarrow{h_t}$ to represent word t's encoding information, denoted as $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$.

The LSTM Decoding Layer. We also adopt a LSTM structure to produce the tag sequence. When detecting the tag of word w_t, the inputs of decoding layer are: h_t obtained from Bi-LSTM encoding layer, former predicted tag embedding T_{t-1}, former cell value $c^{(2)}_{t-1}$, and the former hidden vector in decoding layer $h^{(2)}_{t-1}$. The structure diagram of the memory block in LSTM$_d$ is shown in Figure 3 (c), and detail operations are defined as follows:

$$i^{(2)}_t = \delta(W^{(2)}_{wi}h_t + W^{(2)}_{hi}h^{(2)}_{t-1} + W_{ti}T_{t-1} + b^{(2)}_i), \quad (7)$$

$$f_t^{(2)} = \delta(W_{wf}^{(2)} h_t + W_{hf}^{(2)} h_{t-1}^{(2)} + W_{tf} T_{t-1} + b_f^{(2)}), \tag{8}$$

$$z_t^{(2)} = tanh(W_{wc}^{(2)} h_t + W_{hc}^{(2)} h_{t-1}^{(2)} + W_{tc} T_{t-1} + b_c^{(2)}), \tag{9}$$

$$c_t^{(2)} = f_t^{(2)} c_{t-1}^{(2)} + i_t^{(2)} z_t^{(2)}, \tag{10}$$

$$o_t^{(2)} = \delta(W_{wo}^{(2)} h_t + W_{ho}^{(2)} h_{t-1}^{(2)} + W_{co}^{(2)} c_t + b_o^{(2)}), \tag{11}$$

$$h_t^{(2)} = o_t^{(2)} tanh(c_t^{(2)}), \tag{12}$$

$$T_t = W_{ts} h_t^{(2)} + b_{ts}. \tag{13}$$

The final softmax layer computes normalized entity tag probabilities based on the tag predicted vector T_t:

$$y_t = W_y T_t + b_y, \tag{14}$$

$$p_t^i = \frac{exp(y_t^i)}{\sum_{j=1}^{N_t} exp(y_t^j)}, \tag{15}$$

where W_y is the softmax matrix, N_t is the total number of tags. Because T is similar to tag embedding and LSTM is capable of learning long-term dependencies, the decoding manner can model tag interactions.

The Bias Objective Function. We train our model to maximize the log-likelihood of the data and the optimization method we used is RMSprop proposed by Hinton in (Tieleman and Hinton, 2012). The objective function can be defined as:

$$L = max \sum_{j=1}^{|\mathbb{D}|} \sum_{t=1}^{L_j} (log(p_t^{(j)} = y_t^{(j)} | x_j, \Theta) \cdot I(O) + \alpha \cdot log(p_t^{(j)} = y_t^{(j)} | x_j, \Theta) \cdot (1 - I(O))),$$

where $|\mathbb{D}|$ is the size of training set, L_j is the length of sentence x_j, $y_t^{(j)}$ is the label of word t in sentence x_j and $p_t^{(j)}$ is the normalized probabilities of tags which defined in Formula 15. Besides, $I(O)$ is a switching function to distinguish the loss of tag 'O' and relational tags that can indicate the results. It is defined as follows:

$$I(O) = \begin{cases} 1, & if \quad tag = 'O' \\ 0, & if \quad tag \neq 'O'. \end{cases}$$

α is the bias weight. The larger α is, the greater influence of relational tags on the model.

4 Experiments

4.1 Experimental setting

Dataset To evaluate the performance of our methods, we use the public dataset NYT [2] which is produced by distant supervision method (Ren et al., 2017). A large amount of training data can be obtained by means of distant supervision methods without manually labeling. While the test set is manually labeled to ensure its quality. In total, the training data contains $353k$ triplets, and the test set contains $3,880$ triplets. Besides, the size of relation set is 24.

Evaluation We adopt standard Precision (Prec), Recall (Rec) and F1 score to evaluate the results. Different from classical methods, our method can extract triplets without knowing the information of entity types. In other words, we did not use the label of entity types to train the model, therefore we do not need to consider the entity types in the evaluation. A triplet is regarded as correct when its relation type and the head offsets of two corresponding entities are both correct. Besides, the ground-truth relation mentions are given and "None" label is excluded as (Ren et al., 2017; Li and Ji, 2014; Miwa and Bansal, 2016) did. We create a validation set by randomly sampling 10% data from test set and use the remaining data as evaluation based on (Ren et al., 2017)'s suggestion. We run 10 times for each experiment then report the average results and their standard deviation as Table 1 shows.

Hyperparameters Our model consists of a Bi-LSTM encoding layer and a LSTM decoding layer with bias objective function. The word embeddings used in the encoding part are initialed by running word2vec[3] (Mikolov et al., 2013) on NYT training corpus. The dimension of the word embeddings is $d = 300$. We regularize our network using dropout on embedding layer and the dropout ratio is 0.5. The number of lstm units in encoding layer is 300 and the number in decoding layer is 600. The bias parameter α corresponding to the results in Table 1 is 10.

[2]The dataset can be downloaded at: https://github.com/shanzhenren/CoType. There are three data sets in the public resource and we only use the NYT dataset. Because more than 50% of the data in BioInfer has overlapping relations which is beyond the scope of this paper. As for dataset Wiki-KBP, the number of relation type in the test set is more than that of the train set, which is also not suitable for a supervised training method. Details of the data can be found in Ren's(Ren et al., 2017) paper.

[3]https://code.google.com/archive/p/word2vec/

Methods	*Prec.*	*Rec.*	*F1*
FCM	0.553	0.154	0.240
DS+logistic	0.258	0.393	0.311
LINE	0.335	0.329	0.332
MultiR	0.338	0.327	0.333
DS-Joint	0.574	0.256	0.354
CoType	0.423	**0.511**	0.463
LSTM-CRF	$\mathbf{0.693 \pm 0.008}$	0.310 ± 0.007	0.428 ± 0.008
LSTM-LSTM	0.682 ± 0.007	0.320 ± 0.006	0.436 ± 0.006
LSTM-LSTM-Bias	0.615 ± 0.008	0.414 ± 0.005	$\mathbf{0.495 \pm 0.006}$

Table 1: The predicted results of different methods on extracting both entities and their relations. The first part (from row 1 to row 3) is the pipelined methods and the second part (row 4 to 6) is the jointly extracting methods. Our tagging methods are shown in part three (row 7 to 9). In this part, we not only report the results of precision, recall and F1, we also compute their standard deviation.

Baselines We compare our method with several classical triplet extraction methods, which can be divided into the following categories: the pipelined methods, the jointly extracting methods and the end-to-end methods based our tagging scheme.

For the pipelined methods, we follow (Ren et al., 2017)'s settings: The NER results are obtained by CoType (Ren et al., 2017) then several classical relation classification methods are applied to detect the relations. These methods are: (1) DS-logistic (Mintz et al., 2009) is a distant supervised and feature based method, which combines the advantages of supervised IE and unsupervised IE features; (2) LINE (Tang et al., 2015) is a network embedding method, which is suitable for arbitrary types of information networks; (3) FCM (Gormley et al., 2015) is a compositional model that combines lexicalized linguistic context and word embeddings for relation extraction.

The jointly extracting methods used in this paper are listed as follows: (4) DS-Joint (Li and Ji, 2014) is a supervised method, which jointly extracts entities and relations using structured perceptron on human-annotated dataset; (5) MultiR (Hoffmann et al., 2011) is a typical distant supervised method based on multi-instance learning algorithms to combat the noisy training data; (6) CoType (Ren et al., 2017) is a domain independent framework by jointly embedding entity mentions, relation mentions, text features and type labels into meaningful representations.

In addition, we also compare our method with two classical end-to-end tagging models: LSTM-CRF (Lample et al., 2016) and LSTM-LSTM (Vaswani et al., 2016). LSTM-CRF is proposed for entity recognition by using a bidirectional LSTM to encode input sentence and a conditional random fields to predict the entity tag sequence. Different from LSTM-CRF, LSTM-LSTM uses a LSTM layer to decode the tag sequence instead of CRF. They are used for the first time to jointly extract entities and relations based on our tagging scheme.

4.2 Experimental Results

We report the results of different methods as shown in Table 1. It can be seen that our method, LSTM-LSTM-Bias, outperforms all other methods in F1 score and achieves a 3% improvement in $F1$ over the best method CoType (Ren et al., 2017). It shows the effectiveness of our proposed method. Furthermore, from Table 1, we also can see that the jointly extracting methods are better than pipelined methods, and the tagging methods are better than most of the jointly extracting methods. It also validates the validity of our tagging scheme for the task of jointly extracting entities and relations.

When compared with the traditional methods, the precisions of the end-to-end models are significantly improved. But only LSTM-LSTM-Bias can be better to balance the precision and recall. The reason may be that these end-to-end models all use a Bi-LSTM encoding input sentence and different neural networks to decode the results. The methods based on neural networks can well fit the data. Therefore, they can learn the common features of the training set well and may lead to the lower expansibility. We also find that the LSTM-LSTM

Elements	E1			E2			(E1,E2)		
PRF	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*
LSTM-CRF	**0.596**	0.325	0.420	0.605	0.325	0.423	**0.724**	0.341	0.465
LSTM-LSTM	0.593	0.342	0.434	**0.619**	0.334	0.434	0.705	0.340	0.458
LSTM-LSTM-Bias	0.590	**0.479**	**0.529**	0.597	**0.451**	**0.514**	0.645	**0.437**	**0.520**

Table 2: The predicted results of triplet's elements based on our tagging scheme.

model is better than LSTM-CRF model based on our tagging scheme. Because, LSTM is capable of learning long-term dependencies and CRF (Lafferty et al., 2001) is good at capturing the joint probability of the entire sequence of labels. The related tags may have a long distance from each other. Hence, LSTM decoding manner is a little better than CRF. LSTM-LSTM-Bias adds a bias weight to enhance the effect of entity tags and weaken the effect of invalid tag. Therefore, in this tagging scheme, our method can be better than the common LSTM-decoding methods.

5 Analysis and Discussion

5.1 Error Analysis

In this paper, we focus on extracting triplets composed of two entities and a relation. Table 1 has shown the predict results of the task. It treats an triplet is correct only when the relation type and the head offsets of two corresponding entities are both correct. In order to find out the factors that affect the results of end-to-end models, we analyze the performance on predicting each element in the triplet as Table 2 shows. $E1$ and $E2$ represent the performance on predicting each entity, respectively. If the head offset of the first entity is correct, then the instance of $E1$ is correct, the same to $E2$. Regardless of relation type, if the head offsets of two corresponding entities are both correct, the instance of $(E1, E2)$ is correct.

As shown in Table 2, $(E1, E2)$ has higher precision when compared with $E1$ and $E2$. But its recall result is lower than $E1$ and $E2$. It means that some of the predicted entities do not form a pair. They only obtain $E1$ and do not find its corresponding $E2$, or obtain $E2$ and do not find its corresponding $E1$. Thus it leads to the prediction of more single E and less $(E1, E2)$ pairs. Therefore, entity pair $(E1, E2)$ has higher precision and lower recall than single E. Besides, the predicted results of $(E1, E2)$ in Table 2 have about 3% improvement when compared predicted results in Table 1, which means that 3% of the test data is predicted to be wrong because the relation type is predicted to be wrong.

5.2 Analysis of Biased Loss

Different from LSTM-CRF and LSTM-LSTM, our approach is biased towards relational labels to enhance links between entities. In order to further analyze the effect of the bias objective function, we visualize the ratio of predicted single entities for each end-to-end method as Figure 4. The single entities refer to those who cannot find their corresponding entities. Figure 4 shows whether it is E1 or E2, our method can get a relatively low ratio on the single entities. It means that our method can effectively associate two entities when compared LSTM-CRF and LSTM-LSTM which pay little attention to the relational tags.

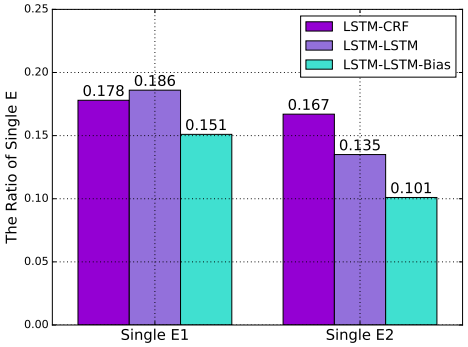


Figure 4: The ratio of predicted single entities for each method. The higher of the ratio the more entities are left.

Besides, we also change the Bias Parameter α from 1 to 20, and the predicted results are shown in Figure 5. If α is too large, it will affect the accuracy of prediction and if α is too small, the recall will decline. When $\alpha = 10$, LSTM-LSTM-Bias can balance the precision and recall, and can achieve the best F1 scores.

Standard S1	**[Panama City Beach]**$_{E2contain}$ has condos , but the area was one of only two in **[Florida]**$_{E1contain}$ where sales rose in March , compared with a year earlier.
LSTM-LSTM	Panama City Beach has condos , but the area was one of only two in **[Florida]**$_{E1contain}$ where sales rose in March , compared with a year earlier.
LSTM-LSTM-Bias	**[Panama City Beach]**$_{E2contain}$ has condos , but the area was one of only two in **[Florida]**$_{E1contain}$ where sales rose in March , compared with a year earlier.
Standard S2	All came from **[Nuremberg]**$_{E2contain}$, **[Germany]**$_{E1contain}$, a center of brass production since the Middle Ages.
LSTM-LSTM	All came from Nuremberg , **[Germany]**$_{E1contain}$, a center of brass production since the **[Middle Ages]**$_{E2contain}$.
LSTM-LSTM-Bias	All came from Nuremberg , **[Germany]**$_{E1contain}$, a center of brass production since the **[Middle Ages]**$_{E2contain}$.
Standard S3	**[Stephen A.]**$_{E2CF}$, the co-founder of the **[Blackstone Group]**$_{E1CF}$, which is in the process of going public , made $ 400 million last year.
LSTM-LSTM	**[Stephen A.]**$_{E1CF}$, the co-founder of the **[Blackstone Group]**$_{E1CF}$, which is in the process of going public , made $ 400 million last year.
LSTM-LSTM-Bias	**[Stephen A.]**$_{E1CF}$, the co-founder of the **[Blackstone Group]**$_{E2CF}$, which is in the process of going public , made $ 400 million last year.

Table 3: Output from different models. Standard S_i represents the gold standard of sentence i. The blue part is the correct result, and the red one is the wrong one. $E1CF$ in case '3' is short for $E1_{Company-Founder}$.

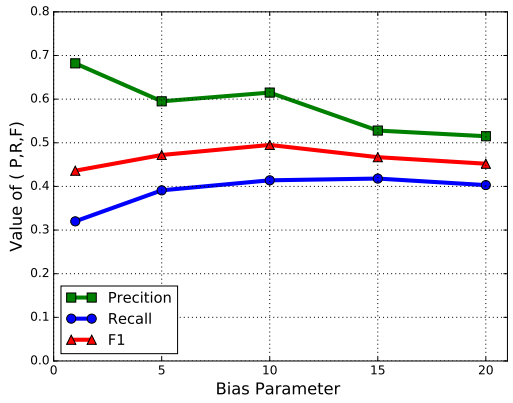


Figure 5: The results predicted by LSTM-LSTM-Bias on different bias parameter α.

5.3 Case Study

In this section, we observe the prediction results of end-to-end methods, and then select several representative examples to illustrate the advantages and disadvantages of the methods as Table 3 shows. Each example contains three row, the first row is the gold standard, the second and the third rows are the extracted results of model LSTM-LSTM and LSTM-LSTM-Bias respectively.

$S1$ represents the situation that the distance between the two interrelated entities is far away from each other, which is more difficult to detect their relationships. When compared with LSTM-LSTM, LSTM-LSTM-Bias uses a bias objective function which enhance the relevance between entities. Therefore, in this example, LSTM-LSTM-Bias can extract two related entities, while LSTM-LSTM can only extract one entity of "Florida" and can not detect entity "Panama City Beach".

$S2$ is a negative example that shows these methods may mistakenly predict one of the entity. There are no indicative words between entities $Nuremberg$ and $Germany$. Besides, the patten "a * of *" between $Germany$ and $MiddleAges$ may be easy to mislead the models that there exists a relation of "Contains" between them. The problem can be solved by adding some samples of this kind of expression patterns to the training data.

$S3$ is a case that models can predict the entities' head offset right, but the relational role is wrong. LSTM-LSTM treats both "Stephen A. Schwarzman" and "Blackstone Group" as entity $E1$, and can not find its corresponding $E2$. Although, LSTM-LSMT–Bias can find the entities pair $(E1, E2)$, it reverses the roles of "Stephen A. Schwarzman" and "Blackstone Group". It shows that LSTM-LSTM-Bias is able to better on pre-

dicting entities pair, but it remains to be improved in distinguishing the relationship between the two entities.

6 Conclusion

In this paper, we propose a novel tagging scheme and investigate the end-to-end models to jointly extract entities and relations. The experimental results show the effectiveness of our proposed method. But it still has shortcoming on the identification of the overlapping relations. In the future work, we will replace the softmax function in the output layer with multiple classifier, so that a word can has multiple tags. In this way, a word can appear in multiple triplet results, which can solve the problem of overlapping relations. Although, our model can enhance the effect of entity tags, the association between two corresponding entities still requires refinement in next works.

Acknowledgments

We thank Xiang Ren for dataset details and helpful discussions. This work is also supported by the National High Technology Research and Development Program of China (863 Program) (Grant No. 2015AA015402), the National Natural Science Foundation of China (No. 61602479) and the NSFC project 61501463.

References

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*. volume 7, pages 2670–2676.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. In *Processings of Transactions of the Association for Computational Linguistics*.

Cıcero Nogueira et al. dos Santos. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53th ACL international conference*. volume 1, pages 626–634.

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the EMNLP*.

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

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 541–550.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*. volume 3, page 413.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the 43th ACL international conference*. page 22.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proceedings of the 54th ACL international conference*.

John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*. volume 1, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the NAACL international conference*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52rd Annual Meeting of the Association for Computational Linguistics*. pages 402–412.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Conference on Empirical Methods in Natural Language Processing*. pages 879–888.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL*. Association for Computational Linguistics, pages 1003–1011.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics*.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 1858–1869.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1):3–26.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *International Conference on Computational Linguistics*. pages 78–86.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. 2017. Cotype: Joint extraction of typed entities and relations with knowledge bases. In *Proceedings of the 26th WWW international conference*.

Bryan et al. Rink. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. pages 256–259.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 workshop on Automated knowledge base construction*. ACM, pages 1–6.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, pages 1067–1077.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop. In *COURSERA: Neural networks for machine learning*.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with lstms. In *Proceedings of the NAACL international conference*. pages 232–237.

Kun et al. Xu. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the EMNLP*.

Yan et al. Xu. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of EMNLP international conference*.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51rd Annual Meeting of the Association for Computational Linguistics*. pages 1640–1649.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of the 21th COLING international conference*. pages 1399–1407.

Daojian et al. Zeng. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 25th COLING international conference*. pages 2335–2344.

Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. 2017. Neural models for sequence chunking. In *Proceedings of the AAAI international conference*.

Suncong Zheng, Jiaming Xu, Peng Zhou, Hongyun Bao, Zhenyu Qi, and Bo Xu. 2016. A neural network framework for relation extraction: Learning entity semantic and relation pattern. *Knowledge-Based Systems* 114:12–23.

A Local Detection Approach for Named Entity Recognition and Mention Detection

Mingbin Xu, Hui Jiang, Sedtawut Watcharawittayakul
Department of Electrical Engineering and Computer Science
Lassonde School of Engineering, York University
4700 Keele Street, Toronto, Ontario, Canada
{xmb, hj, watchara}@eecs.yorku.ca

Abstract

In this paper, we study a novel approach for named entity recognition (NER) and mention detection (MD) in natural language processing. Instead of treating NER as a sequence labeling problem, we propose a new local detection approach, which relies on the recent fixed-size ordinally forgetting encoding (FOFE) method to fully encode each sentence fragment and its left/right contexts into a fixed-size representation. Subsequently, a simple feedforward neural network (FFNN) is learned to either reject or predict entity label for each individual text fragment. The proposed method has been evaluated in several popular NER and MD tasks, including CoNLL 2003 NER task and TAC-KBP2015 and TAC-KBP2016 Tri-lingual Entity Discovery and Linking (EDL) tasks. Our method has yielded pretty strong performance in all of these examined tasks. This local detection approach has shown many advantages over the traditional sequence labeling methods.

1 Introduction

Natural language processing (NLP) plays an important role in artificial intelligence, which has been extensively studied for many decades. Conventional NLP techniques include the rule-based symbolic approaches widely used about two decades ago, and the more recent statistical approaches relying on feature engineering and statistical models. In the recent years, deep learning approach has achieved huge successes in many applications, ranging from speech recognition to image classification. It is drawing increasing attention in the NLP community.

In this paper, we are interested in a fundamental problem in NLP, namely named entity recognition (NER) and mention detection (MD). NER and MD are very challenging tasks in NLP, laying the foundation of almost every NLP application. NER and MD are tasks of identifying entities (named and/or nominal) from raw text, and classifying the detected entities into one of the pre-defined categories such as person (PER), organization (ORG), location (LOC), etc. Some tasks focus on named entities only, while the others also detect nominal mentions. Moreover, nested mentions may need to be extracted too. For example,

$$[Sue]_{PER} \text{ and her } [brother]_{PER\_N} \text{ studied in } [University\ of\ [Toronto]_{LOC}]_{ORG}.$$

where *Toronto* is a LOC entity, embedded in another longer ORG entity *University of Toronto*.

Similar to many other NLP problems, NER and MD is formulated as a sequence labeling problem, where a tag is sequentially assigned to each word in the input sentence. It has been extensively studied in the NLP community (Borthwick et al., 1998). The core problem is to model the conditional probability of an output sequence given an arbitrary input sequence. Many hand-crafted features are combined with statistical models, such as conditional random fields (CRFs) (Nguyen et al., 2010), to compute conditional probabilities. More recently, some popular neural networks, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are proposed to solve sequence labelling problems. In the inference stage, the learned models compute the conditional probabilities and the output sequence is generated by the Viterbi decoding algorithm (Viterbi, 1967).

In this paper, we propose a novel local detection approach for solving NER and MD problems. The idea can be easily extended to many other se-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1237–1247
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1114

quence labeling problems, such as chunking, part-of-speech tagging (POS). Instead of globally modeling the whole sequence in training and jointly decode the entire output sequence in test, our method examines all word segments (up to a certain length) in a sentence. A word segment will be examined individually based on the underlying segment itself and its left and right contexts in the sentence so as to determine whether this word segment is a valid named entity and the corresponding label if it is. This approach conforms to the way human resolves an NER problem. Given any word fragment and its contexts in a sentence or paragraph, people accurately determine whether this word segment is a named entity or not. People rarely conduct a global decoding over the entire sentence to make such a decision. The key to making an accurate local decision for each individual fragment is to have full access to the fragment itself as well as its complete contextual information. The main pitfall to implement this idea is that we can not easily encode the segment and its contexts in models since they are of varying lengths in natural languages. Many feature engineering techniques have been proposed but all of these methods will inevitably lead to information loss. In this work, we propose to use a recent fixed-size encoding method, namely fixed-size ordinally forgetting encoding (FOFE) (Zhang et al., 2015a,b), to solve this problem. The FOFE method is a simple recursive encoding method. FOFE theoretically guarantees (almost) unique and lossless encoding of any variable-length sequence. The left and the right contexts for each word segment are encoded by FOFE method, and then a simple neural network can be trained to make a precise recognition for each individual word segment based on the fixed-size presentation of the contextual information. This FOFE-based local detection approach is more appealing to NER and MD. Firstly, feature engineering is almost eliminated. Secondly, under this local detection framework, nested mention is handled with little modification. Next, it makes better use of partially-labeled data available from many application scenarios. Sequence labeling model requires all entities in a sentence to be labeled. If only some (not all) entities are labeled, it is not effective to learn a sequence labeling model. However, every single labeled entity, along with its contexts, may be used to learn the proposed model. At last, due to the simplicity of FOFE, simple neural networks, such as multilayer perceptrons, are sufficient for recognition. These models are much faster to train and easier to tune. In the test stage, all possible word segments from a sentence may be packed into a mini-batch, jointly recognized in parallel on GPUs. This leads to a very fast decoding process as well.

In this paper, we have applied this FOFE-based local detection approach to several popular NER and MD tasks, including the CoNLL 2003 NER task and TAC-KBP2015 and TAC-KBP2016 Tri-lingual Entity Discovery and Linking (EDL) tasks. Our proposed method has yielded strong performance in all of these examined tasks.

2 Related Work

It has been a long history of research involving neural networks (NN). In this section, we briefly review some recent NN-related research work in NLP, which may be relevant to our work.

The success of word embedding (Mikolov et al., 2013; Liu et al., 2015) encourages researchers to focus on machine-learned representation instead of heavy feature engineering in NLP. Using word embedding as the typical feature representation for words, NNs become competitive to traditional approaches in NER. Many NLP tasks, such as NER, chunking and part-of-speech (POS) tagging can be formulated as sequence labeling tasks. In (Collobert et al., 2011), deep convolutional neural networks (CNN) and conditional random fields (CRF) are used to infer NER labels at a sentence level, where they still use many hand-crafted features to improve performance, such as capitalization features explicitly defined based on first-letter capital, non-initial capital and so on.

Recently, recurrent neural networks (RNNs) have demonstrated the ability in modeling sequences (Graves, 2012). Huang et al. (2015) built on the previous CNN-CRF approach by replacing CNNs with bidirectional Long Short-Term Memory (B-LSTM). Though they have reported improved performance, they employ heavy feature engineering in that work, most of which is language-specific. There is a similar attempt in (Rondeau and Su, 2016) with full-rank CRF. CNNs are used to extract character-level features automatically in (dos Santos et al., 2015).

Gazetteer is a list of names grouped by the predefined categories. Gazetteer is shown to be one of the most effective external knowledge sources

to improve NER performance (Sang and Meulder, 2003). Thus, gazetteer is widely used in many NER systems. In (Chiu and Nichols, 2016), state-of-the-art performance on a popular NER task, i.e., CoNLL2003, is achieved by incorporating a large gazetteer. Different from previous ways to use a set of bits to indicate whether a word is in gazetteer or not, they have encoded a match in BIOES (Begin, Inside, Outside, End, Single) annotation, which captures positional information.

Interestingly enough, none of these recent successes in NER was achieved by a vanilla RNN. Rather, these successes are often established by sophisticated models combining CNNs, LSTMs and CRFs in certain ways. In this paper, based on recent work in (Zhang et al., 2015a,b) and (Zhang et al., 2016), we propose a novel but simple solution to NER by applying DNN on top of FOFE-based features. This simpler approach can achieve performance very close to state-of-the-art on various NER and MD tasks, without using any external knowledge or feature engineering.

3 Preliminary

In this section, we will briefly review some background techniques, which are important to our proposed NER and mention detection approach.

3.1 Deep Feedforward Neural Networks

It is well known that neural network is a universal approximator under certain conditions (Hornik, 1991). A feedforward neural network (FFNN) is a weighted graph with a layered architecture. Each layer is composed of several nodes. Successive layers are fully connected. Each node applies a function on the weighted sum of the lower layer. An NN can learn by adjusting its weights in a process called back-propagation. The learned NN may be used to generalize and extrapolate to new inputs that have not been seen during training.

3.2 Fixed-size Ordinally Forgetting Encoding

FFNN is a powerful computation model. However, it requires fixed-size inputs and lacks the ability of capturing long-term dependency. Because most NLP problems involves variable-length sequences of words, RNNs/LSTMs are more popular than FFNNs in dealing with these problems. The Fixed-size Ordinally Forgetting Encoding (FOFE), originally proposed in (Zhang et al., 2015a,b), nicely overcomes the limitations of FFNNs because it can uniquely and losslessly encode a variable-length sequence of words into a fixed-size representation.

Give a vocabulary V, each word can be represented by a one-hot vector. FOFE mimics bag-of-words (BOW) but incorporates a forgetting factor to capture positional information. It encodes any sequence of variable length composed by words in V. Let $S = w_1, w_2, w_3, ..., w_T$ denote a sequence of T words from V, and $\boldsymbol{e_t}$ be the one-hot vector of the t-th word in S, where $1 \leq t \leq T$. The FOFE of each partial sequence $\boldsymbol{z_t}$ from the first word to the t-th word is recursively defined as:

$$\boldsymbol{z_t} = \begin{cases} \mathbf{0}, & \text{if } t = 0 \\ \alpha \cdot \boldsymbol{z_{t-1}} + \boldsymbol{e_t}, & \text{otherwise} \end{cases} \quad (1)$$

where the constant α is called forgetting factor, and it is picked between 0 and 1 exclusively. Obviously, the size of $\boldsymbol{z_t}$ is $|V|$, and it is irrelevant to the length of original sequence, T.

Here's an example. Assume that we have three words in our vocabulary, e.g. A, B, C, whose one-hot representations are $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$ respectively. When calculating from left to right, the FOFE for the sequence "ABC" is $[\alpha^2, \alpha, 1]$ and that of "ABCBC" is $[\alpha^4, \alpha+\alpha^3, 1+\alpha^2]$.

The word sequences can be unequivocally recovered from their FOFE representations (Zhang et al., 2015a,b). The uniqueness of FOFE representation is theoretically guaranteed by the following two theorems:

Theorem 1. *If the forgetting factor α satisfies $0 < \alpha \leq 0.5$, FOFE is unique for any countable vocabulary V and any finite value T.*

Theorem 2. *For $0.5 < \alpha < 1$, given any finite value T and any countable vocabulary V, FOFE is almost unique everywhere, except only a finite set of countable choices of α.*

Though in theory uniqueness is not guaranteed when α is chosen from 0.5 to 1, in practice the chance of hitting such scenarios is extremely slim, almost impossible due to quantization errors in the system. Furthermore, in natural languages, normally a word does not appear repeatedly within a near context. Simply put, FOFE is capable of uniquely encoding any sequence of arbitrary length, serving as a fixed-size but theoretically lossless representation for any sequence.

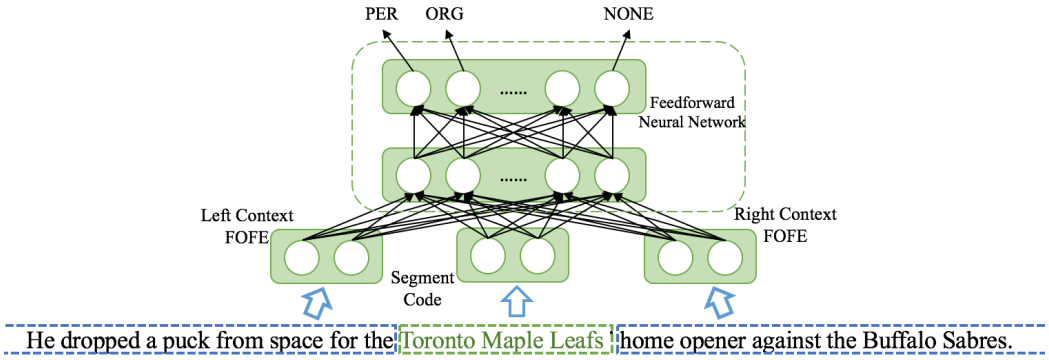


Figure 1: Illustration of the local detection approach for NER using FOFE codes as input and an FFNN as model. The window currently examines the fragment of *Toronto Maple Leafs*. The window will scan and scrutinize all fragments up to K words.

3.3 Character-level Models in NLP

Kim et al. (2016) model morphology in the character level since this may provide some additional advantages in dealing with unknown or out-of-vocabulary (OOVs) words in a language. In the literature, convolutional neural networks (CNNs) have been widely used as character-level models in NLP (Kim et al., 2016). A trainable character embedding is initialized based on a set of possible characters. When a word fragment comes, character vectors are retrieved according to its spelling to construct a matrix. This matrix can be viewed as a single-channel image. CNN is applied to generate a more abstract representation of the word fragment.

The above FOFE method can be easily extended to model character-level feature in NLP. Any word, phrase or fragment can be viewed as a sequence of characters. Based on a pre-defined set of all possible characters, we apply the same FOFE method to encode the sequence of characters. This always leads to a fixed-size representation, irrelevant to the number of characters in question. For example, a word fragment of "Walmart" may be viewed as a sequence of seven characters: 'W', 'a', 'l', 'm', 'a', 'r', 't'. The FOFE codes of character sequences are always fixed-sized and they can be directly fed to an FFNN for morphology modeling.

4 FOFE-based Local Detection for NER

As described above, our FOFE-based local detection approach for NER, called **FOFE-NER** hereafter, is motivated by the way how human actually infers whether a word segment in text is an entity or mention, where the entity types of the other entities in the same sentence is not a must. Particularly, the dependency between adjacent entities is fairly weak in NER. Whether a fragment is an entity or not, and what class it may belong to, largely depend on the internal structure of the fragment itself as well as the left and right contexts in which it appears. To a large extent, the meaning and spelling of the underlying fragment are informative to distinguish named entities from the rest of the text. Contexts play a very important role in NER or MD when it involves multi-sense words/phrases or out-of-vocabulary (OOV) words.

As shown in Figure 1, our proposed **FOFE-NER** method will examine all possible fragments in text (up to a certain length) one by one. For each fragment, it uses the FOFE method to fully encode the underlying fragment itself, its left context and right context into some fixed-size representations, which are in turn fed to an FFNN to predict whether the current fragment is NOT a valid entity mention (*NONE*), or its correct entity type (*PER*, *LOC*, *ORG* and so on) if it is a valid mention. This method is appealing because the FOFE codes serves as a theoretically lossless representation of the hypothesis and its full contexts. FFNN is used as a universal approximator to map from text to the entity labels.

In this work, we use FOFE to explore both word-level and character-level features for each fragment and its contexts.

4.1 Word-level Features

FOFE-NER generates several word-level features for each fragment hypothesis and its left and right contexts as follows:

- Bag-of-word (BoW) of the fragment, e.g.

bag-of-word vector of 'Toronto', 'Maple' and 'Leafs' in Figure 1.

- FOFE code for left context including the fragment, e.g. FOFE code of the word sequence of "... *puck from space for the Toronto Maple Leafs*" in Figure 1.
- FOFE code for left context excluding the fragment, e.g. the FOFE code of the word sequence of "... *puck from space for the*" in Figure 1..
- FOFE code for right context including the fragment, e.g. the FOFE code of the word sequence of "... *against opener home ' Leafs Maple Toronto*" in Figure 1.
- FOFE code for right context excluding the fragment, e.g. the FOFE code of the word sequence of "... *against opener home* " in Figure 1.

Moreover, all of the above word features are computed for both case-sensitive words in raw text as well as case-insensitive words in normalized lower-case text. These FOFE codes are projected to lower-dimension dense vectors based on two projection matrices, $\mathbf{W}_s$ and $\mathbf{W}_i$, for case-sensitive and case-insensitive FOFE codes respectively. These two projection matrices are initialized by word embeddings trained by *word2vec*, and fine-tuned during the learning of the neural networks.

Due to the recursive computation of FOFE codes in eq.(1), all of the above FOFE codes can be jointly computed for one sentence or document in a very efficient manner.

4.2 Character-level Features

On top of the above word-level features, we also augment character-level features for the underlying segment hypothesis to further model its morphological structure. For the example in Figure 1, the current fragment, *Toronto Maple Leafs*, is considered as a sequence of case-sensitive characters, i.e. "{'T', 'o', ..., 'f' , 's' }", we then add the following character-level features for this fragment:

- Left-to-right FOFE code of the character sequence of the underlying fragment. That is the FOFE code of the sequence, "*'T', 'o', ..., 'f' , 's'* ".
- Right-to-left FOFE code of the character sequence of the underlying fragment. That is the FOFE code of the sequence, "*'s' , 'f' , ..., 'o', 'T'* ".

These case-sensitive character FOFE codes are also projected by another character embedding matrix, which is randomly initialized and fine-tuned during model training.

Alternatively, we may use the character CNNs, as described in Section 3.3, to generate character-level features for each fragment hypothesis as well.

5 Training and Decoding Algorithm

Obviously, the above **FOFE-NER** model will take each sentence of words, $S = [w_1, w_2, w_3, ..., w_m]$, as input, and examine all continuous sub-sequences $[w_i, w_{i+1}, w_{i+2}, ..., w_j]$ up to n words in S for possible entity types. All sub-sequences longer than n words are considered as non-entities in this work.

When we train the model, based on the entity labels of all sentences in the training set, we will generate many sentence fragments up to n words. These fragments fall into three categories:

- Exact-match with an entity label, e.g., the fragment "*Toronto Maple Leafs*" in the previous example.
- Partial-overlap with an entity label, e.g., "*for the Toronto*".
- Disjoint with all entity label, e.g. "*from space for*".

For all exact-matched fragments, we generate the corresponding outputs based on the types of the matched entities in the training set. For both partial-overlap and disjoint fragments, we introduce a new output label, **NONE**, to indicate that these fragments are not a valid entity. Therefore, the output nodes in the neural networks contains all entity types plus a rejection option denoted as **NONE**.

During training, we implement a producer-consumer software design such that a thread fetches training examples, computes all FOFE codes and packs them as a mini-batch while the other thread feeds the mini-batches to neural networks and adjusts the model parameters and all projection matrices. Since "partial-overlap" and "disjoint" significantly outnumber "exact-match", they are down-sampled so as to balance the data set.

During inference, all fragments not longer than

n words are all fed to **FOFE-NER** to compute their scores over all entity types. In practice, these fragments can be packed as one mini-batch so that we can compute them in parallel on GPUs. As the NER result, the **FOFE-NER** model will return a subset of fragments only if: i) they are recognized as a valid entity type (not **NONE**); AND ii) their NN scores exceed a global pruning threshold.

Occasionally, some partially-overlapped or nested fragments may occur in the above pruned prediction results. We can use one of the following simple post-processing methods to remove overlappings from the final results:

1. *highest-first*: We check every word in a sentence. If it is contained by more than one fragment in the pruned results, we only keep the one with the maximum NN score and discard the rest.

2. *longest-first*: We check every word in a sentence. If it is contained by more than one fragment in the pruned results, we only keep the longest fragment and discard the rest.

Either of these strategies leads to a collection of non-nested, non-overlapping, non-NONE entity labels.

In some tasks, it may require to label all nested entities. This has imposed a big challenge to the sequence labeling methods. However, the above post-processing can be slightly modified to generate nested entities' labels. In this case, we first run either *highest-first* or *longest-first* to generate the first round result. For every entity survived in this round, we will recursively run either *highest-first* or *longest-first* on all entities in the original set, which are completely contained by it. This will generate more prediction results. This process may continue to allow any levels of nesting. For example, for a sentence of "w_1 w_2 w_3 w_4 w_5", if the model first generates the prediction results after the global pruning, as ["w_2w_3", PER, 0.7], ["w_3w_4", LOC, 0.8], ["$w_1w_2w_3w_4$", ORG, 0.9], if we choose to run *highest-first*, it will generate the first entity label as ["$w_1w_2w_3w_4$", ORG, 0.9]. Secondly, we will run *highest-first* on the two fragments that are completely contained by the first one, i.e., ["w_2w_3", PER, 0.7], ["w_3w_4", LOC, 0.8], then we will generate the second nested entity label as ["w_3w_4", LOC, 0.8]. Fortunately, in any real NER and MD tasks, it is pretty rare to have overlapped predictions in the NN outputs. Therefore, the extra expense to run this recursive post-processing method is minimal.

6 Second-Pass Augmentation

As we know, CRF brings marginal performance gain to all taggers (but not limited to NER) because of the dependancies (though fairly weak) between entity types. We may easily add this level of information to our model by introducing another pass of **FOFE-NER**. We call it *2nd-pass FOFE-NER*.

In 2nd-pass FOFE-NER, another set of model is trained on outputs from the first-pass **FOFE-NER**, including all predicted entities. For example, given a sentence

$$S = [w_1, w_2, ...w_i, ...w_j, ...w_n]$$

and an underlying word segment $[w_i, ..., w_j]$ in the second pass, every predicted entity outside this segment is substituted by its entity type predicted from the first pass. For example, in the first pass, a sentence like *"Google has also recruited Fei-Fei Li, director of the AI lab at Stanford University."* is predicted as: *"<ORG> has also recruited Fei-Fei Li, director of the AI lab at <ORG>."* In 2nd-pass FOFE-NER, when examining the segment "*Fei-Fei Li*", the predicted entity types *<ORG>* are used to replace the actual named entities. The 2nd-pass FOFE-NER model is trained on the outputs of the first pass, where all detected entities are replaced by their predicted types as above.

During inference, the results returned by the 1st-pass model are substituted in the same way. The scores for each hypothesis from 1st-pass model and 2nd-pass model are linear interpolated and then decoded by either *highest-first* or *longest-first* to generate the final results of 2nd-pass FOFE-NER.

Obviously, 2nd-pass FOFE-NER may capture the semantic roles of other entities while filtering out unwanted constructs and sparse combonations. On the other hand, it enables longer context expansion, since FOFE memorizes contextual information in an unselective decaying fashion.

7 Experiments

In this section, we evaluate the effectiveness of our proposed methods on several popular NER and MD tasks, including CoNLL 2003 NER task and TAC-KBP2015 and TAC-KBP2016 Trilingual Entity Discovery and Linking (EDL) tasks.

We have made our codes available at https://github.com/xmb-cipher/fofe-ner for readers to reproduce the results in this paper.

7.1 CoNLL 2003 NER task

The CoNLL-2003 dataset (Sang and Meulder, 2003) consists of newswire from the Reuters RCV1 corpus tagged with four types of non-nested named entities: location (LOC), organization (ORG), person (PER), and miscellaneous (MISC).

The top 100,000 words, are kept as vocabulary, including punctuations. For the case-sensitive embedding, an OOV is mapped to <unk> if it contains no upper-case letter and <UNK> otherwise. We perform grid search on several hyper-parameters using a held-out dev set. Here we summarize the set of hyper-parameters used in our experiments: i) *Learning rate*: initially set to 0.128 and is multiplied by a decay factor each epoch so that it reaches 1/16 of the initial value at the end of the training; ii) *Network structure*: 3 fully-connected layers of 512 nodes with ReLU activation, randomly initialized based on a uniform distribution between $-\sqrt{\frac{6}{N_i+N_o}}$ and $\sqrt{\frac{6}{N_i+N_o}}$ (Glorot et al., 2011); iii) *Character embeddings*: 64 dimensions, randomly initialized. iv) *mini-batch*: 512; v) *Dropout rate*: initially set to 0.4, slowly decreased during training until it reaches 0.1 at the end. vi) *Number of epochs*: 128; vii)*Embedding matrices* case-sensitive and case-insensitive word embeddings of 256 dimensions, trained from Reuters RCV1; viii) We stick to the official data train-dev-test partition. ix) *Forgetting factor* $\alpha = 0.5$. [1]

We have investigated the performance of our method on the CoNLL-2003 dataset by using different combinations of the FOFE features (both word-level and character-level). The detailed comparison results are shown in Table 1. In Table 2, we have compared our best performance with some top-performing neural network systems on this task. As we can see from Table 2, our system (highest-first decoding) yields very strong performance (90.85 in F_1 score) in this task, outperforming most of neural network models reported on this dataset. More importantly, we have not used any hand-crafted features in our systems, and all features (either word or char level) are automatically derived from the data. Highest-first and longest-first perform similarly. In (Chiu and Nichols, 2016)[2], a slightly better performance (91.62 in F_1 score) is reported but a customized gazetteer is used in theirs.

7.2 KBP2015 EDL Task

Given a document collection in three languages (English, Chinese and Spanish), the KBP2015 trilingual EDL task (Ji et al., 2015) requires to automatically identify entities **(including nested entities)** from a source collection of textual documents in multiple languages as in Table 3, and classify them into one of the following pre-defined five types: Person (PER), Geo-political Entity (GPE), Organization (ORG), Location (LOC) and Facility (FAC). The corpus consists of news articles and discussion forum posts published in recent years, related but non-parallel across languages.

Three models are trained and evaluated independently. Unless explicitly listed, hyperparameters follow those used for CoNLL2003 as described in section 7.1 and 2nd-pass model is not used. Three sets of word embeddings of 128 dimensions are derived from English Gigaword (Parker et al., 2011), Chinese Gigaword (Graff and Chen, 2005) and Spanish Gigaword (Mendonca et al., 2009) respectively. Some language-specific modifications are made:

- **Chinese**: Because Chinese segmentation is not reliable, we label Chinese at character level. The analogous roles of case-sensitive word-embedding and case-sensitive word-embedding are played by character embedding and word-embedding in which the character appears. Neither Char FOFE features nor Char CNN features are used for Chinese.
- **Spanish**: Character set of Spanish is a super set of that of English. When building character-level features, we use the mod function to hash each character's UTF8 encoding into a number between 0 (inclusive) and 128 (exclusive).

As shown in Table 4, our FOFE-based local detection method has obtained fairly strong perfor-

[1] The choice of the forgetting factor α is empirical. We've evaluated $\alpha = 0.5, 0.6, 0.7, 0.8$ on a development set in some early experiments. It turns out that $\alpha = 0.5$ is the best. As a result, $\alpha = 0.5$ is used for all NER/MD tasks throughout this paper.

[2] In their work, they have used a combination of training-set and dev-set to train the model, differing from all other systems (including ours) in Table 2.

FEATURE			P	R	F1
word-level	case-insensitive	context FOFE incl. word fragment	86.64	77.04	81.56
		context FOFE excl. word fragment	53.98	42.17	47.35
		BoW of word fragment	82.92	71.85	76.99
	case-sensitive	context FOFE incl. word fragment	88.88	79.83	84.12
		context FOFE excl. word fragment	50.91	42.46	46.30
		BoW of word fragment	85.41	74.95	79.84
char-level	Char FOFE of word fragment		67.67	52.78	59.31
	Char CNN of word fragment		78.93	69.49	73.91
all case-insensitive features			90.11	82.75	86.28
all case-sensitive features			90.26	86.63	88.41
all word-level features			92.03	86.08	88.96
all word-level & Char FOFE features			91.68	88.54	**90.08**
all word-level & Char CNN features			91.80	88.58	**90.16**
all word-level & all char-level features			93.29	88.27	**90.71**
all features + **dev set** + 5-fold cross-validation			92.58	89.31	**90.92**
all features + **2nd-pass**			92.13	89.61	**90.85**
all features + **2nd-pass** + **dev set** + 5-fold cross-validation			92.62	89.77	**91.17**

Table 1: Effect of various FOFE feature combinations on the CoNLL2003 test data.

algorithm	word	char	gaz	cap	pos	F1
CNN-BLSTM-CRF (Collobert et al., 2011)	✓	✗	✓	✓	✗	89.59
BLSTM-CRF (Huang et al., 2015)	✓	✓	✓	✓	✓	90.10
BLSTM-CRF (Rondeau and Su, 2016)	✓	✗	✓	✓	✓	89.28
BLSTM-CRF, char-CNN (Chiu and Nichols, 2016)	✓	✓	✓	✗	✗	**91.62**
Stack-LSTM-CRF, char-LSTM (Lample et al., 2016)	✓	✓	✗	✗	✗	**90.94**
this work	✓	✓	✗	✗	✗	**90.85**

Table 2: Performance (F_1 score) comparison among various neural models reported on the CoNLL dataset, and the different features used in these methods.

	English	Chinese	Spanish	ALL
Train	168	147	129	444
Eval	167	167	166	500

Table 3: Number of Documents in KBP2015

	2015 track best			ours		
	P	R	F_1	P	R	F_1
Trilingual	75.9	69.3	72.4	78.3	69.9	**73.9**
English	79.2	66.7	**72.4**	77.1	67.8	72.2
Chinese	79.2	74.8	**76.9**	79.3	71.7	75.3
Spanish	78.4	72.2	75.2	79.9	71.8	**75.6**

Table 4: Entity Discovery Performance of our method on the KBP2015 EDL evaluation data, with comparison to the best systems in KBP2015 official evaluation.

mance in the KBP2015 dataset. The overall trilingual entity discovery performance is slightly better than the best systems participated in the official KBP2015 evaluation, with 73.9 vs. 72.4 as measured by F_1 scores. Outer and inner decodings are *longest-first* and *highest-first* respectively.

7.3 KBP2016 EDL task

In KBP2016, the trilingual EDL task is extended to detect nominal mentions of all 5 entity types for all three languages. In our experiments, for simplicity, we treat nominal mention types as some extra entity types and detect them along with named entities together with a single model.

7.3.1 Data Description

No official training set is provided in KBP2016. We make use of three sets of training data:

- **Training and evaluation data in KBP2015**: as described in 7.2

LANG	NAME			NOMINAL			OVERALL			2016 BEST		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
ENG	0.898	0.789	0.840	0.554	0.336	0.418	0.836	0.680	0.750	0.846	0.710	0.772
CMN	0.848	0.702	0.768	0.414	0.258	0.318	0.789	0.625	0.698	0.789	0.737	0.762
SPA	0.835	0.778	0.806	0.000	0.000	0.000	0.835	0.602	0.700	0.839	0.656	0.736
ALL	0.893	0.759	0.821	0.541	0.315	0.398	0.819	0.639	**0.718**	0.802	0.704	0.756

Table 5: Official entity discovery performance of our methods on KBP2016 trilingual EDL track. Neither KBP2015 nor in-house data labels nominal mentions. Nominal mentions in Spanish are totally ignored since no training data is found for them.

training data	P	R	F_1
KBP2015	0.836	0.598	0.697
KBP2015 + WIKI	0.837	0.628	**0.718**
KBP2015 + in-house	0.836	0.680	**0.750**

Table 6: Our entity discovery official performance (English only) in KBP2016 is shown as a comparison of three models trained by different combinations of training data sets.

- **Machine-labeled Wikipedia (WIKI)**: When terms or names are first mentioned in a Wikipedia article they are often linked to the corresponding Wikipedia page by hyperlinks, which clearly highlights the possible named entities with well-defined boundary in the text. We have developed a program to automatically map these hyperlinks into KBP annotations by exploring the infobox (if existing) of the destination page and/or examining the corresponding Freebase types. In this way, we have created a fairly large amount of weakly-supervised trilingual training data for the KBP2016 EDL task. Meanwhile, a gazeteer is created and used in KBP2016.
- **In-house dataset**: A set of 10,000 English and Chinese documents is manually labeled using some annotation rules similar to the KBP 2016 guidelines.

We split the available data into training, validation and evaluation sets in a ratio of 90:5:5. The models are trained for 256 epochs if the in-house data is not used, and 64 epochs otherwise.

7.3.2 Effect of various training data

In our first set of experiments, we investigate the effect of using different training data sets on the final entity discovery performance. Different training runs are conducted on different combinations of the aforementioned data sources. In Table 6, we have summarized the official English entity discovery results from several systems we submitted to KBP2016 EDL evaluation round I and II. The first system, using only the KBP2015 data to train the model, has achieved 0.697 in F_1 score in the official KBP2016 English evaluation data. After adding the weakly labeled data, WIKI, we can see the entity discovery performance is improved to 0.718 in F_1 score. Moreover, we can see that it yields even better performance by using the KBP2015 data and the in-house data sets to train our models, giving 0.750 in F_1 score.

7.3.3 The official trilingual EDL performance in KBP2016

The official best results of our system are summarized in Table 5. We have broken down the system performance according to different languages and categories of entities (named or nominal). Our system, achieving 0.718 in F_1 score in the KBP2016 trilingual EDL track, ranks second among all participants. Note that our result is produced by a single system while the top system is a combination of two different models, each of which is based on 5-fold cross-validation (Liu et al., 2016).

8 Conclusion

In this paper, we propose a novel solution to NER and MD by applying FFNN on top of FOFE features. This simple local-detection based approach has achieved almost state-of-the-art performance on various NER and MD tasks, without using any external knowledge or feature engineering.

Acknowledgement

This work is supported mainly by a research donation from iFLYTEK Co., Ltd., Hefei, China, and partially by a discovery grant from Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proc. of the Sixth Workshop on Very Large Corpora*. volume 182. http://ucrel.lancs.ac.uk/acl/W/W98/W98-1118.pdf.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics* 4:357–370. https://www.aclweb.org/anthology/Q16-1026.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537. http://www.jmlr.org/papers/volume12/collobert11a/collobert11a.pdf.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*. Association for Computational Linguistics (ACL), page 25. https://doi.org/10.18653/v1/w15-3904.

X. Glorot, A. Bordes, and Y. Bengio. 2011. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics. JMLR W&CP:*. volume 15, pages 315–323. http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf.

David Graff and Ke Chen. 2005. Chinese gigaword. *LDC Catalog No.: LDC2003T09, ISBN* 1:58563–58230.

Alex Graves. 2012. Neural networks. In *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer, pages 15–35. https://doi.org/10.1007/978-3-642-24797-2.

Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4(2):251–257. https://doi.org/10.1016/0893-6080(91)90009-t.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* https://arxiv.org/abs/1508.01991.

Heng Ji, Joel Nothman, and Ben Hachey. 2015. Overview of tac-kbp2015 tri-lingual entity discovery and linking. In *Proceedings of Text Analysis Conference (TAC2015)*. http://nlp.cs.rpi.edu/paper/kbp2015.pdf.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*. Citeseer. https://arxiv.org/abs/1508.06615.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* https://arxiv.org/abs/1603.01360.

Dan Liu, Wei Lin, Shiliang Zhang, Si Wei, and Hui Jiang. 2016. Neural networks models for entity discovery and linking. *arXiv preprint arXiv:1611.03558* https://arxiv.org/abs/1611.03558.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1501–1511. http://www.aclweb.org/anthology/P15-1145.

Angelo Mendonca, David Andrew Graff, and Denise DiPersio. 2009. *Spanish gigaword second edition*. Linguistic Data Consortium.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119. https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Truc-Vien T Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2010. Kernel-based reranking for named-entity extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 901–909. http://www.anthology.aclweb.org/C/C10/C10-2104.pdf.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword. *Linguistic Data Consortium* .

Marc-Antoine Rondeau and Yi Su. 2016. LSTM-based NeuroCRFs for named entity recognition. In *Interspeech 2016*. International Speech Communication Association, pages 665–669. https://doi.org/10.21437/interspeech.2016-288.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003,*. page 142147. http://www.aclweb.org/anthology/W03-0419.

A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13(2):260–269. https://doi.org/10.1109/tit.1967.1054010.

Shiliang Zhang, Hui Jiang, Shifu Xiong, Si Wei, and Li-Rong Dai. 2016. Compact feedforward sequential memory networks for large vocabulary continuous speech recognition. In *Interspeech 2016*. International Speech Communication Association. https://doi.org/10.21437/interspeech.2016-121.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015a. A fixed-size encoding method for variable-length sequences with its application to neural network language models. *arXiv preprint arXiv:1505.01504*. https://arxiv.org/abs/1505.01504.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015b. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics (ACL). https://doi.org/10.3115/v1/p15-2081.

Vancouver Welcomes You! Minimalist Location Metonymy Resolution

Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham and Nigel Collier

Language Technology Lab
Department of Theoretical and Applied Linguistics
University of Cambridge

{mg711,mp792,nl347,nhc30}@cam.ac.uk

Abstract

Named entities are frequently used in a metonymic manner. They serve as references to related entities such as people and organisations. Accurate identification and interpretation of metonymy can be directly beneficial to various NLP applications, such as Named Entity Recognition and Geographical Parsing. Until now, metonymy resolution (MR) methods mainly relied on parsers, taggers, dictionaries, external word lists and other handcrafted lexical resources. We show how a minimalist neural approach combined with a novel predicate window method can achieve competitive results on the SemEval 2007 task on Metonymy Resolution. Additionally, we contribute with a new Wikipedia-based MR dataset called *RelocaR*, which is tailored towards locations as well as improving previous deficiencies in annotation guidelines.

1 Introduction

In everyday language, we come across many types of figurative speech. These irregular expressions are understood with little difficulty by humans but require special attention in NLP. One of these is metonymy, a type of common figurative language, which stands for the substitution of the concept, phrase or word being meant with a semantically related one. For example, in "**Moscow** traded gas and aluminium with **Beijing**.", both location names were substituted in place of governments.

Named Entity Recognition (NER) taggers have no provision for handling metonymy, meaning that this frequent linguistic phenomenon goes largely undetected within current NLP. Classification decisions presently focus on the entity using features such as orthography to infer its word sense, largely ignoring the context, which provides the strongest clue about whether a word is used metonymically. A common classification approach is choosing the N words to the immediate left and right of the entity or the whole paragraph as input to the model. However, this "greedy" approach also processes input that should in practice be ignored.

Metonymy is problematic for applications such as Geographical Parsing (Monteiro et al., 2016; Gritta et al., 2017, GP) and other information extraction tasks in NLP. In order to accurately identify and ground location entities, for example, we must recognise that metonymic entities constitute false positives and should not be treated the same way as regular locations. For example, in "**London** voted for the change.", London refers to the concept of "people" and should not be classified as a location. There are many types of metonymy (Shutova et al., 2013), however, in this paper, we primarily address metonymic location mentions with reference to GP and NER.

Contributions: (1) We investigate how to improve classification tasks by introducing a novel minimalist method called Predicate Window (PreWin), which outperforms common feature selection baselines. Our final minimalist classifier is comparable to systems which use many external features and tools. (2) We improve the annotation guidelines in MR and contribute with a new Wikipedia-based MR dataset called ReLocaR to address the training data shortage. (3) We make an annotated subset of the CoNLL 2003 (NER) Shared Task available for extra MR training data, alongside models, tools and other data.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1248–1259
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1115

2 Related Work

Some of the earliest work on MR that used an approach similar to our method (machine learning and dependency parsing) was by Nissim and Markert (2003a). The decision list classifier with backoff was evaluated using syntactic head-modifier relations, grammatical roles and a thesaurus to overcome data sparseness and generalisation problems. However, the method was still limited for classifying unseen data. Our method uses the same paradigm but adds more features, a different machine learning architecture and a better usage of the parse tree structure.

Much of the later work on MR comes from the SemEval 2007 Shared Task 8 (Markert and Nissim, 2007) and later by Markert and Nissim (2009). The feature set of Nissim and Markert (2003a) was updated to include: grammatical role of the potentially metonymic word (PMW) (such as subj, obj), lemmatised head/modifier of PMW, determiner of PMW, grammatical number of PMW (singular, plural), number of words in PMW and number of grammatical roles of PMW in current context. The winning system by Farkas et al. (2007) used these features and a maximum entropy classifier to achieve 85.2% accuracy. This was also the "leanest" system but still made use of feature engineering and some external tools. Brun et al. (2007) achieved 85.1% accuracy using local syntactical and global distributional features generated with an adapted, proprietary Xerox deep parser. This was the only unsupervised approach, based on using syntactic context similarities calculated on large corpora such as the the British National Corpus (BNC) with 100M tokens.

Nastase and Strube (2009) used a Support Vector Machine (SVM) with handcrafted features (in addition to the features provided by Markert and Nissim (2007)) including grammatical collocations extracted from the BNC to learn selectional preferences, WordNet 3.0, Wikipedia's category network, whether the entity "has-a-product" such as Suzuki and whether the entity "has-an-event" such as Vietnam (both obtained from Wikipedia). The bigger set of around 60 features and leveraging global (paragraph) context enabled them to achieve 86.1% accuracy. Once again, we draw attention to the extra training, external tools and additional feature generation.

Similar recent work by Nastase and Strube (2013) which extends that of Nastase et al. (2012) involved transforming Wikipedia into a large-scale multilingual concept network called WikiNet. By building on Wikipedia's existing network of categories and articles, their method automatically discovers new relations and their instances. As one of their extrinsic evaluations, metonymy resolution was tested. Global context (whole paragraph) was used to interpret the target word. Using an SVM and a powerful knowledge base built from Wikipedia, the highest performance to date (a 0.1% improvement from Nastase and Strube (2009)) was achieved at 86.2%, which has remained the SOTA until now.

The related work on MR so far has made limited use of dependency trees. Typical features came in the form of a head dependency of the target entity, its dependency label and its role (subj-of-win, dobj-of-visit, etc). However, other classification tasks made good use of dependency trees. Liu et al. (2015) used the shortest dependency path and dependency sub-trees successfully to improve relation classification (new SOTA on SemEval 2010 Shared Task). Bunescu and Mooney (2005) show that using dependency trees to generate the input sequence to a model performs well in relation extraction tasks. Dong et al. (2014) used dependency parsing for Twitter sentiment classification to find the words syntactically connected to the target of interest. Joshi and Penstein-Rosé (2009) used dependency parsing to explore how features based on syntactic dependency relations can be used to improve performance on opinion mining. In unsupervised lymphoma (type of cancer) classification, Luo et al. (2014) constructed a sentence graph from the results of a two-phase dependency parse to mine pathology reports for the relationships between medical concepts. Our methods also exploit the versatility of dependency parsing to leverage information about the sentence structure.

2.1 SemEval 2007 Dataset

Our main standard for performance evaluation is the SemEval 2007 Shared Task 8 (Markert and Nissim, 2007) dataset first introduced in Nissim and Markert (2003b). Two types of entities were evaluated, organisations and locations, randomly retrieved from the British National Corpus (BNC).

We only use the locations dataset, which comprises a train (925 samples) and a test (908 samples) partition. For *medium* evaluation, the classes are **literal** (geographical territories and political entities), **metonymic** (place-for-people, place-for-product, place-for-event, capital-for-government or place-for-organisation) and **mixed** (metonymic and literal frames invoked simultaneously or unable to distinguish). The metonymic class further breaks down into two levels of subclasses allowing for *fine* evaluation. The class distribution within SemEval is approx 80% literal, 18% metonymic and 2% mixed. This seems to be the approximate natural distribution of the classes for location metonymy, which we have also observed while sampling Wikipedia for our new dataset.

3 Our Approach

Our contribution broadly divides into two main parts, data and methodology. Section 3 introduces our new dataset, Section 4 introduces our new feature extraction method.

3.1 Design and Motivation

As part of our contribution, we created a new MR dataset called ReLocaR (Real Location Retrieval), partly due to the lack of quality annotated train/test data and partly because of the shortcomings with the SemEval 2007 dataset (see Section 3.2). Our corpus is designed to evaluate the capability of a classifier to distinguish **literal**, **metonymic** and **mixed** location mentions. In terms of dataset size, ReLocaR contains 1,026 training and 1,000 test instances. The data was sampled using Wikipedia's Random Article API[1]. We kept the sentences, which contained at least one of the places from a manually compiled list[2] of countries and capitals of the world. The natural distribution of literal versus metonymic examples is approximately 80/20 so we had to discard the excess literal examples during sampling to balance the classes.

3.2 ReLocaR - Improvements over SemEval

1. We do not break down the metonymic class further as the distinction between the subclasses is subtle and hard to agree on.

2. The distribution of the three classes in ReLocaR (literal, metonymic, mixed) is approximately (49%, 49%, 2%) eliminating the high bias (80%, 18%, 2%) of SemEval. We will show how such a high bias transpires in the test results (Section 5).

3. We have reviewed the annotation of the test partition and found that we disagreed with up to 11% of the annotations. Zhang and Gelernter (2015) disagreed with the annotation 8% of the time. Poibeau (2007) also challenged some annotation decisions. ReLocaR was annotated by 4 trained linguists (undergraduate and graduate) and 2 computational linguists (authors). Linguists were independently instructed (see section 3.3) to assign one of the two classes to each example with little guidance. We leveraged their linguistic training and expertise to make decisions rather than imposing some specific scheme. Unresolved sentences would receive the mixed class label.

4. The most prominent difference is a small change in the annotation scheme (after independent linguistic advice). The SemEval 2007 Task 8 annotation scheme (Markert and Nissim, 2007) considers the political entity interpretation a literal reading. It suggests that in "**Britain**'s current account deficit...", Britain refers to a literal location, rather than a government (which is an organisation). This is despite acknowledging that "The locative and the political sense is often distinguished in dictionaries as well as in the ACE annotation scheme...". In ReLocaR datasets, we consider a political entity a metonymic reading.

3.2.1 Why government is not a location

A government/nation/political entity is semantically much closer to Organisation/Person than a Location. "**Moscow** talks to **Beijing**." does not tell us *where* this is happening. It most likely means a politician is talking to another politician. These are not places but people and/or groups. It is paramount to separate references to "inanimate" places from references to "animate" entities.

3.3 Annotation Guidelines (Summary)

ReLocaR has three classes, **literal**, **metonymic** and **mixed**. Literal reading comprises territorial interpretations (the geographical territory, the land, soil and physical location) i.e. inanimate places that serve to point to a set of coordinates (where something might be located and/or happening) such as "The treaty was signed in **Italy**.", "Peter comes from **Russia**.", "**Britain**'s

[1] https://www.mediawiki.org/wiki/API:Random
[2] https://github.com/milangritta/Minimalist-Location-Metonymy-Resolution/data/locations.txt

Andy Murray won the Grand Slam today.", "**US** companies increased exports by 50%.", "**China**'s artists are among the best in the world." or "The reach of the transmission is as far as **Brazil**.".

A metonymic reading is any location occurrence that expresses animacy (Coulson and Oakley, 2003) such as "**Jamaica**'s indifference will not improve the negotiations.", "**Sweden**'s budget deficit may rise next year.". The following are other metonymic scenarios: a location name, which stands for any persons or organisations associated with it such as "We will give aid to **Afghanistan**.", a location as a product such as "I really enjoyed that delicious **Bordeaux**.", a location posing as a sports team "**India** beat **Pakistan** in the playoffs.", a governmental or other legal entity posing as a location "**Zambia** passed a new justice law today.", events acting as locations "**Vietnam** was a bad experience for me".

The mixed reading is assigned in two cases: either both readings are invoked at the same time such as in "The Central European country of **Slovakia** recently joined the EU." or there is not enough context to ascertain the reading i.e. both are plausible such as in "We marvelled at the art of ancient **Mexico**.". In difficult cases such as these, the mixed class is assigned.

3.4 Inter-Annotator Agreement

We give the IAA for the test partition only. The whole dataset was annotated by the first author as the main annotator. Two pairs of annotators (4 linguists) then labelled 25% of the dataset each for a 3-way agreement. The agreement before adjudication was 91% and 93%, 97.2% and 99.2% after adjudication (for pair one and two respectively). The other 50% of sentences were then once again labelled by the main annotator with a 97% agreement with self. The remainder of the sentences (unable to agree on among annotators even after adjudication) were labelled as a mixed class (1.8% of all sentences).

3.5 CoNLL 2003 and MR

We have also annotated a small subset of the CoNLL 2003 NER Shared Task data for metonymy resolution (locations only). Respecting the Reuters RCV1 Corpus (Lewis et al., 2004) distribution permissions[3], we make only a heavily processed subset available on GitHub[4]. There are 4,089 positive (literal) and 2,126 negative (metonymic) sentences to assist with algorithm experimentation and model prototyping. Due to the lack of annotated training data for MR, this is a valuable resource. The data was annotated by the first author, there are no IAA figures.

4 Methodology

4.1 Predicate Window (PreWin)

Through extensive experimentation and observation, we arrived at the intuition behind PreWin, our novel feature extraction method. The classification decision of the class of the target entity is mostly informed not by the whole sentence (or paragraph), rather it is a small and focused "predicate window" pointed to by the entity's head dependency. In other words, most of the sentence is not only superfluous for the task, it actually lowers the accuracy of the model due to irrelevant input. This is particularly important in metonymy resolution as the entity's surface form is not taken into consideration, only its context.

In Figure 1, we show the process of extracting the Predicate Window from a sample sentence (more examples are available in the Appendix). We start by using the SpaCy dependency parser by Honnibal and Johnson (2015), which is the fastest in the world, open source and highly customisable. Each dependency tree provides the following features: dependency labels and entity head dependency. Rather than using most of the tree, we only use a single local head dependency relationship to point to the predicate. Leveraging a dependency parser helps PreWin with selecting the minimum relevant input to the model while discarding irrelevant input, which may cause the neural model to behave unpredictably. Finally, the entity itself is never used as input in any of our methods, we only rely on context.

PreWin then extracts up to 5 words and their dependency labels starting at the **head** of the entity (see the next paragraph for exceptions), going in the away (from the entity) direction. The method always skips the **conjunct** ("and", "or")

[3]http://trec.nist.gov/data/reuters/reuters.html
[4]https://github.com/milangritta/Minimalist-Location-Metonymy-Resolution

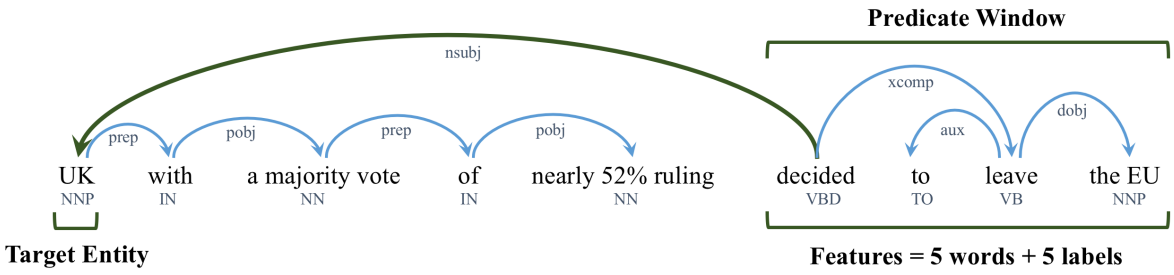


Figure 1: The predicate window starts at the head of the target entity and ends up to 4 words further, going away from the entity. The "conj" relations are always skipped. In the above example, the head of "UK" is "decided" so PreWin takes 5 words plus dependency labels as the input to the model. The left-hand side input to the model is empty and is set to zeroes (see Figure 2 for a full model diagram).

relationships in order to find the predicate (see Figure 3 in the Appendix for a visual example of why this is important). The reason for the choice of 5 words is the balance between too much input, feeding the model with less relevant context and just enough context to capture the necessary semantics. We have experimented with lengths of 3-10 words, however 5 words typically achieved the best results.

The following are the three types of exceptions when the output will not start with the head of the entity. In these cases, PreWin will include the neighbouring word as well. In a sentence "The pub is located in southern **Zambia**.", the head of the entity is "in", however in this case PreWin will include "southern" (*adjectival modifier*) as this carries important semantics for the classification. Similarly, PreWin will also include the neighbouring *compound noun* as in: "Lead coffins were very rare in colonial **America**.", the output will include "colonial" as a feature plus the next four words. In another sentence: "**Vancouver**'s security is the best in the world.', PreWin will include the "**'s**" (*case*) plus the next four words continuing from the head of the entity (the word "security").

4.2 Neural Network Architecture

The output of PreWin is used to train the following machine learning model. We decided to use the Long Short Term Memory (LSTM) architecture by Keras[5] (Chollet, 2015). Two LSTMs are used, one for the left and right side (up to 5 words each). Two fully connected (dense) layers are used for the left and right dependency relation labels (up to 5 labels each, encoded as one-hot). The full architecture is available in the Appendix, please see Figure 2. You can download the models and data from GitHub[6]. LSTMs are excellent at processing language sequences (Hochreiter and Schmidhuber, 1997; Sak et al., 2014; Graves et al., 2013), which is why we use this architecture. It allows the model to encode the word sequences, preserve important word order and provide superior classification performance. Both the Multilayer Perceptron and the Convolutional Neural Network were consistently inferior (typically 5% - 10% lower accuracy) in our earlier performance comparisons. For all experiments, we used a vocabulary of the first (most frequent) 100,000 word vectors in GloVe[7] (Pennington et al., 2014). Finally, unless explicitly stated otherwise, the standard dimension of word embeddings was 50, which we found to work best.

4.3 "Immediate" Baseline

A common approach in lexical classification tasks is choosing the 5 to 10 words to the immediate right and left of the entity as input to a model (Mikolov et al., 2013; Mesnil et al., 2013; Baroni et al., 2014; Collobert et al., 2011). We evaluate this method (its 5 and 10-word variant) alongside PreWin and Paragraph.

4.4 Paragraph Baseline

The paragraph baseline method extends the "immediate" one by taking 50 words from each side of the entity as the input to the classifier. In practice, this extends the feature window to include extra-sentential evidence in the paragraph. This ap-

[5]https://keras.io/

[6]https://github.com/milangritta/Minimalist-Location-Metonymy-Resolution

[7]http://nlp.stanford.edu/projects/glove/

proach is also popular in machine learning (Melamud et al., 2016; Zhang et al., 2016).

4.5 Ensemble of Models

In addition to a single best performing model, we have combined several models trained on different data and/or using different model configurations. For the SemEval test, we combined three separate models trained on the newly annotated CoNLL dataset and the training data for SemEval. For the ReLocaR test, we once again let three models vote, trained on CooNLL and ReLocaR data.

5 Results

We evaluate all methods using three datasets for **training** (ReLocaR, SemEval, CoNLL) and two for **testing** (ReLocaR, SemEval). Due to inherent randomness in the deep learning libraries, we performed 10 runs for each setup and averaged the figures (we also report standard deviation).

5.1 Metrics and Significance

Following the SemEval 2007 convention, we use two metrics to evaluate performance, accuracy and f-scores (for each class). We only evaluate at the **coarse level**, which means literal versus non-literal (metonymic and mixed are merged into one class). In terms of statistical significance, our best score on the SemEval dataset (908 samples) is not significant at the 95% confidence level. However, the accuracy improvements of PreWin over the common baselines are highly statistically significant with 99.9%+ confidence.

5.2 Predicate Window

Tables 1 and 2 show PreWin performing consistently better than other baselines, in many instances, significantly better and with fewer words (smaller input). The standard deviation is also lower for PreWin meaning more stable test runs. Compared with the 5 and 10 window "immediate" baseline, which is the common approach in classification, PreWin is more discriminating with its input. Due to the linguistic variety and the myriad of ways the target word sense can be triggered in a sentence, it is not always the case that the 5 or 10 nearest words inform us of the target entity's meaning/type. We ought to ask what else is being expressed in the same 5 to 10-word window?

Conventional classification methods (Immediate, Paragraph) can also be seen as prioritising either feature precision or feature recall. **Paragraph** maximises the input sequence size, which maximises recall at the expense of including features that are either irrelevant or mislead the model, lowering precision. **Immediate** baseline maximises precision by using features close to the target entity at the expense of missing important features positioned outside of its small window, lowering recall. PreWin can be understood as an integration of both approaches. It retains high precision by limiting the size of the feature window to 5 while maximising recall by searching anywhere in the sentence, frequently outside of a limited "immediate" window.

Perhaps we can also caution against a simple adherence to Firth (1957) *"You shall know a word by the company it keeps"*. This does not appear to be the case in our experiments as PreWin regularly performs better than the "immediate" baseline. Further prototypical examples of the method can be viewed in the Appendix. Our intuition that most words in the sentence, indeed in the paragraph do not carry the semantic information required to classify the target entity is ultimately based on evidence. The model uses only a small window, linked to the entity via a head dependency relationship for the final classification decision.

5.3 Common Errors

Most of the time (typically 85% for the two datasets), PreWin is sufficient for an accurate classification. However, it does not work well in some cases. The typical 15% error rate breaks down as follows (percentages were estimated based on extensive experimentation and observation):

Discarding important context (3%): Sometimes the 5 or 10 word "immediate" baseline method would actually have been preferred such as in the sentence "...REF in 2014 ranked **Essex** in the top 20 universities...". PreWin discards the right-hand side input, which is required in this case for a correct classification. Since "ranked" is the head of "Essex", the rest of the sentence gets ignored and the valuable context gets lost.

More complex semantic patterns (11%): Many common mistakes were due to the lack

of the model's understanding of more complex predicates such as in the following sentences: " ...of military presence of **Germany**.", "**Houston** also served as a member and treasurer of the..." or "...invitations were extended to **Yugoslavia** ...". We think this is due to a lack of training data (around 1,000 sentences per dataset). Additional examples such as "...days after the tour had exited **Belgium**." expose some of the limitations of the neural model to recognise uncommon ways of expressing a reference to a literal place. Recall that no external resources or tools were used to supplement the training/features, the model had to learn to generalise from what it has seen during training, which was limited in our experiments.

Parsing mistakes (1%): were less common though still present. It is important to choose the right dependency parser for the task since different parsers will often generate slightly different parse trees. We have used SpaCy[8] for all our experiments, which is a Python-based industrial strength NLP library. Sometimes, tokenisation errors for acronyms like "U.S.A." and wrongly hyphenated words may also cause parsing errors, however, this was infrequent.

Method	Training (Size)	Acc (STD)
PreWin	SemEval (925)	**62.4** (2.30)
Immediate 5	SemEval (925)	60.6 (2.34)
Immediate 10	SemEval (925)	59.2 (2.26)
Paragraph	SemEval (925)	58.0 (2.49)
PreWin	CoNLL (6,215)	**82.8** (0.46)
Immediate 5	CoNLL (6,215)	78.2 (0.61)
Immediate 10	CoNLL (6,215)	79.1 (0.76)
Paragraph	CoNLL (6,215)	79.5 (1.50)
PreWin	ReLocaR (1,026)	**83.6** (0.71)
Immediate 5	ReLocaR (1,026)	81.4 (1.34)
Immediate 10	ReLocaR (1,026)	81.3 (1.44)
Paragraph	ReLocaR (1,026)	80.0 (2.25)
Ensemble	ReLocaR/CoNLL	**84.8** (0.34)

Table 1: Results for ReLocaR data. Figures are averaged over 10 runs. STD - Standard deviation.

5.4 Flexibility of Neural Model

The top accuracy figures for ReLocaR are almost identical to SemEval. The highest single model accuracy for ReLocaR was 83.6% (84.8% with Ensemble), which was within 0.5% of the equivalent methods for SemEval (83.1%, 84.6% for Ensemble). Both were achieved using the same methods (PreWin or Ensemble), neural architecture and size of corpora. When the models were trained on the CoNLL data, the accuracies were 82.8% and 79.5%. However, when the models trained on ReLocaR and tested on SemEval (and vice versa), accuracy dropped to between 62.4% and 69% showing that what was learnt does not seem to transfer well to another dataset. We think the reason for this is the difference in annotation guidelines; the government is a **metonymic** reading, not a literal one. This causes the model to make more mistakes.

Method	Training (Size)	Acc (STD)
PreWin	SemEval (925)	**83.1** (0.64)
Immediate 5	SemEval (925)	81.3 (1.11)
Immediate 10	SemEval (925)	81.9 (0.89)
Paragraph	SemEval (925)	81.3 (0.88)
PreWin	CoNLL (6,215)	**79.5** (0.34)
Immediate 5	CoNLL (6,215)	77.8 (1.47)
Immediate 10	CoNLL (6,215)	77.8 (1.22)
Paragraph	CoNLL (6,215)	77.2 (2.10)
PreWin	ReLocaR (1,026)	**69.0** (3.13)
Immediate 5	ReLocaR (1,026)	63.6 (5.42)
Immediate 10	ReLocaR (1,026)	64.2 (4.12)
Paragraph	ReLocaR (1,026)	64.4 (7.76)
Nastase et al.	SemEval (925)	**86.2** (N/A)
Ensemble	SemEval/CoNLL	84.6 (0.43)

Table 2: Results for SemEval data. Figures are averaged over 10 runs. STD - standard deviation.

5.5 Ensemble Method

The highest accuracy and f-scores were achieved with the ensemble method for both datasets. We combined three models (previously described in section 4.5) for SemEval to achieve 84.6% accuracy and three models for ReLocaR to achieve 84.8% for the new dataset. Training separate models with different parameters and/or on different datasets does increase classification capability as various models learn distinct aspects of the task, enabling the 1.2 - 1.5% improvement.

5.6 Dimensionality of Word Embeddings

We found that increasing dimension size (up to 300) did not materially improve performance.

[8] https://spacy.io/

The neural network tended to overfit, even with fewer epochs, the results were comparable to our default 50-dimensional embeddings. We posit that fewer dimensions of the distributed word representations force the abstraction level higher as the meaning of words must be expressed more succinctly. We think this helps the model generalise better, particularly for smaller datasets. Lastly, learning word embeddings from scratch on datasets this small (around 1,000 samples) is possible but impractical, the performance typically decreases by around 5% if word embeddings are not initialised first.

Dataset / Method	Literal	Non-Literal
SemEval / PreWin	90.6	57.3
SemEval / SOTA	**91.6**	**59.1**
ReLocaR / PreWin	84.4	84.8

Table 3: Per class f-scores - all figures obtained using the Ensemble method, averaged over 10 runs. Note the model class bias for SemEval.

5.7 F-Scores and Class Imbalance

Table 3 shows the SOTA f-scores, our best results for SemEval 2007 and the best f-scores for ReLocaR. The class imbalance inside SemEval (80% literal, 18% metonymic, 2% mixed) is reflected as a high bias in the final model. This is not the case with ReLocaR and its 49% literal, 49% metonymic and 2% mixed ratio of 3 classes. The model was equally capable of distinguishing between literal and non-literal cases.

5.8 Another baseline

There was another baseline we tested, however, it was not covered anywhere so far because of its low performance. It was a type of extreme parse tree pruning, during which most of the sentence gets discarded and we only retain 3 to 4 content words. The method uses non-local (long range) dependencies to construct a short input sequence. However, the method was a case of ignoring too many relevant words and accuracy was fluctuating in the mid-60% range, which is why we did not report the results. However, it serves to further justify the choice of 5 words as the predicate window as fewer words caused the model to underperform.

6 Discussion

6.1 NER, GP and Metonymy

We think the next frontier is a NER tagger, which actively handles metonymy. The task of labelling entities should be mainly driven by context rather than the word's surface form. If the target entity looks like "**London**", this should not mean the entity is automatically a location. Metonymy is a frequent linguistic phenomenon (around 20% of location mentions are metonymic, see section 3.1) and could be handled by NER taggers to enable many innovative downstream NLP applications.

Geographical Parsing is a pertinent use case. In order to monitor/mine text documents for geographical information only, the current NER technology does not have a solution. We think it is incorrect for any NER tagger to label "Vancouver" as a location in "*Vancouver welcomes you!*". A better output might be something like the following: *Vancouver = location AND metonymy = True*. This means Vancouver is usually a location but is used metonymically in this case. How this information is used will be up to the developer. Organisations behaving as persons, share prices or products are but a few other examples of metonymy.

6.2 Simplicity and Minimalism

Previous work in MR such as most of the SemEval 2007 participants (Farkas et al., 2007; Nicolae et al., 2007; Leveling, 2007; Brun et al., 2007; Poibeau, 2007) and the more recent contributions used a selection of many of the following features/tools for classification: handmade trigger word lists, WordNet, VerbNet, FrameNet, extra features generated/learnt from parsing Wikipedia (approx 3B words) and BNC (approx 100M words), custom databases, handcrafted features, multiple (sometimes proprietary) parsers, Levin's verb classes, 3,000 extra training instances from a corpus called MAScARA[9] by Markert and Nissim (2002) and other extra resources including the SemEval Task 8 features. We managed to achieve comparable performance with a small neural network typically trained in no more than 5 epochs, minimal training data, a basic dependency parser and the new PreWin method by being highly discriminating in choosing signal over noise.

[9] http://homepages.inf.ed.ac.uk/mnissim/mascara/

7 Conclusions and Future Work

We showed how a minimalist neural approach can replace substantial external resources, handcrafted features and how the PreWin method can even ignore most of the paragraph where the entity is positioned and still achieve competitive performance in metonymy resolution. The pressing new question is: "How much better the performance could have been if our method availed itself of the extra training data and resources used by previous works?" Indeed this may be the next research chapter for PreWin.

We discussed how tasks such as Geographical Parsing can benefit from "metonymy-enhanced" NER tagging. We have also presented a case for better annotation guidelines for MR (after consulting with a number of linguists), which now means that a government is not a literal class, rather it is a metonymic one. We fully agreed with the rest of the previous annotation guidelines. We also introduced ReLocaR, a new corpus for (location) metonymy resolution and encourage researchers to make effective use of it (including the additional CoNLL 2003 subset we annotated for metonymy).

Future work may involve testing PreWin on an NER task to see if and how it can generalise to a different classification task and how the results compare to the SOTA and similar methods such as that of Collobert et al. (2011) using the CoNLL 2003 NER datasets. Word Sense Disambiguation (Yarowsky, 2010; Pilehvar and Navigli, 2014) with neural networks (Melamud et al., 2016) is another related classification task suitable for testing PreWin. If it does perform better, this will be of considerable interest to classification research (and beyond) in NLP.

Acknowledgments

We gratefully acknowledge the funding support of the Natural Environment Research Council (NERC) PhD Studentship NE/M009009/1 (Milan Gritta, DREAM CDT), EPSRC (Nigel Collier and Nut Limsopatham - Grant No. EP/M005089/1) and MRC (Mohammad Taher Pilehvar) Grant No. MR/M025160/1 for PheneBank.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*. pages 238–247.

Caroline Brun, Maud Ehrmann, and Guillaume Jacquet. 2007. XRCE-M: A hybrid system for named entity metonymy resolution. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. pages 488–491.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*. pages 724–731.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Seana Coulson and Todd Oakley. 2003. Metonymy and conceptual blending. *Pragmatics and beyond - new series* pages 51–80.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL (2)*. pages 49–54.

Richárd Farkas, Eszter Simon, György Szarvas, and Dániel Varga. 2007. Gyder: maxent metonymy resolution. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. pages 161–164.

J. R. Firth. 1957 1952-59:1–32.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, pages 6645–6649.

Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham, and Nigel Collier. 2017. What's missing in geographical parsing? *Language Resources and Evaluation* pages 1–21.

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

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1373–1378. https://aclweb.org/anthology/D/D15/D15-1162.

Mahesh Joshi and Carolyn Penstein-Rosé. 2009. Generalizing dependency features for opinion mining. In *Proceedings of the ACL-IJCNLP 2009 conference short papers*. pages 313–316.

Johannes Leveling. 2007. Fuh (fernuniversität in hagen): Metonymy recognition using different kinds of context for a memory-based learner. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. pages 153–156.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research* 5(Apr):361–397.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. *arXiv preprint arXiv:1507.04646* .

Yuan Luo, Aliyah R Sohani, Ephraim P Hochberg, and Peter Szolovits. 2014. Automatic lymphoma classification with sentence subgraph mining from pathology reports. *Journal of the American Medical Informatics Association* 21(5):824–832.

Katja Markert and Malvina Nissim. 2002. Metonymy resolution as a classification task. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. pages 204–213.

Katja Markert and Malvina Nissim. 2007. Semeval-2007 task 08: Metonymy resolution at semeval-2007. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. pages 36–41.

Katja Markert and Malvina Nissim. 2009. Data and models for metonymy resolution. *Language Resources and Evaluation* 43(2):123–138.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of CONLL*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Bruno R Monteiro, Clodoveu A Davis, and Fred Fonseca. 2016. A survey on the geographic scope of textual documents. *Computers & Geosciences* 96:23–34.

Vivi Nastase, Alex Judea, Katja Markert, and Michael Strube. 2012. Local and global context for supervised and unsupervised metonymy resolution. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 183–193.

Vivi Nastase and Michael Strube. 2009. Combining collocations, lexical and encyclopedic knowledge for metonymy resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*. pages 910–918.

Vivi Nastase and Michael Strube. 2013. Transforming wikipedia into a large scale multilingual concept network. *Artificial Intelligence* 194:62–85.

Cristina Nicolae, Gabriel Nicolae, and Sanda Harabagiu. 2007. Utd-hlt-cg: Semantic architecture for metonymy resolution and classification of nominal relations. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. pages 454–459.

Malvina Nissim and Katja Markert. 2003a. Syntactic features and word similarity for supervised metonymy resolution. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. pages 56–63.

Malvina Nissim and Katja Markert. 2003b. Syntactic features and word similarity for supervised metonymy resolution. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. pages 56–63.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Mohammad Taher Pilehvar and Roberto Navigli. 2014. A large-scale pseudoword-based evaluation framework for state-of-the-art word sense disambiguation. *Computational Linguistics* .

Thierry Poibeau. 2007. Up13: Knowledge-poor methods (sometimes) perform poorly. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. pages 418–421.

Haşim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128* .

Ekaterina Shutova, Jakub Kaplan, Simone Teufel, and Anna Korhonen. 2013. A computational model of logical metonymy. *ACM Transactions on Speech and Language Processing (TSLP)* 10(3):11.

David Yarowsky. 2010. Word sense disambiguation. In *Handbook of Natural Language Processing, Second Edition*, Chapman and Hall/CRC, pages 315–338.

Jinchao Zhang, Fandong Meng, Mingxuan Wang, Daqi Zheng, Wenbin Jiang, and Qun Liu. 2016. Is local window essential for neural network based chinese word segmentation? In *China National Conference on Chinese Computational Linguistics*. Springer, pages 450–457.

Wei Zhang and Judith Gelernter. 2015. Exploring metaphorical senses and word representations for identifying metonyms. *arXiv preprint arXiv:1508.04515* .

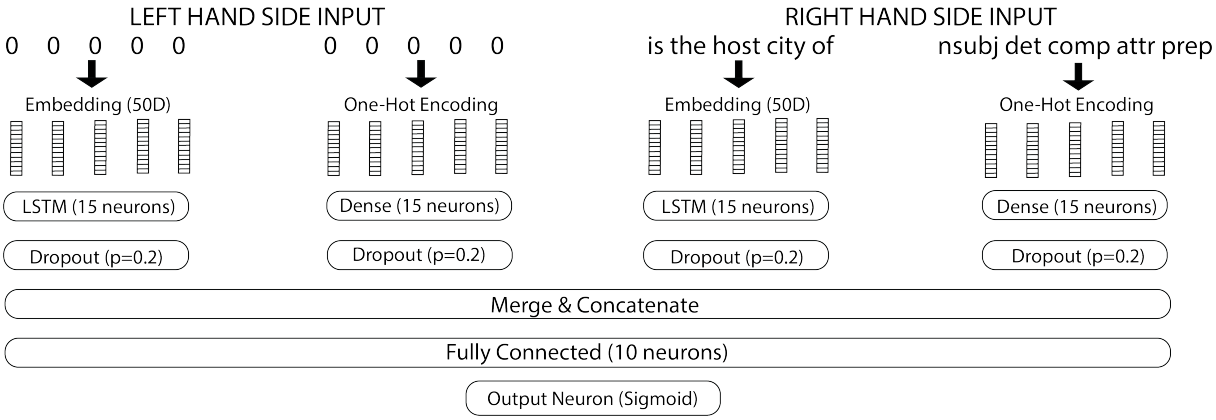


Figure 2: The neural architecture of the final model. The sentence is *Vancouver is the host city of the ACL 2017.* Small, separate sequential models are merged and trained as one. The 50-dimensional embeddings were initiated using GloVe. The *right hand* input is processed from right to left, the *left hand* input is processed from left to right. This is to emphasise the importance of the words **closer** to the entity.

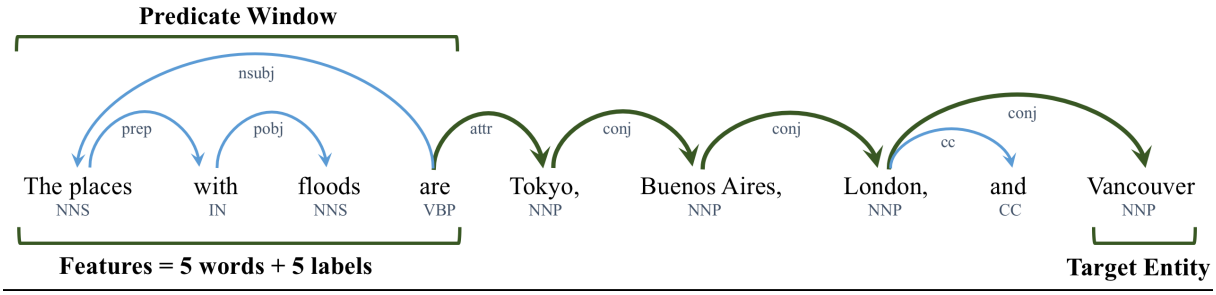


Figure 3: Why it is important for PreWin to always skip the **conjunct** dependency relation.

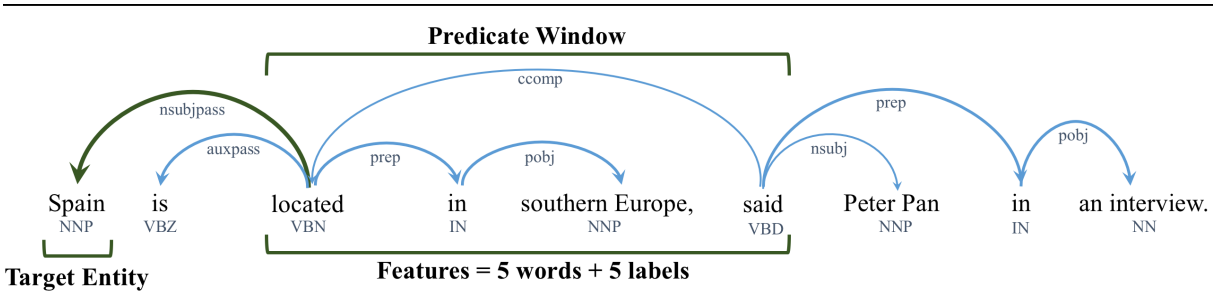


Figure 4: A lot of irrelevant input is skipped such as “is” and “Peter Pan in an interview.”.

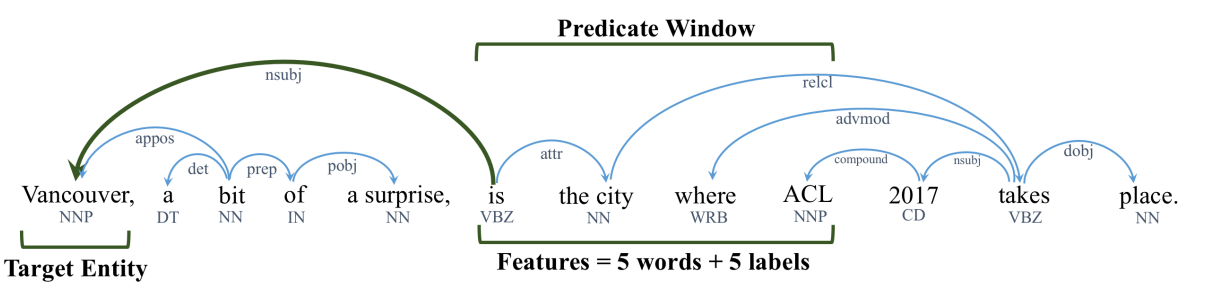


Figure 5: By looking for the predicate window, the model skips many irrelevant words.

Unifying Text, Metadata, and User Network Representations with a Neural Network for Geolocation Prediction

Yasuhide Miura[†,‡]
yasuhide.miura@fujixerox.co.jp

Motoki Taniguchi[†]
motoki.taniguchi@fujixerox.co.jp

Tomoki Taniguchi[†]
taniguchi.tomoki@fujixerox.co.jp

Tomoko Ohkuma[†]
ohkuma.tomoko@fujixerox.co.jp

[†]Fuji Xerox Co., Ltd.
[‡]Tokyo Institute of Technology

Abstract

We propose a novel geolocation prediction model using a complex neural network. Our model unifies text, metadata, and user network representations with an attention mechanism to overcome previous ensemble approaches. In an evaluation using two open datasets, the proposed model exhibited a maximum 3.8% increase in accuracy and a maximum of 6.6% increase in accuracy@161 against previous models. We further analyzed several intermediate layers of our model, which revealed that their states capture some statistical characteristics of the datasets.

1 Introduction

Social media sites have become a popular source of information to analyze current opinions of numerous people. Many researchers have worked to realize various automated analytical methods for social media because manual analysis of such vast amounts of data is difficult. Geolocation prediction is one such analytical method that has been studied widely to predict a user location or a document location. Location information is crucially important information for analyses such as disaster analysis (Sakaki et al., 2010), disease analysis (Culotta, 2010), and political analysis (Tumasjan et al., 2010). Such information is also useful for analyses such as sentiment analysis (Martínez-Cámara et al., 2014) and user attribute analysis (Rao et al., 2010) to undertake detailed region-specific analyses.

Geolocation prediction has been performed for Wikipedia (Overell, 2009), Flickr (Serdyukov et al., 2009; Crandall et al., 2009), Facebook (Backstrom et al., 2010), and Twitter (Cheng et al., 2010; Eisenstein et al., 2010). Among these sources, Twitter is often preferred because of its characteristics, which are suited for geolocation prediction. First, some tweets include geotags, which are useful as ground truth locations. Secondly, tweets include *metadata* such as timezones and self-declared locations that can facilitate geolocation prediction. Thirdly, a user network is obtainable by consideration of the interaction between two users as a network link.

Herein, we propose a neural network model to tackle geolocation prediction in Twitter. Past studies have combined text, metadata, and user network information with ensemble approaches (Han et al., 2013, 2014; Rahimi et al., 2015a; Jayasinghe et al., 2016) to achieve state-of-the-art performance. Our model combines text, metadata, and user network information using a complex neural network. Neural networks have recently shown effectiveness to capture complex representations combining simpler representations from large-scale datasets (Goodfellow et al., 2016). We intend to obtain unified text, metadata, and user network representations with an attention mechanism (Bahdanau et al., 2014) that is superior to the earlier ensemble approaches. The contributions of this paper are the following:

1. We propose a neural network model that learns unified text, metadata, and user network representations with an attention mechanism.
2. We show that the proposed model outperforms the previous ensemble approaches in two open datasets.
3. We analyze some components of the proposed model to gain insight into the unification processes of the model.

Our model specifically emphasizes geolocation prediction in Twitter to use benefits derived from the characteristics described above. However, our

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1260–1272
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1116

model can be readily extended to other social media analyses such as user attribute analysis and political analysis, which can benefit from metadata and user network information.

In subsequent sections of this paper, we explain the related works in four perspectives in Section 2. The proposed neural network model is described in Section 3 along with two open datasets that we used for evaluations in Section 4. Details of an evaluation are reported in Section 5 with discussions in Section 6. Finally, Section 7 concludes the paper with some future directions.

2 Related Works

2.1 Text-based Approach

Probability distributions of words over locations have been used to estimate the geolocations of users. Maximum likelihood estimation approaches (Cheng et al., 2010, 2013) and language modeling approaches minimizing KL-divergence (Wing and Baldridge, 2011; Kinsella et al., 2011; Roller et al., 2012) have succeeded in predicting user locations using word distributions. Topic modeling approaches to extract latent topics with geographical regions (Eisenstein et al., 2010, 2011; Hong et al., 2012; Ahmed et al., 2013) have also been explored considering word distributions.

Supervised machine learning methods with word features are also popular in text-based geolocation prediction. Multinomial Naive Bayes (Han et al., 2012, 2014; Wing and Baldridge, 2011), logistic regression (Wing and Baldridge, 2014; Han et al., 2014), hierarchical logistic regression (Wing and Baldridge, 2014), and a multilayer neural network with stacked denoising autoencoder (Liu and Inkpen, 2015) have realized geolocation prediction from text. A semi-supervised machine learning approach by Cha et al. (2015) has also been produced using a sparse-coding and dictionary learning.

2.2 User-network-based Approach

Social media often include interactions of several kinds among users. These interactions can be regarded as links that form a network among users. Several studies have used such user network information to predict geolocation. Backstrom et al. (2010) introduced a probabilistic model to predict the location of a user using friendship information in Facebook. Friend and follower information in Twitter were used to predict user locations with a most frequent friend algorithm (Davis Jr. et al., 2011), a unified descriptive model (Li et al., 2012b), location-based generative models (Li et al., 2012a), dynamic Bayesian networks (Sadilek et al., 2012), a support vector machine (Rout et al., 2013), and maximum likelihood estimation (McGee et al., 2013). Mention information in Twitter is also used with label propagation models (Jurgens, 2013; Compton et al., 2014) and an energy and social local coefficient model (Kong et al., 2014). Jurgens et al. (2015) compared nine user-network-based approaches targeting Twitter, controlling data conditions.

2.3 Metadata-based Approach

Metadata such as location fields are useful as effective clues to predict geolocation. Hecht et al. (2011) reported that decent accuracy of geolocation prediction can be achieved using location fields. Approaches to combine metadata with texts are also proposed to extend text-based approaches. Combinatory approaches such as a dynamically weighted ensemble method (Mahmud et al., 2012), polygon stacking (Schulz et al., 2013), stacking (Han et al., 2013, 2014), and average pooling with a neural network (Miura et al., 2016) have strengthened geolocation prediction.

2.4 Combinatory Approach Extending User-network-based Approach

Several attempts have been made to combine user-network-based approaches with other approaches. A text-based approach with logistic regression was combined with label propagation approaches to enhance geolocation prediction (Rahimi et al., 2015a,b, 2016). Jayasinghe et al. (2016) combined nine components including text-based approaches, metadata-based approaches, and a user-network-based approach with a cascade ensemble method.

2.5 Comparisons with Proposed Model

A model we propose in Section 3 which combines text, metadata, and user network information with a neural network, can be regarded as an alternative to approaches using text and metadata (Mahmud et al., 2012; Schulz et al., 2013; Han et al., 2013, 2014; Miura et al., 2016), approaches with text and user network information (Rahimi et al., 2015a,b), and an approach with text, metadata, and user network information (Jayasinghe et al., 2016). In Section 5, we demonstrate that our model outperforms earlier models.

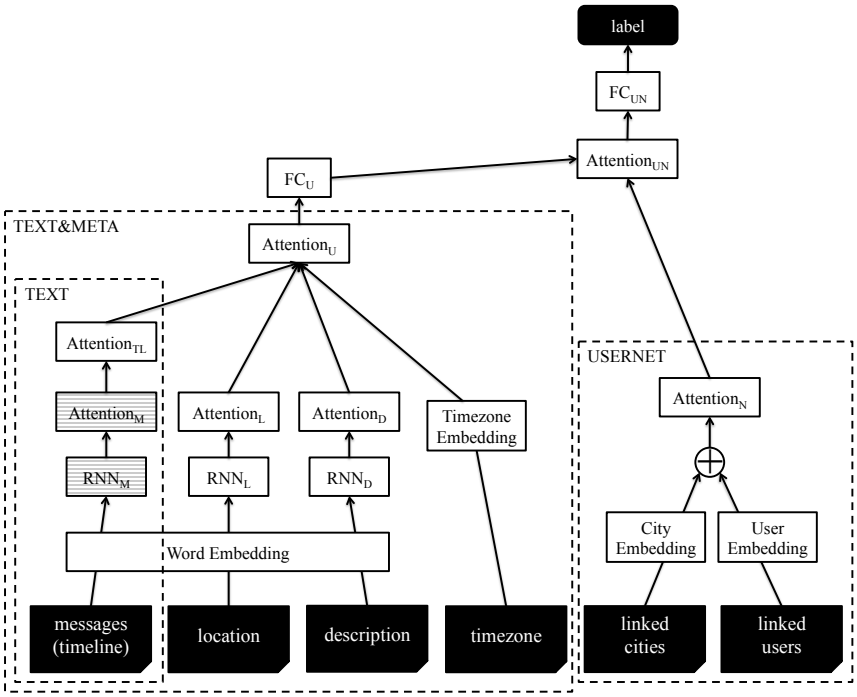


Figure 1: Overview of the proposed model. RNN denotes a recurrent neural network layer. FC denotes a fully connected layer. The striped layers are message-level processes. $\oplus$ represents element-wise addition.

In terms of machine learning methods, our model is a neural network model that shares some similarity with previous neural network models (Liu and Inkpen, 2015; Miura et al., 2016). Our model and these previous models have two key differences. First, our model integrates user network information along with other information. Secondly, our model combines text and metadata with an attention mechanism (Bahdanau et al., 2014).

3 Model

3.1 Proposed Model

Figure 1 presents an overview of our model: a complex neural network for classification with a city as a label. For each user, the model accepts inputs of messages, a location field, a description field, a timezone, linked users, and the cities of linked users.

User network information is incorporated by city embeddings and user embeddings of linked users. User embeddings are introduced along with city embeddings because linked users with city information[1] are limited. We chose to let the model learn geolocation representations of linked users directly via user embeddings. The model can be broken down to several components, details of which are described in Section 3.1.1–3.1.4.

3.1.1 Text Component

We describe the text component of the model, which is the "TEXT" section in Figure 1. Figure 2 presents an overview of the text component. The component consists of a recurrent neural network (RNN) (Graves, 2012) layer and attention layers. An input of the component is a *timeline* of a user, which consists of messages in a time sequence.

As an implementation of RNN, we used Gated Recurrent Unit (GRU) (Cho et al., 2014) with a bi-directional setting. In the RNN layer, word embeddings $\boldsymbol{x}$ of a message are processed with the following transition functions:

$$\boldsymbol{z}_t = \sigma\left(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z\right) \quad (1)$$

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{W}_r \boldsymbol{x}_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_r\right) \quad (2)$$

$$\tilde{\boldsymbol{h}}_t = \tanh\left(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{U}_h \left(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}\right) + \boldsymbol{b}_h\right) \quad (3)$$

$$\boldsymbol{h}_t = \left(\boldsymbol{1} - \boldsymbol{z}_t\right) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t \quad (4)$$

where $\boldsymbol{z}_t$ is an update gate, $\boldsymbol{r}_t$ is a reset gate, $\tilde{\boldsymbol{h}}_t$ is a candidate state, $\boldsymbol{h}_t$ is a state, $\boldsymbol{W}_z, \boldsymbol{W}_r, \boldsymbol{W}_h, \boldsymbol{U}_z, \boldsymbol{U}_r, \boldsymbol{U}_h$ are weight matrices, $\boldsymbol{b}_z, \boldsymbol{b}_r, \boldsymbol{b}_h$ are bias vectors, σ is a logistic sigmoid function, and $\odot$ is an element-wise multiplication operator. The bi-directional GRU outputs $\overrightarrow{\boldsymbol{h}}$

[1] City information are provided by a dataset. The detail of the city information is explained in Section 4.

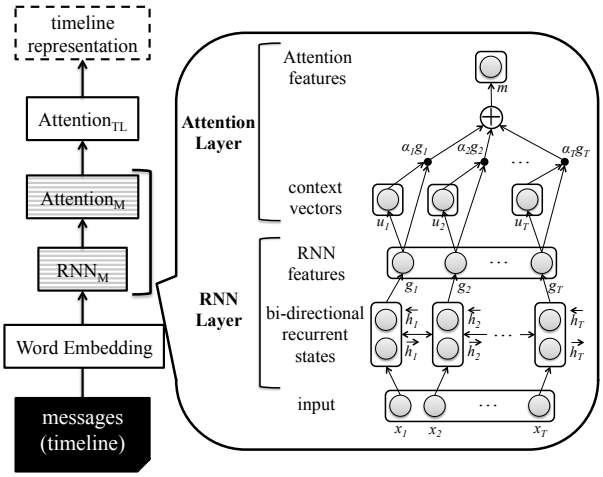


Figure 2: Overview of the text component with detailed description of $\mathrm{RNN_M}$ and $\mathrm{Attention_M}$.

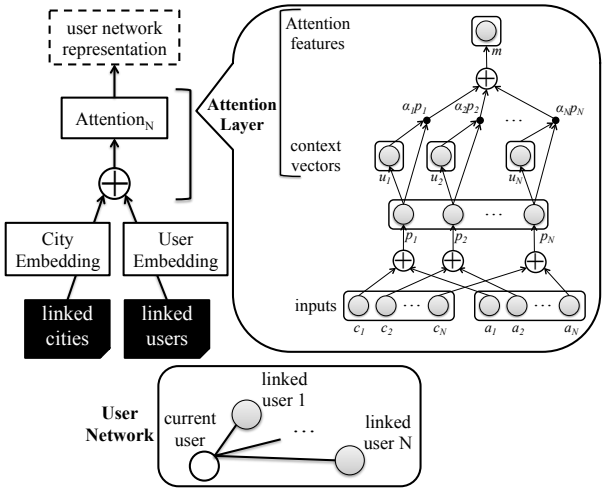


Figure 3: Overview of the user network component with a detailed description of the element-wise addition and $\mathrm{Attention_N}$.

and $\overleftarrow{\boldsymbol{h}}$ are concatenated to form $\boldsymbol{g}$ *where* $\boldsymbol{g}_t = \overrightarrow{\boldsymbol{h}_t} \| \overleftarrow{\boldsymbol{h}_t}$ and are passed to the first attention layer $\mathrm{Attention_M}$.

$\mathrm{Attention_M}$ computes a message representation $\boldsymbol{m}$ as a weighted sum of $\boldsymbol{g}_t$ with weight α_t:

$$\boldsymbol{m} = \sum_t \alpha_t \boldsymbol{g}_t \tag{5}$$

$$\alpha_t = \frac{\exp\left(\boldsymbol{v}_\alpha^T \boldsymbol{u}_t\right)}{\sum_t \exp\left(\boldsymbol{v}_\alpha^T \boldsymbol{u}_t\right)} \tag{6}$$

$$\boldsymbol{u}_t = \tanh\left(\boldsymbol{W}_\alpha \boldsymbol{g}_t + \boldsymbol{b}_\alpha\right) \tag{7}$$

where $\boldsymbol{v}_\alpha$ is a weight vector, $\boldsymbol{W}_\alpha$ is a weight matrix, and $\boldsymbol{b}_\alpha$ a bias vector. $\boldsymbol{u}_t$ is an attention context vector calculated from $\boldsymbol{g}_t$ with a single fully-connected layer (Eq. 7). $\boldsymbol{u}_t$ is normalized with softmax to obtain α_t as a probability (Eq. 6). The message representation $\boldsymbol{m}$ is passed to the second attention layer $\mathrm{Attention_{TL}}$ to obtain a timeline representation from message representations.

3.1.2 Text and Metadata Component

We describe text and metadata components of the model, which is the "TEXT&META" section in Figure 1. This component considers the following three types of metadata along with text: **location** a text field in which a user is allowed to write the user location freely, **description** a text field a user can use for self-description, and **timezone** a selective field from which a user can choose a timezone. Note that certain percentages of these fields are not available[2], and *unknown* tokens are used for inputs in such cases.

We process location fields and description fields similarly to messages using an RNN layer and an attention layer. Because there is only one location and one description per user, a second attention layer is not required, as it is in the text component. We also chose to share word embeddings among the messages, the location, and the description processes because these inputs are all textual information. For the timezone, an embedding is assigned for each timezone value. A processed timeline representation, a location representation, and a description representation are then passed to the attention layer $\mathrm{Attention_U}$ with a timezone representation. $\mathrm{Attention_U}$ combines these four representations and outputs a user representation. This combination is done as in $\mathrm{Attention_{TL}}$ with four representations as $g_1 \ldots g_4$ in Eq. 5.

3.1.3 User Network Component

We describe the user network component of the model, which is the "USERNET" section in Figure 1. Figure 3 presents an overview of the user network component. The model has two inputs *linked cities* and *linked users*. Users connected with a user network are extracted as linked users. We treat their cities[3] as linked cities. Linked cities and linked users are assigned with city embeddings $\boldsymbol{c}$ and user embeddings $\boldsymbol{a}$ respectively. $\boldsymbol{c}$ and $\boldsymbol{a}$ are then processed to output $\boldsymbol{p} = \boldsymbol{c} \oplus \boldsymbol{a}$, where $\oplus$ is an element-wise addition operator. $\boldsymbol{p}$ is then passed to the subsequent attention layer $\mathrm{Attention_N}$ to obtain a user network representa-

[2]Han et al. (2014) reported missing percentages of 19% for location, 24% for description, and 25%for timezone.

[3]A user with city information implies that the user is included in a training set.

	TwitterUS (train)	W-NUT (train)
#user	279K	782K
#tweet	23.8M	9.03M
tweet/user	85.6	11.6
#edge	3.69M	3.21M
#reduced-edge	2.11M	1.01M
reduced-edge/user	7.04	1.29
#city	339	3028

Table 1: Some properties of TwitterUS (train) and W-NUT (train). We were able to obtain approximately 70–78% of the full datasets because of accessibility changes in Twitter.

tion as in $\mathrm{Attention_U}$.

3.1.4 Model Output

An output of the text and metadata component and an output of the mention network component are further passed to the final attention layer $\mathrm{Attention_{UN}}$ to obtain a merged user representation as in $\mathrm{Attention_U}$. The merged user representation is then connected to labels with a fully connected layer $\mathrm{FC_{UN}}$.

3.2 Sub-models of the Proposed Model

SUB-NN-TEXT We prepare a sub-model SUB-NN-TEXT by adding $\mathrm{FC_U}$ and $\mathrm{FC_{UN}}$ to the text component. This sub-model can be considered as a variant of a neural network model by Yang et al. (2016), which learns a representation of hierarchical text.

SUB-NN-UNET We prepare a sub-model SUB-NN-UNET by connecting the text component and the user network component with $\mathrm{FC_U}$, $\mathrm{Attention_{UN}}$, and $\mathrm{FC_{UN}}$. This model can be regarded as a model that uses text and user network information.

SUB-NN-META We prepare a sub-model SUB-NN-META by adding $\mathrm{FC_U}$ and $\mathrm{FC_{UN}}$ to the metadata component. This model is a text-meta-based model that uses text and metadata.

4 Data

4.1 Dataset Specifications

TwitterUS The first dataset we used is TwitterUS assembled by Roller et al. (2012), which consists of 429K training users, 10K development users, and 10K test users in a North American region. The ground truth location of a user is set to the first geotag of the user in the dataset. We collected TwitterUS tweets using TwitterAPI to reconstruct TwitterUS to obtain metadata along with text. Up to date versions in November–December 2016 were used for the metadata[4]. We additionally assigned city centers to ground truth geotags using the city category of Han et al. (2012) to make city prediction possible in this dataset. TwitterUS (train) in Table 1 presents some properties related to the TwitterUS training set.

W-NUT The second dataset we used is W-NUT, a user-level dataset of the geolocation prediction shared task of W-NUT 2016 (Han et al., 2016). The dataset consists of 1M training users, 10K development users, and 10K test users. The ground truth location of a user is decided by majority voting of the closest city center. Like in TwitterUS, we obtained metadata and texts using TwitterAPI. Up to date versions in August–September 2016 were used for the metadata. W-NUT (train) in Table 1 presents some properties related to the W-NUT training set.

4.2 Construction of the User Network

We construct mention networks (Jurgens, 2013; Compton et al., 2014; Rahimi et al., 2015a,b) from datasets as user networks. To do so, we follow the approach of Rahimi et al. (2015a) and Rahimi et al. (2015b) who use uni-directional mention to set edges of a mention network. An edge is set between the two users nodes if a user mentions another user. The number of uni-directional mention edges for TwitterUS and W-NUT can be found in Table 1.

The uni-directional setting results to large numbers of edges, which often are computationally expensive to process. We restricted edges to satisfy one of the following conditions to reduce the size: (1) both users have ground truth locations or (2) one user has a ground truth location and another user is mentioned 5 times or more in a training set. The number of reduced-edges with these conditions in TwitterUS and W-NUT can be confirmed in Table 1.

5 Evaluation

5.1 Implemented Baselines

5.1.1 LR

LR is an l_1-regularized logistic regression model with k-d tree regions (Roller et al., 2012) used

[4]TwitterAPI returns the current version of metadata even for an old tweet.

in Rahimi et al. (2015a). The model uses tf-idf weighted bag-of-words unigrams for features. This model is simple, but it has shown state-of-the-art performance in cases when only text is available.

5.1.2 MADCEL-B-LR

MADCEL-B-LR, a model presented by (Rahimi et al., 2015a), combines LR with Modified Adsorption (MAD) (Talukdar and Crammer, 2009). MAD is a graph-based label propagation algorithm that optimizes an objective with a prior term, a smoothness term, and an uninformativeness term. LR is combined with MAD by introducing LR results as dongle nodes to MAD.

This model includes an algorithm for the construction of a mention network. The algorithm removes *celebrity users*[5] and *collapses a mention network*[6]. We use binary edges for user network edges because they performed slightly better than weighted edges by accuracy@161 metric in Rahimi et al. (2015a).

5.1.3 LR-STACK

LR-STACK is an ensemble learning model that combines four LR classifiers (LR-MSG, LR-LOC, LR-DESC, LR-TZ) with an l_2-regularized logistic regression meta-classifier (LR-2ND). LR-MSG, LR-LOC, LR-DESC, and LR-TZ respectively use messages, location fields, description fields, and timezones as their inputs. This model is similar to the stacking (Wolpert, 1992) approach taken in Han et al. (2013) and Han et al. (2014), which showed superior performance compared to a feature concatenation approach.

The model takes the following three steps to combine text and metadata: **Step 1** LR-MSG, LR-LOC, LR-DESC, and LR-TZ are trained using a training set, **Step 2** the outputs of the four classifiers on the training set are obtained with 10-fold cross validation, and **Step 3** LR-2ND is trained using the outputs of the four classifiers.

5.1.4 MADCEL-B-LR-STACK

MADCEL-B-LR-STACK is a combined model of MADCEL-B-LR and LR-STACK. LR-STACK results are introduced as dongle nodes to MAD instead of LR results to combine text, metadata, and network information.

[5]Users with more than t unique mentions.

[6]Users not included in training users or test users are removed and disconnected edges with the removals are converted to direct edges.

5.2 Model Configurations

5.2.1 Text Processor

We applied a lower case conversion, a unicode normalization, a Twitter user name normalization, and a URL normalization for text pre-processing. The pre-processed text is then segmented using Twokenizer (Owoputi et al., 2013) to obtain words.

5.2.2 Pre-training of Embeddings

We pre-trained word embeddings using messages, location fields, and description fields of a training set using fastText (Bojanowski et al., 2016) with the skip-gram algorithm. We also pre-trained user embeddings using the non-reduced mention network described in Section 4.2 of a training set with LINE (Tang et al., 2015). The detail of pre-training parameters are described in Appendix A.1.

5.2.3 Neural Network Optimization

We chose an objective function of our models to cross-entropy loss. l_2 regularization was applied to the RNN layers, the attention context vectors, and the FC layers of our models to avoid overfitting. The objective function was minimized through stochastic gradient descent over shuffled mini-batches with Adam (Kingma and Ba, 2014).

5.2.4 Model Parameters

The layers and the embeddings in our models have unit size and embedding dimension parameters. Our models and the baseline models have regularization parameter α, which is sensitive to a dataset. The baseline models have additional k-d tree bucket size c, celebrity threshold t, and MAD parameters μ_1, μ_2, and μ_3, which are also data sensitive.

We chose optimal values for these parameters in terms of accuracy with a grid search using the development sets of TwitterUS and W-NUT. Details of the parameter selection strategies and the selected values are described in Appendix A.2.

5.2.5 Metrics

We evaluate the models in the following four commonly used metrics in geolocation prediction: **accuracy** the percentage of correctly predicted cities, **accuracy@161** a relaxed accuracy that takes prediction errors within 161 km as correct predictions, **median error distance** median value of error distances in predictions, and **mean error distance** mean value of error distances in predictions.

	Model	Sign. Test ID	Accuracy	Accuracy @161	Error Distance	
					Median	Mean
Baselines (reported)	Han et al. (2012)		26.0	45.0	260	814
	Wing and Baldridge (2014)		-	49.2	170.5	703.6
	LR (Rahimi et al. 2015b)		-	50	159	686
	LR-NA (Rahimi et al. 2016)		-	51	148	636
	MADCEL-B-LR (Rahimi et al. 2015a)		-	60	77	533
	MADCEL-W-LR (Rahimi et al. 2015a)		-	60	78	529
Baselines (implemented)	LR	i	42.0	52.7	121.1	666.6
	MADCEL-B-LR	ii	50.2	60.1	66.5	582.8
	LR-STACK	iii	50.8	64.1	42.3*	427.7
	MADCEL-B-LR-STACK	iv	55.7	67.7	45.1	412.7
Our Models	SUB-NN-TEXT	i	44.9**	55.6**	110.5	585.1**
	SUB-NN-UNET	ii	51.0	61.5*	65.0	481.5**
	SUB-NN-META	iii	54.6**	67.2**	46.8	356.3**
	Proposed Model	iv	58.5**	70.1**	41.9*	335.7**

Table 2: Performances of our models and the baseline models on TwitterUS. Significance tests were performed between models with same Sign. Test IDs. The shaded lines represent values copied from related papers. Asterisks denote significant improvements against paired counterparts with 1% confidence (**) and 5% confidence (*).

	Model	Sign. Test ID	Accuracy	Accuracy @161	Error Distance	
					Median	Mean
Baselines (reported)	Miura et al. (2016)		47.6	-	16.1	1122.3
	Jayasinghe et al. (2016)		52.6	-	21.7	1928.8
Baselines (implemented)	LR	i	34.1	46.7	248.7	2216.4
	MADCEL-B-LR	ii	36.2	49.7	166.3	2120.6
	LR-STACK	iii	51.2	64.9	0.0	1496.4
	MADCEL-B-LR-STACK	iv	51.6	65.3	0.0	1471.9
Our Models	SUB-NN-TEXT	i	35.4**	50.3**	155.8**	1592.6**
	SUB-NN-UNET	ii	38.1**	53.3**	99.9**	1498.6**
	SUB-NN-META	iii	54.7**	70.2**	0.0	825.8**
	Proposed Model	iv	56.4**	71.9**	0.0	780.5**

Table 3: Performance of our models and baseline models on W-NUT. The same notations as those in Table 2 are used in this table.

5.3 Result

Performance on TwitterUS

Table 2 presents results of our models and the implemented baseline models on TwitterUS. We also list values from earlier reports (Han et al., 2012; Wing and Baldridge, 2014; Rahimi et al., 2015a,b, 2016) to make our results readily comparable with past reported values.

We performed some statistical significance tests among model pairs that share the same inputs. The values in the Sign. Test ID column of Table 2 represent the IDs of these pairs. As a preparation of statistical significance tests, accuracies, accuracy@161s, and error distances of each test user were calculated for each model pair. Two-sided Fisher-Pittman Permutation tests were used for testing accuracy and accuracy@161. Mood's median test was used for testing error distance in terms of median. Paired t-tests were used for testing error distance in terms of mean.

We confirmed the significance of improvements in accuracy@161 and mean distance error for all of our models. Three of our models also improved in terms of accuracy. Especially, the proposed model achieved a 2.8% increase in accuracy and a 2.4% increase in accuracy@161 against the counterpart baseline model MADCEL-B-LR-STACK. One negative result we found was the median error distance between SUB-NN-META and LR-STACK. The baseline model LR-STACK performed 4.5 km significantly better than our model.

Performance on W-NUT

Table 3 presents the results of our models and the implemented baseline models on W-NUT. As for TwitterUS, we listed values from Miura et al. (2016) and Jayasinghe et al. (2016). We tested the significance of these results in the same way as we did for TwitterUS.

We confirmed significant improvement in the four metrics for all of our models. The proposed model achieved a 4.8% increase in accuracy and a

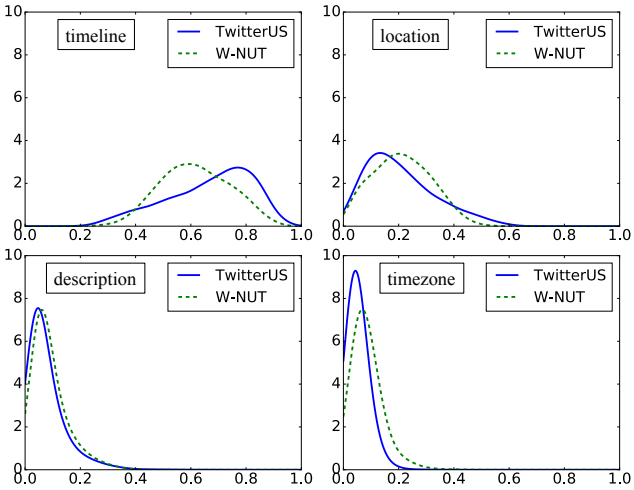


Figure 4: Estimated probability density functions of the four representations in $\text{Attention}_{\text{U}}$.

6.6% increase in accuracy@161 against the counterpart baseline model MADCEL-B-LR-STACK. The accuracy is 3.8% higher against the previously reported best value (Jayasinghe et al., 2016) which combined texts, metadata, and user network information with an ensemble method.

6 Discussion

6.1 Analyses of Attention Probabilities

6.1.1 Unification Strategies

In the evaluation, the proposed model has implicitly shown effectiveness at unifying text, metadata, and user network representations through improvements in the four metrics. However, details of the unification processes are not clear from the model outputs because they are merely the probabilities of estimated locations. To gain insight into the unification processes, we analyzed the states of two attention layers: $\text{Attention}_{\text{U}}$ and $\text{Attention}_{\text{UN}}$ in Figure 1.

Figure 4 presents the estimated probability density functions (PDFs) of the four input representations for $\text{Attention}_{\text{U}}$. These PDFs are estimated with kernel density estimation from the development sets of TwitterUS and W-NUT, where all four representations are available. From the PDFs, it is apparent that the model assigns higher probabilities to time line representations than to other three representations in TwitterUS compared to W-NUT. This finding is reasonable because timelines in TwitterUS consist of more tweets (tweet/user in Table 1) and are likely to be more informative than in W-NUT.

Figure 5 presents the estimated PDFs of user network representations for $\text{Attention}_{\text{UN}}$. These

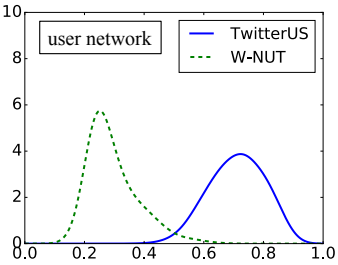


Figure 5: Estimated probability density functions of user network representations in $\text{Attention}_{\text{UN}}$.

PDFs are estimated from the development sets of TwitterUS and W-NUT, where both input representations are available. Strong preference of network representation for TwitterUS against W-NUT is found in the PDFs. This finding is intuitive because TwitterUS has substantially more user network edges (reduced-edge/user in Table 1) than W-NUT, which is likely to benefit more from user network information.

6.1.2 Attention Patterns

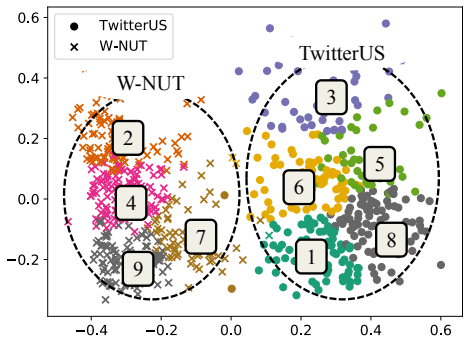
We further analyzed the proposed model by clustering attention probabilities to capture typical attention patterns. For each user, we assigned six attention probabilities of $\text{Attention}_{\text{U}}$ and $\text{Attention}_{\text{UN}}$ as features for a clustering. A k-means clustering was performed over these users with 9 clusters. The clustering clearly separated the users to 5 clusters for TwitterUS users and 4 clusters for W-NUT users. We extracted typical users of each cluster by selecting the closest users of the cluster centroids. Figure 6 shows a clustering result and the attention probabilities of these users.

These attention probabilities can be considered as typical attention patterns of the proposed model and match with the previously estimated PDFs. For example, cluster 2 and 3 represent an attention pattern that processes users by balancing the representations of locations along with the representations of timelines. Additionally, the location probabilities in this pattern are in the right tail region of the location PDF.

6.2 Limitations of Proposed Model

6.2.1 City Prediction

The evaluation produced improvements in most of our models in the four metrics. One exception we found was the median distance error between SUB-NN-META and LR-STACKING in TwitterUS. Because the median distance error of SUB-NN-META was quite low (46.8 km), we



Cluster ID	Dataset	Timeline	Location	Description	Timezone	User	User Network
1	TwitterUS	0.843	0.082	0.040	0.035	0.359	0.641
2	W-NUT	0.517	0.317	0.081	0.085	0.732	0.268
3	TwitterUS	0.432	0.430	0.069	0.069	0.319	0.681
4	W-NUT	0.637	0.160	0.097	0.105	0.737	0.263
5	TwitterUS	0.593	0.219	0.114	0.075	0.230	0.770
6	TwitterUS	0.672	0.214	0.069	0.045	0.365	0.635
7	W-NUT	0.741	0.077	0.080	0.102	0.605	0.395
8	TwitterUS	0.766	0.099	0.068	0.067	0.222	0.778
9	W-NUT	0.800	0.067	0.056	0.078	0.730	0.270

Figure 6: A k-means clustering result and the attention probabilities of users that are closest to the cluster centroids. The underlined values are the max values of the two datasets for each column.

Model	Error Distance		
	Median	Mean	σ
Oracle	23.3	31.4	30.1

Table 4: Error distance values in TwitterUS with oracle predictions. σ in the table denotes the standard deviation.

measured the performance of an oracle model where city predictions are all correct (accuracy of 100%) in the test set.

Table 4 denotes this oracle performance. The oracle mean error distance is 31.4 km. Its standard deviation is 30.1. Note that ground truth locations of TwitterUS are geotags and will not exactly match the oracle city centers. These oracle values imply that the current median error distances are close to the lower bound of the city classification approach and that they are difficult to improve.

6.2.2 Errors with High Confidences

The proposed model still contains 28–30% errors even in accuracy@161. A qualitative analysis of errors with high confidences was performed to investigate cases that the model fails. We found two common types of error in the error analysis. The first is a case when a location field is incorrect due to a reason such as a house move. For example, the model predicted "Hong Kong" for a user with a location field of "Hong Kong" but has the gold location of "Toronto". The second is a case when a user tweets a place name of a travel. For example, the model predicted "San Francisco" for a user who tweeted about a travel to "San Francisco" but has the gold location of "Boston".

These two types of error are difficult to handle with the current architecture of the proposed model. The architecture only supports single location field which disables the model to track location changes. The architecture also treats each tweet independently which forbids the model to express a temporal state like traveling.

7 Conclusion

As described in this paper, we proposed a complex neural network model for geolocation prediction. The model unifies text, metadata, and user network information. The model achieved the maximum of a 3.8% increase in accuracy and a maximum of 6.6% increase in accuracy@161 against several previous state-of-the-art models. We further analyzed the states of several attention layers, which revealed that the probabilities assigned to timeline representations and user network representations match to some statistical characteristics of datasets.

As future works of this study, we are planning to expand the proposed model to handle multiple locations and a temporal state to capture location changes and states like traveling. Additionally, we plan to apply the proposed model to other social media analyses such as gender analysis and age analysis. In these analyses, metadata like location fields and timezones may not be effective like in geolocation prediction. However, a user network is known to include various user attributes information including gender and age (McPherson et al., 2001) which suggests the unification of text and user network information to result in a success as in geolocation prediction.

Acknowledgments

We would like to thank the members of Okumura–Takamura Group at Tokyo Institute of Technology for having insightful discussions about user profiling models in social media. We would also like to thank the anonymous reviewer for their comments to improve this paper.

References

Amr Ahmed, Liangjie Hong, and Alexander J. Smola. 2013. Hierarchical geographical modeling of user locations from social media posts. In *Proceedings of the 22nd International Conference on World Wide Web*. pages 25–36.

Lars Backstrom, Eric Sun, and Cameron Marlow. 2010. Find me if you can: Improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th International Conference on World Wide Web*. pages 61–70.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computing Research Repository* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Miriam Cha, Youngjune Gwon, and H. T. Kung. 2015. Twitter geolocation and regional classification via sparse coding. In *Proceedings of the Ninth International AAAI Conference on Web and Social Media*.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: A content-based approach to geo-locating Twitter users. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. pages 759–768.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2013. A content-driven framework for geolocating microblog users. *ACM Transactions on Intelligent Systems and Technology* 4(1):1–27. Article 2.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 1724–1734.

Ryan Compton, David Jurgens, and David Allen. 2014. Geotagging one hundred million Twitter accounts with total variation minimization. In *Proceedings of the 2014 IEEE International Conference on Big-Data*. pages 393–401.

David J. Crandall, Lars Backstrom, Daniel Huttenlocher, and Jon Kleinberg. 2009. Mapping the world's photos. In *Proceedings of the 18th International Conference on World Wide Web*. pages 761–770.

Aron Culotta. 2010. Towards detecting influenza epidemics by analyzing Twitter messages. In *Proceedings of the First Workshop on Social Media Analytics*. pages 115–122.

Clodoveu A. Davis Jr., Gisele L. Pappa, Diogo Rennó Rocha de Oliveira, and Filipe de L. Arcanjo. 2011. Inferring the location of Twitter messages based on user relationships. *Transactions in GIS* 15(6):735–751.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proceedings of the 28th International Conference on Machine Learning*. pages 1041–1048.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. pages 1277–1287.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer-Verlag Berlin Heidelberg.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Geolocation prediction in social media data by finding location indicative words. In *Proceedings of COLING 2012*. pages 1045–1062.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. A stacking-based approach to twitter user geolocation prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. pages 7–12.

Bo Han, Paul Cook, and Timothy Baldwin. 2014. Text-based Twitter user geolocation prediction. *Journal of Artificial Intelligence Research* 49(1):451–500.

Bo Han, Afshin Rahimi, Leon Derczynski, and Timothy Baldwin. 2016. Twitter geolocation prediction shared task of the 2016 workshop on noisy user-generated text. In *Proceedings of the Second Workshop on Noisy User-generated Text*. pages 213–217.

Brent Hecht, Lichan Hong, Bongwon Suh, and Ed H. Chi. 2011. Tweets from Justin Bieber's heart: the dynamics of the location field in user profiles. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. pages 237–246.

Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the Twitter stream. In *Proceedings of the 21st International Conference on World Wide Web*. pages 769–778.

Gaya Jayasinghe, Brian Jin, James Mchugh, Bella Robinson, and Stephen Wan. 2016. CSIRO Data61 at the WNUT geo shared task. In *Proceedings of the Second Workshop on Noisy User-generated Text*. pages 218–226.

David Jurgens. 2013. That's what friends are for: Inferring location in online social media platforms based on social relationships. In *Proceedings of the Seventh International AAAI Conference on Web and Social Media*.

David Jurgens, Tyler Finethy, James McCorriston, Yi Xu, and Derek Ruths. 2015. Geolocation prediction in Twitter using social networks: A critical analysis and review of current practice. In *Proceedings of the Ninth International AAAI Conference on Web and Social Media*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Sheila Kinsella, Vanessa Murdock, and Neil O'Hare. 2011. "I'm eating a sandwich in Glasgow": Modeling locations with tweets. In *Proceedings of the Third International Workshop on Search and Mining User-generated Contents*. pages 61–68.

Longbo Kong, Zhi Liu, and Yan Huang. 2014. SPOT: Locating social media users based on social network context. *Proceedings of the VLDB Endowment* 7(13):1681–1684.

Rui Li, Shengjie Wang, and Kevin Chen-Chuan Chang. 2012a. Multiple location profiling for users and relationships from social network and content. *Proceedings of the VLDB Endowment* 5(11):1603–1614.

Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, and Kevin Chen-Chuan Chang. 2012b. Towards social user profiling: Unified and discriminative influence model for inferring home locations. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 1023–1031.

Ji Liu and Diana Inkpen. 2015. Estimating user location in social media with stacked denoising autoencoders. In *Proceedings of the First Workshop on Vector Space Modeling for Natural Language Processing*. pages 201–210.

Jalal Mahmud, Jeffrey Nichols, and Clemens Drews. 2012. Where is this tweet from? Inferring home locations of Twitter users. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media*.

Eugenio Martínez-Cámara, Maria Teresa Martín-Valdivia, Luis Alfonso Ureña López, and Arturo Montejo Raéz. 2014. Sentiment analysis in Twitter. *Natural Language Engineering* 20(1):1–28.

Jeffrey McGee, James Caverlee, and Zhiyuan Cheng. 2013. Location prediction in social media based on tie strength. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. pages 459–468.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27(1):415–444.

Yasuhide Miura, Motoki Taniguchi, Tomoki Taniguchi, and Tomoko Ohkuma. 2016. A simple scalable neural networks based model for geolocation prediction in Twitter. In *Proceedings of the Second Workshop on Noisy User-generated Text*. pages 235–239.

Simon E. Overell. 2009. *Geographic Information Retrieval: Classification, Disambiguation, and Modelling*. Ph.D. thesis, Imperial College London.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 380–390.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2015a. Twitter user geolocation using a unified text and network prediction model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. pages 630–636.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2016. pigeo: A python geotagging tool. In *Proceedings of ACL-2016 System Demonstrations*. pages 127–132.

Afshin Rahimi, Duy Vu, Trevor Cohn, and Timothy Baldwin. 2015b. Exploiting text and network context for geolocation of social media users. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1362–1367.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in Twitter. In *Proceedings of the Second International Workshop on Search and Mining User-generated Contents*. pages 37–44.

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 1500–1510.

Dominic Rout, Kalina Bontcheva, Daniel Preoţiuc-Pietro, and Trevor Cohn. 2013. Where's @wally?: A classification approach to geolocating users based on their social ties. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*. pages 11–20.

Adam Sadilek, Henry Kautz, and Jeffrey P. Bigham. 2012. Finding your friends and following them to where you are. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*. pages 723–732.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*. pages 851–860.

Axel Schulz, Aristotelis Hadjakos, Heiko Paulheim, Johannes Nachtwey, and Max Mühlhäuser. 2013. A multi-indicator approach for geolocalization of tweets. In *Proceedings of the Seventh International AAAI Conference on Web and Social Media*.

Pavel Serdyukov, Vanessa Murdock, and Roelof van Zwol. 2009. Placing Flickr photos on a map. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 484–491.

Partha Pratim Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*. pages 442–457.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. pages 1067–1077.

Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner, and Isabell M. Welpe. 2010. Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*. pages 178–185.

Benjamin Wing and Jason Baldridge. 2011. Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 955–964.

Benjamin Wing and Jason Baldridge. 2014. Hierarchical discriminative classification for text-based geolocation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 336–348.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks* 5(2):241–259.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1480–1489.

A Supplemental Materials

A.1 Parameters of Embedding Pre-training

Word embeddings were pre-trained with the parameters of learning rate=0.025, window size=5, negative sample size=5, and epoch=5. User embeddings were pre-trained with the parameters of initial learning rate=0.025, order=2, negative sample size=5, and training sample size=100M.

A.2 Model Parameters and Parameter Selection Strategies

Unit Sizes, Embedding Dimensions, and a Max Tweet Number

The layers and the embeddings in our models have unit size and embedding dimension parameters. We also restricted the maximum number of tweets per user for TwitterUS to reduce memory footprints. Table 5 shows the values for these parameters. Smaller values were set for TwitterUS because TwitterUS is approximately 2.6 times larger in terms of tweet number. It was computationally expensive to process TwiiterUS in the same settings as W-NUT.

Regularization Parameters and Bucket Sizes

We chose optimal values of α using a grid search with the development sets of TwitterUS and W-NUT. The range of α was set as the following:
$\alpha \in \{1e^{-4}, 5e^{-5}, 1e^{-5}, 5e^{-6}, 1e^{-6}, 5e^{-7}, 1e^{-7}, 5e^{-8}, 1e^{-8}\}$.

We also chose optimal values of c using grid search with the development sets of TwitterUS and W-NUT for the baseline models. The range of c was set as the following for TwitterUS:
$c \in \{50, 100, 150, 200, 250, 300, 339\}$.
The following was set for W-NUT:
$c \in \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000, 2500, 3000, 3028\}$.
Table 6 presents selected values of α and c. For LR-STACK and MADCEl-B-LR-STACK, different parameters of α and c were selected for each logistic regression classifier.

MAD Parameters and Celebrity Threshold

The MAD parameters μ_1, μ_2, and μ_3 and celebrity threshold t were also chosen using grid search with the development sets of TwitterUS and W-NUT. The ranges of μ_1, μ_2, and μ_3 were set as the following:
$\mu_1 \in \{1.0\}$, $\mu_2 \in \{0.001, 0.01, 0.1, 1.0, 10.0\}$,
$\mu_3 \in \{0.0, 0.001, 0.01, 0.1, 1.0, 10.0\}$.
The range of t for TwitterUS was set as $t \in \{2, \ldots, 16\}$. The range of t for W-NUT was set as $t \in \{2, \ldots, 6\}$. Table 6 presents selected values of μ_1, μ_2, μ_3, and t.

	TwitterUS	W-NUT
RNN unit size	100	200
Attention context vector size	200	400
FC unit size	200	400
Word embedding dimension	100	200
Timezone embedding dimension	200	400
City embedding dimension	200	400
User embedding dimension	200	400
Max tweet number per user	200	-

Table 5: Unit sizes, embedding dimensions, and max tweet numbers of our models.

Model	Parameter	TwitterUS	W-NUT
SUB-NN-TEXT	α	1e-8	1e-7
SUB-NN-UNET		1e-6	5e-8
SUB-NN-META		1e-8	5e-8
Proposed Model		1e-6	5e-8
LR MADCEL-B-LR	α	1e-6	5e-7
	c	300	3000
LR-STACK MADCEL-B-LR-STACK	α_{MSG}	1e-6	5e-7
	α_{LOC}	1e-6	1e-6
	α_{DESC}	5e-6	1e-6
	α_{TZ}	1e-4	5e-6
	α_{2ND}	1e-6	1e-7
	c_{MSG}	300	3000
	c_{LOC}	300	3000
	c_{DESC}	250	1500
	c_{TZ}	100	2500
	c_{2ND}	300	2000

Table 6: Regularization parameters and bucket sizes selected for our models and baseline models.

Model	Parameter	TwitterUS	W-NUT
MADCEL-B-LR	μ_1	1.0	1.0
	μ_2	1.0	10.0
	μ_3	0.01	0.1
	t	5	4
MADCEL-B-LR-STACK	μ_1	1.0	1.0
	μ_2	1.0	1.0
	μ_3	0.1	0.0
	t	4	2

Table 7: MAD parameters and celebrity threshold selected for baseline models.

Multi-Task Video Captioning with Video and Entailment Generation

Ramakanth Pasunuru and **Mohit Bansal**
UNC Chapel Hill
{ram, mbansal}@cs.unc.edu

Abstract

Video captioning, the task of describing the content of a video, has seen some promising improvements in recent years with sequence-to-sequence models, but accurately learning the temporal and logical dynamics involved in the task still remains a challenge, especially given the lack of sufficient annotated data. We improve video captioning by sharing knowledge with two related directed-generation tasks: a temporally-directed unsupervised video prediction task to learn richer context-aware video encoder representations, and a logically-directed language entailment generation task to learn better video-entailing caption decoder representations. For this, we present a many-to-many multi-task learning model that shares parameters across the encoders and decoders of the three tasks. We achieve significant improvements and the new state-of-the-art on several standard video captioning datasets using diverse automatic and human evaluations. We also show mutual multi-task improvements on the entailment generation task.

1 Introduction

Video captioning is the task of automatically generating a natural language description of the content of a video, as shown in Fig. 1. It has various applications such as assistance to a visually impaired person and improving the quality of online video search or retrieval. This task has gained recent momentum in the natural language processing and computer vision communities, esp. with the advent of powerful image processing features as well as sequence-to-sequence LSTM models. It is also a step forward from static image captioning, because in addition to modeling the spatial visual features, the model also needs to learn the temporal across-frame action dynamics and the logical storyline language dynamics.

Ground truth: A person is mixing powdered ingradients with water.
A woman is mixing flour and water in a bowl.
Our model: A person is mixing ingredients in a bowl.

Figure 1: A video captioning example from the YouTube2Text dataset, with the ground truth captions and our many-to-many multi-task model's predicted caption.

Previous work in video captioning (Venugopalan et al., 2015a; Pan et al., 2016b) has shown that recurrent neural networks (RNNs) are a good choice for modeling the temporal information in the video. A sequence-to-sequence model is then used to 'translate' the video to a caption. Venugopalan et al. (2016) showed linguistic improvements over this by fusing the decoder with external language models. Furthermore, an attention mechanism between the video frames and the caption words captures some of the temporal matching relations better (Yao et al., 2015; Pan et al., 2016a). More recently, hierarchical two-level RNNs were proposed to allow for longer inputs and to model the full paragraph caption dynamics of long video clips (Pan et al., 2016a; Yu et al., 2016).

Despite these recent improvements, video captioning models still suffer from the lack of sufficient temporal and logical supervision to be able to correctly capture the action sequence and story-dynamic language in videos, esp. in the case of short clips. Hence, they would benefit from incorporating such complementary directed knowledge, both visual and textual. We address this by jointly training the task of video captioning with two related directed-generation tasks: a temporally-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1273–1283
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1117

directed unsupervised video prediction task and a logically-directed language entailment generation task. We model this via many-to-many multi-task learning based sequence-to-sequence models (Luong et al., 2016) that allow the sharing of parameters among the encoders and decoders across the three different tasks, with additional shareable attention mechanisms.

The unsupervised video prediction task, i.e., video-to-video generation (adapted from Srivastava et al. (2015)), shares its encoder with the video captioning task's encoder, and helps it learn richer video representations that can predict their temporal context and action sequence. The entailment generation task, i.e., premise-to-entailment generation (based on the image caption domain SNLI corpus (Bowman et al., 2015)), shares its decoder with the video captioning decoder, and helps it learn better video-entailing caption representations, since the caption is essentially an entailment of the video, i.e., it describes subsets of objects and events that are logically implied by or follow from the full video content). The overall many-to-many multi-task model combines all three tasks.

Our three novel multi-task models show statistically significant improvements over the state-of-the-art, and achieve the best-reported results (and rank) on multiple datasets, based on several automatic and human evaluations. We also demonstrate that video captioning, in turn, gives mutual improvements on the new multi-reference entailment generation task.

2 Related Work

Early video captioning work (Guadarrama et al., 2013; Thomason et al., 2014; Huang et al., 2013) used a two-stage pipeline to first extract a subject, verb, and object (S,V,O) triple and then generate a sentence based on it. Venugopalan et al. (2015b) fed mean-pooled static frame-level visual features (from convolution neural networks pre-trained on image recognition) of the video as input to the language decoder. To harness the important frame sequence temporal ordering, Venugopalan et al. (2015a) proposed a sequence-to-sequence model with video encoder and language decoder RNNs.

More recently, Venugopalan et al. (2016) explored linguistic improvements to the caption decoder by fusing it with external language models. Moreover, an attention or alignment mechanism was added between the encoder and the decoder to learn the temporal relations (matching) between the video frames and the caption words (Yao et al., 2015; Pan et al., 2016a). In contrast to static visual features, Yao et al. (2015) also considered temporal video features from a 3D-CNN model pre-trained on an action recognition task.

To explore long range temporal relations, Pan et al. (2016a) proposed a two-level hierarchical RNN encoder which limits the length of input information and allows temporal transitions between segments. Yu et al. (2016)'s hierarchical RNN generates sentences at the first level and the second level captures inter-sentence dependencies in a paragraph. Pan et al. (2016b) proposed to simultaneously learn the RNN word probabilities and a visual-semantic joint embedding space that enforces the relationship between the semantics of the entire sentence and the visual content. Despite these useful recent improvements, video captioning still suffers from limited supervision and generalization capabilities, esp. given the complex action-based temporal and story-based logical dynamics that need to be captured from short video clips. Our work addresses this issue by bringing in complementary temporal and logical knowledge from video prediction and textual entailment generation tasks (respectively), and training them together via many-to-many multi-task learning.

Multi-task learning is a useful learning paradigm to improve the supervision and the generalization performance of a task by jointly training it with related tasks (Caruana, 1998; Argyriou et al., 2007; Kumar and Daumé III, 2012). Recently, Luong et al. (2016) combined multi-task learning with sequence-to-sequence models, sharing parameters across the tasks' encoders and decoders. They showed improvements on machine translation using parsing and image captioning. We additionally incorporate an attention mechanism to this many-to-many multi-task learning approach and improve the multimodal, temporal-logical video captioning task by sharing its video encoder with the encoder of a video-to-video prediction task and by sharing its caption decoder with the decoder of a linguistic premise-to-entailment generation task.

Image representation learning has been successful via supervision from very large object-labeled datasets. However, similar amounts of supervision are lacking for video representation learning. Srivastava et al. (2015) address this by propos-

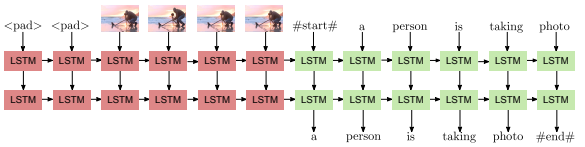


Figure 2: Baseline sequence-to-sequence model for video captioning: standard encoder-decoder LSTM-RNN model.

ing unsupervised video representation learning via sequence-to-sequence RNN models, where they reconstruct the input video sequence or predict the future sequence. We model video generation with an attention-enhanced encoder-decoder and harness it to improve video captioning.

The task of recognizing textual entailment (RTE) is to classify whether the relationship between a premise and hypothesis sentence is that of entailment (i.e., logically follows), contradiction, or independence (neutral), which is helpful for several downstream NLP tasks. The recent Stanford Natural Language Inference (SNLI) corpus by Bowman et al. (2015) allowed training end-to-end neural networks that outperform earlier feature-based RTE models (Lai and Hockenmaier, 2014; Jimenez et al., 2014). However, directly generating the entailed hypothesis sentences given a premise sentence would be even more beneficial than retrieving or reranking sentence pairs, because most downstream generation tasks only come with the source sentence and not pairs. Recently, Kolesnyk et al. (2016) tried a sequence-to-sequence model for this on the original SNLI dataset, which is a single-reference setting and hence restricts automatic evaluation. We modify the SNLI corpus to a new multi-reference (and a more challenging zero train-test premise overlap) setting, and present a novel multi-task training setup with the related video captioning task (where the caption also entails a video), showing mutual improvements on both the tasks.

3 Models

We first discuss a simple encoder-decoder model as a baseline reference for video captioning. Next, we improve this via an attention mechanism. Finally, we present similar models for the unsupervised video prediction and entailment generation tasks, and then combine them with video captioning via the many-to-many multi-task approach.

3.1 Baseline Sequence-to-Sequence Model

Our baseline model is similar to the standard machine translation encoder-decoder RNN model (Sutskever et al., 2014) where the final state of the encoder RNN is input as an initial state to the decoder RNN, as shown in Fig. 2. The RNN is based on Long Short Term Memory (LSTM) units, which are good at memorizing long sequences due to forget-style gates (Hochreiter and Schmidhuber, 1997). For video captioning, our input to the encoder is the video frame features[1] $\{f_1, f_2, ..., f_n\}$ of length n, and the caption word sequence $\{w_1, w_2, ..., w_m\}$ of length m is generated during the decoding phase. The distribution of the output sequence w.r.t. the input sequence is:

$$p(w_1, ..., w_m | f_1, ..., f_n) = \prod_{t=1}^{m} p(w_t | h_t^d) \quad (1)$$

where h_t^d is the hidden state at the t^{th} time step of the decoder RNN, obtained from h_{t-1}^d and w_{t-1} via the standard LSTM-RNN equations. The distribution $p(w_t|h_t^d)$ is given by *softmax* over all the words in the vocabulary.

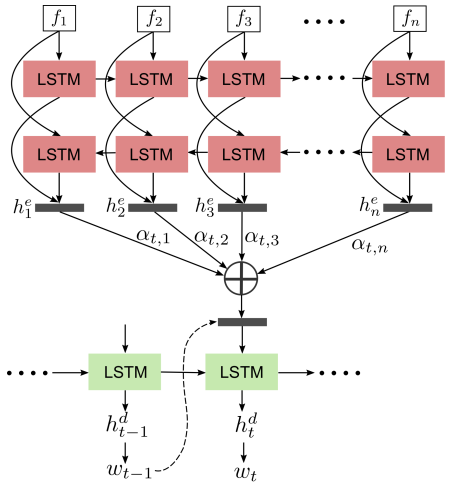


Figure 3: Attention-based sequence-to-sequence baseline model for video captioning (similar models also used for video prediction and entailment generation).

3.2 Attention-based Model

Our attention model architecture is similar to Bahdanau et al. (2015), with a bidirectional LSTM-RNN as the encoder and a unidirectional LSTM-RNN as the decoder, see Fig. 3. At each time step t, the decoder LSTM hidden state h_t^d is a non-linear recurrent function of the previous decoder hidden state h_{t-1}^d, the previous time-step's generated word w_{t-1}, and the context vector c_t:

$$h_t^d = S(h_{t-1}^d, w_{t-1}, c_t) \quad (2)$$

[1] We use several popular image features such as VGGNet, GoogLeNet and Inception-v4. Details in Sec. 4.1.

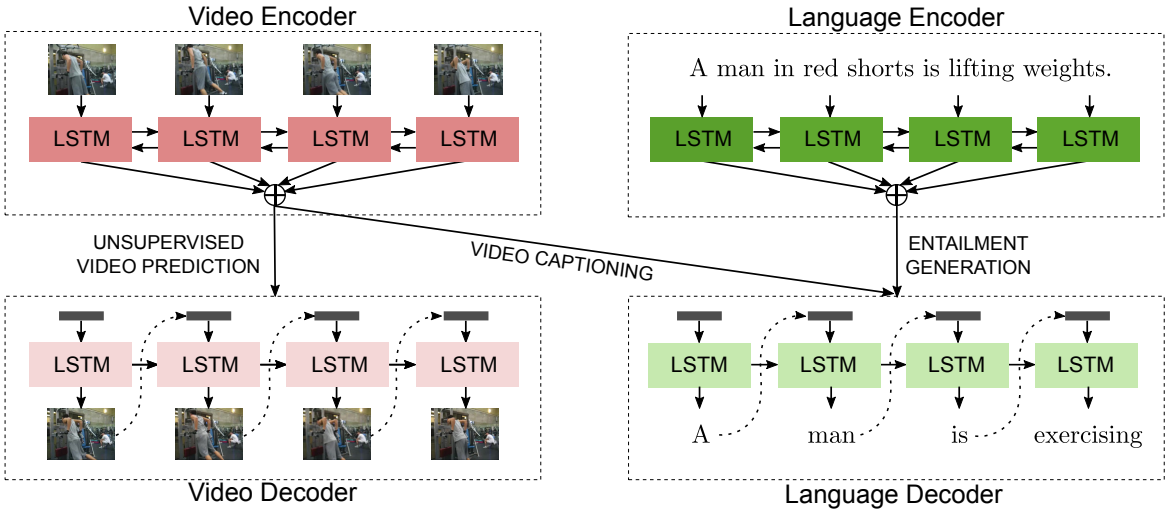


Figure 4: Our many-to-many multi-task learning model to share encoders and decoders of the video captioning, unsupervised video prediction, and entailment generation tasks.

where c_t is a weighted sum of encoder hidden states $\{h_i^e\}$:

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i^e \tag{3}$$

These attention weights $\{\alpha_{t,i}\}$ act as an alignment mechanism by giving higher weights to certain encoder hidden states which match that decoder time step better, and are computed as:

$$\alpha_{t,i} = \frac{exp(e_{t,i})}{\sum_{k=1}^{n} exp(e_{t,k})} \tag{4}$$

where the attention function $e_{t,i}$ is defined as:

$$e_{t,i} = w^T tanh(W_a^e h_i^e + W_a^d h_{t-1}^d + b_a) \tag{5}$$

where w, W_a^e, W_a^d, and b_a are learned parameters. This attention-based sequence-to-sequence model (Fig. 3) is our enhanced baseline for video captioning. We next discuss similar models for the new tasks of unsupervised video prediction and entailment generation and then finally share them via multi-task learning.

3.3 Unsupervised Video Prediction

We model unsupervised video representation by predicting the sequence of future video frames given the current frame sequence. Similar to Sec. 3.2, a bidirectional LSTM-RNN encoder and an LSTM-RNN decoder is used, along with attention. If the frame level features of a video of length n are $\{f_1, f_2, ..., f_n\}$, these are divided into two sets such that given the current frames $\{f_1, f_2, .., f_k\}$ (in its encoder), the model has to predict (decode) the rest of the frames $\{f_{k+1}, f_{k+2}, .., f_n\}$. The motivation is that this helps the video encoder learn rich temporal representations that are aware of their action-based context and are also robust to missing frames and varying frame lengths or motion speeds. The optimization function is defined as:

$$\underset{\phi}{\text{minimize}} \sum_{t=1}^{n-k} ||f_t^d - f_{t+k}||_2^2 \tag{6}$$

where ϕ are the model parameters, f_{t+k} is the true future frame feature at decoder time step t and f_t^d is the decoder's predicted future frame feature at decoder time step t, defined as:

$$f_t^d = S(h_{t-1}^d, f_{t-1}^d, c_t) \tag{7}$$

similar to Eqn. 2, with h_{t-1}^d and f_{t-1}^d as the previous time step's hidden state and predicted frame feature respectively, and c_t as the attention-weighted context vector.

3.4 Entailment Generation

Given a sentence (premise), the task of entailment generation is to generate a sentence (hypothesis) which is a logical deduction or implication of the premise. Our entailment generation model again uses a bidirectional LSTM-RNN encoder and LSTM-RNN decoder with an attention mechanism (similar to Sec. 3.2). If the premise s^p is a sequence of words $\{w_1^p, w_2^p, ..., w_n^p\}$ and the hypothesis s^h is $\{w_1^h, w_2^h, ..., w_m^h\}$, the distribution of the entailed hypothesis w.r.t. the premise is:

$$p(w_1^h, ..., w_m^h | w_1^p, ..., w_n^p) = \prod_{t=1}^{m} p(w_t^h | h_t^d) \tag{8}$$

where the distribution $p(w_t^h | h_t^d)$ is again obtained via softmax over all the words in the vocabulary and the decoder state h_t^d is similar to Eqn. 2.

3.5 Multi-Task Learning

Multi-task learning helps in sharing information between different tasks and across domains. Our primary aim is to improve the video captioning model, where visual content translates to a textual form in a directed (entailed) generation way. Hence, this presents an interesting opportunity to share temporally and logically directed knowledge with both visual and linguistic generation tasks. Fig. 4 shows our overall many-to-many multi-task model for jointly learning video captioning, unsupervised video prediction, and textual entailment generation. Here, the video captioning task shares its video encoder (parameters) with the encoder of the video prediction task (one-to-many setting) so as to learn context-aware and temporally-directed visual representations (see Sec. 3.3).

Moreover, the decoder of the video captioning task is shared with the decoder of the textual entailment generation task (many-to-one setting), thus helping generate captions that can 'entail', i.e., are logically implied by or follow from the video content (see Sec. 3.4).[2] In both the one-to-many and the many-to-one settings, we also allow the attention parameters to be shared or separated. The overall many-to-many setting thus improves both the visual and language representations of the video captioning model.

We train the multi-task model by alternately optimizing each task in mini-batches based on a mixing ratio. Let α_v, α_f, and α_e be the number of mini-batches optimized alternately from each of these three tasks – video captioning, unsupervised video future frames prediction, and entailment generation, resp. Then the mixing ratio is defined as $\frac{\alpha_v}{(\alpha_v+\alpha_f+\alpha_e)} : \frac{\alpha_f}{(\alpha_v+\alpha_f+\alpha_e)} : \frac{\alpha_e}{(\alpha_v+\alpha_f+\alpha_e)}$.

4 Experimental Setup

4.1 Datasets

Video Captioning Datasets We report results on three popular video captioning datasets. First, we use the YouTube2Text or MSVD (Chen and Dolan, 2011) for our primary results, which contains 1970 YouTube videos in the wild with several different reference captions per video (40 on average). We also use MSR-VTT (Xu et al., 2016) with $10,000$ diverse video clips (from a video search engine) – it has $200,000$ video clip-sentence pairs and around 20 captions per video; and M-VAD (Torabi et al., 2015) with $49,000$ movie-based video clips but only 1 or 2 captions per video, making most evaluation metrics (except paraphrase-based METEOR) infeasible. We use the standard splits for all three datasets. Further details about all these datasets are provided in the supplementary.

Video Prediction Dataset For our unsupervised video representation learning task, we use the UCF-101 action videos dataset (Soomro et al., 2012), which contains $13,320$ video clips of 101 action categories, and suits our video captioning task well because it also contains short video clips of a single action or few actions. We use the standard splits – further details in supplementary.

Entailment Generation Dataset For the entailment generation encoder-decoder model, we use the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015), which contains human-annotated English sentence pairs with classification labels of entailment, contradiction and neutral. It has a total of $570,152$ sentence pairs out of which $190,113$ correspond to true entailment pairs, and we use this subset in our multi-task video captioning model. For improving video captioning, we use the same training/validation/test splits as provided by Bowman et al. (2015), which is $183,416$ training, $3,329$ validation, and $3,368$ testing pairs (for the entailment subset).

However, for the entailment generation multi-task results (see results in Sec. 5.3), we modify the splits so as to create a multi-reference setup which can afford evaluation with automatic metrics. A given premise usually has multiple entailed hypotheses but the original SNLI corpus is set up as single-reference (for classification). Due to this, the different entailed hypotheses of the same premise land up in different splits of the dataset (e.g., one in train and one in test/validation) in many cases. Therefore, we regroup the premise-entailment pairs and modify the split as follows: among the $190,113$ premise-entailment pairs subset of the SNLI corpus, there are $155,898$ unique premises; out of which $145,822$ have only one hy-

[2] Empirically, logical entailment helped captioning more than simple fusion with language modeling (i.e., partial sentence completion with no logical implication), because a caption also entails a video in a logically-directed sense and hence the entailment generation task matches the video captioning task better than language modeling. Moreover, a multi-task setup is more suitable to add directed information such as entailment (as opposed to pretraining or fusion with only the decoder). Details in Sec. 5.1.

pothesis and we make this the training set, and the rest of them $(10,076)$ have more than one hypothesis, which we randomly shuffle and divide equally into test and validation sets, so that each of these two sets has approximately the same distribution of the number of reference hypotheses per premise.

These new validation and test sets hence contain premises with multiple entailed hypotheses as ground truth references, thus allowing for automatic metric evaluation, where differing generations still get positive scores by matching one of the multiple references. Also, this creates a more challenging dataset for entailment generation because of zero premise overlap between the training and val/test sets. We will make these split details publicly available.

Pre-trained Visual Frame Features For the three video captioning and UCF-101 datasets, we fix our sampling rate to $3fps$ to bring uniformity in the temporal representation of actions across all videos. These sampled frames are then converted into features using several state-of-the-art pre-trained models on ImageNet (Deng et al., 2009) – VGGNet (Simonyan and Zisserman, 2015), GoogLeNet (Szegedy et al., 2015; Ioffe and Szegedy, 2015), and Inception-v4 (Szegedy et al., 2016). Details of these feature dimensions and layer positions are in the supplementary.

4.2 Evaluation (Automatic and Human)

For our video captioning as well as entailment generation results, we use four diverse automatic evaluation metrics that are popular for image/video captioning and language generation in general: METEOR (Denkowski and Lavie, 2014), BLEU-4 (Papineni et al., 2002), CIDEr-D (Vedantam et al., 2015), and ROUGE-L (Lin, 2004). Particularly, METEOR and CIDEr-D have been justified to be better for generation tasks, because CIDEr-D uses consensus among the (large) number of references and METEOR uses soft matching based on stemming, paraphrasing, and WordNet synonyms. We use the standard evaluation code from the Microsoft COCO server (Chen et al., 2015) to obtain these results and also to compare the results with previous papers.[3]

We also present human evaluation results based on relevance (i.e., how related is the generated caption w.r.t. the video contents such as actions, objects, and events; or is the generated hypothesis entailed or implied by the premise) and coherence (i.e., a score on the logic, readability, and fluency of the generated sentence).

4.3 Training Details

We tune all hyperparameters on the dev splits: LSTM-RNN hidden state size, learning rate, weight initializations, and mini-batch mixing ratios (tuning ranges in supplementary). We use the following settings in all of our models (unless otherwise specified): we unroll video encoder/decoder RNNs to 50 time steps and language encoder/decoder RNNs to 30 time steps. We use a 1024-dimension RNN hidden state size and 512-dim vectors to embed visual features and word vectors. We use Adam optimizer (Kingma and Ba, 2015). We apply a dropout of 0.5. See subsections below and supp for full details.

5 Results and Analysis

5.1 Video Captioning on YouTube2Text

Table 1 presents our primary results on the YouTube2Text (MSVD) dataset, reporting several previous works, all our baselines and attention model ablations, and our three multi-task models, using the four automated evaluation metrics. For each subsection below, we have reported the important training details inline, and refer to the supplementary for full details (e.g., learning rates and initialization).

Baseline Performance We first present all our baseline model choices (ablations) in Table 1. Our baselines represent the standard sequence-to-sequence model with three different visual feature types as well as those with attention mechanisms. Each baseline model is trained with three random seed initializations and the average is reported (for stable results). The final baseline model $\otimes$ instead uses an ensemble (E), which is a standard denoising method (Sutskever et al., 2014) that performs inference over ten randomly initialized models, i.e., at each time step t of the decoder, we generate a word based on the avg. of the likelihood probabilities from the ten models. Moreover, we use beam search with size 5 for all baseline models. Overall, the final baseline model with Inception-v4 features, attention, and 10-ensemble performs

[3] We use avg. of these four metrics on validation set to choose the best model, except for single-reference M-VAD dataset where we only report and choose based on METEOR.

Models	METEOR	CIDEr-D	ROUGE-L	BLEU-4
PREVIOUS WORK				
LSTM-YT (V) (Venugopalan et al., 2015b)	26.9	-	-	31.2
S2VT (V + A) (Venugopalan et al., 2015a)	29.8	-	-	-
Temporal Attention (G + C) (Yao et al., 2015)	29.6	51.7	-	41.9
LSTM-E (V + C) (Pan et al., 2016b)	31.0	-	-	45.3
Glove + DeepFusion (V) (E) (Venugopalan et al., 2016)	31.4	-	-	42.1
p-RNN (V + C) (Yu et al., 2016)	32.6	65.8	-	49.9
HNRE + Attention (G + C) (Pan et al., 2016a)	33.9	-	-	46.7
OUR BASELINES				
Baseline (V)	31.4	63.9	68.0	43.6
Baseline (G)	31.7	64.8	68.6	44.1
Baseline (I)	33.3	75.6	69.7	46.3
Baseline + Attention (V)	32.6	72.2	69.0	47.5
Baseline + Attention (G)	33.0	69.4	68.3	44.9
Baseline + Attention (I)	33.8	77.2	70.3	49.9
Baseline + Attention (I) (E) $\otimes$	35.0	84.4	71.5	52.6
OUR MULTI-TASK LEARNING MODELS				
$\otimes$ + Video Prediction (1-to-M)	35.6	88.1	72.9	54.1
$\otimes$ + Entailment Generation (M-to-1)	35.9	88.0	72.7	54.4
$\otimes$ + Video Prediction + Entailment Generation (M-to-M)	**36.0**	**92.4**	**72.8**	**54.5**

Table 1: Primary video captioning results on Youtube2Text (MSVD), showing previous works, our several strong baselines, and our three multi-task models. Here, V, G, I, C, A are short for VGGNet, GoogLeNet, Inception-v4, C3D, and AlexNet visual features; E = ensemble. The multi-task models are applied on top of our best video captioning baseline $\otimes$, with an ensemble. All the multi-task models are statistically significant over the baseline (discussed inline in the corresponding results sections).

well (and is better than all previous state-of-the-art), and so we next add all our novel multi-task models on top of this final baseline.

Multi-Task with Video Prediction (1-to-M) Here, the video captioning and unsupervised video prediction tasks share their encoder LSTM-RNN weights and image embeddings in a one-to-many multi-task setting. Two important hyperparameters tuned (on the validation set of captioning datasets) are the ratio of encoder vs decoder frames for video prediction on UCF-101 (where we found that 80% of frames as input and 20% for prediction performs best); and the mini-batch mixing ratio between the captioning and video prediction tasks (where we found $100 : 200$ works well). Table 1 shows a statistically significant improvement[4] in all metrics in comparison to the best baseline (non-multitask) model as well as w.r.t. all previous works, demonstrating the effectiveness of multi-task learning for video captioning with video prediction, even with unsupervised signals.

Multi-Task with Entailment Generation (M-to-1) Here, the video captioning and entailment generation tasks share their language decoder LSTM-RNN weights and word embeddings in a many-to-one multi-task setting. We observe that a mixing ratio of $100 : 50$ alternating mini-batches (between the captioning and entailment tasks) works well here. Again, Table 1 shows statistically significant improvements[5] in all the metrics in comparison to the best baseline model (and all previous works) under this multi-task setting. Note that in our initial experiments, our entailment generation model helped the video captioning task significantly more than the alternative approach of simply improving fluency by adding (or deep-fusing) an external language model (or pre-trained word embeddings) to the decoder (using both in-domain and out-of-domain language models), again because a caption also 'entails' a video in a logically-directed sense and hence this matches our captioning task better (also see results of Venugopalan et al. (2016) in Table 1).

Multi-Task with Video and Entailment Generation (M-to-M) Combining the above one-to-many and many-to-one multi-task learning models, our full model is the 3-task, many-to-many model (Fig. 4) where both the video encoder and the language decoder of the video captioning model are shared (and hence improved) with that of the unsupervised video prediction and entailment generation models, respectively.[6] A mixing ratio of $100 : 100 : 50$ alternate mini-batches

[4]Statistical significance of $p < 0.01$ for CIDEr-D and ROUGE-L, $p < 0.02$ for BLEU-4, $p < 0.03$ for METEOR, based on the bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994) with 100K samples.

[5]Statistical significance of $p < 0.01$ for all four metrics.

[6]We found the setting with unshared attention parameters to work best, likely because video captioning and video prediction prefer very different alignment distributions.

Models	M	C	R	B
Venugopalan (2015b)*	23.4	-	-	32.3
Yao et al. (2015)*	25.2	-	-	35.2
Xu et al. (2016)	25.9	-	-	36.6
Rank1: v2t_navigator	28.2	44.8	**60.9**	**40.8**
Rank2: Aalto	26.9	45.7	59.8	39.8
Rank3: VideoLAB	27.7	44.1	60.6	39.1
Our Model (**New Rank1**)	**28.8**	**47.1**	60.2	**40.8**

Table 2: Results on MSR-VTT dataset on the 4 metrics. *Results are reimplementations as per Xu et al. (2016). We also report the top 3 leaderboard systems – our model achieves the new rank 1 based on their ranking method.

Models	METEOR
Yao et al. (2015)	5.7
Venugopalan et al. (2015a)	6.7
Pan et al. (2016a)	6.8
Our M-to-M Multi-Task Model	**7.4**

Table 3: Results on M-VAD dataset.

of video captioning, unsupervised video prediction, and entailment generation, resp. works well. Table 1 shows that our many-to-many multi-task model again outperforms our strongest baseline (with statistical significance of $p < 0.01$ on all metrics), as well as all the previous state-of-the-art results by large absolute margins on all metrics. It also achieves significant improvements on some metrics over the one-to-many and many-to-one models.[7] Overall, we achieve the best results to date on YouTube2Text (MSVD) on all metrics.

5.2 Video Captioning on MSR-VTT, M-VAD

In Table 2, we also train and evaluate our final many-to-many multi-task model on two other video captioning datasets (using their standard splits; details in supplementary). First, we evaluate on the new MSR-VTT dataset (Xu et al., 2016). Since this is a recent dataset, we list previous works' results as reported by the MSR-VTT dataset paper itself.[8] We improve over all of these significantly. Moreover, they maintain a leaderboard[9] on this dataset and we also report the top 3 systems from it. Based on their ranking method, our multi-task model achieves the new rank 1 on this leaderboard. In Table 3, we further evaluate our model on the challenging movie-based M-VAD dataset, and again achieve improvements over all previous work (Venugopalan et al., 2015a; Pan et al., 2016a; Yao et al., 2015).[10]

Models	M	C	R	B
Entailment Generation	28.0	108.4	59.7	36.6
+Video Caption (M-to-1)	**28.7**	**114.5**	**60.8**	**38.9**

Table 4: Entailment generation results with the four metrics.

5.3 Entailment Generation Results

Above, we showed that the new entailment generation task helps improve video captioning. Next, we show that the video captioning task also inversely helps the entailment generation task. Given a premise, the task of entailment generation is to generate an entailed hypothesis. We use only the entailment pairs subset of the SNLI corpus for this, but with a multi-reference split setup to allow automatic metric evaluation and a zero train-test premise overlap (see Sec. 4.1). All the hyperparameter details (again tuned on the validation set) are presented in the supplementary. Table 4 presents the entailment generation results for the baseline (sequence-to-sequence with attention, 3-ensemble, beam search) and the multi-task model which uses video captioning (shared decoder) on top of the baseline. A mixing ratio of $100:20$ alternate mini-batches of entailment generation and video captioning (resp.) works well.[11] The multi-task model achieves stat. significant ($p < 0.01$) improvements over the baseline on all metrics, thus demonstrating that video captioning and entailment generation both mutually help each other.

5.4 Human Evaluation

In addition to the automated evaluation metrics, we present pilot-scale human evaluations on the YouTube2Text (Table 1) and entailment generation (Table 4) results. In each case, we compare our strongest baseline with our final multi-task model by taking a random sample of 200 generated captions (or entailed hypotheses) from the test set and removing the model identity to anonymize the two models, and ask the human evaluator to choose the better model based on *relevance* and *coherence* (described in Sec. 4.2). As shown in Table 5, the multi-task models are always better than the strongest baseline for both video captioning and entailment generation, on both relevance

[7] Many-to-many model's improvements have a statistical significance of $p < 0.01$ on all metrics w.r.t. baseline, and $p < 0.01$ on CIDEr-D w.r.t. both one-to-many and many-to-one models, and $p < 0.04$ on METEOR w.r.t. one-to-many.

[8] In their updated supplementary at `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/10/cvpr16.supplementary.pdf`

[9] `http://ms-multimedia-challenge.com/leaderboard`

[10] Following previous work, we only use METEOR because M-VAD only has a single reference caption per video.

[11] Note that this many-to-one model prefers a different mixing ratio and learning rate than the many-to-one model for improving video captioning (Sec. 5.1), because these hyperparameters depend on the primary task being improved, as also discussed in previous work (Luong et al., 2016).

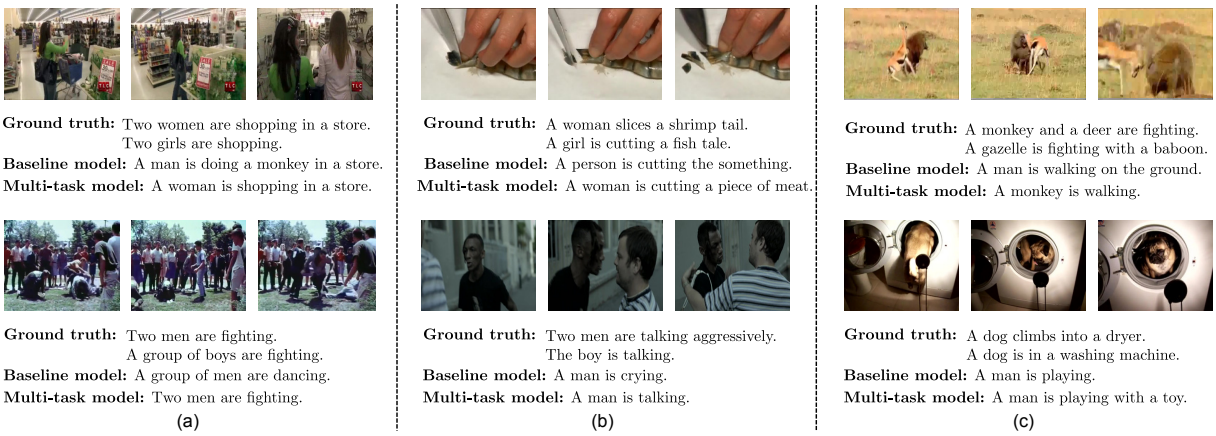


Figure 5: Examples of generated video captions on the YouTube2Text dataset: (a) complex examples where the multi-task model performs better than the baseline; (b) ambiguous examples (i.e., ground truth itself confusing) where multi-task model still correctly predicts one of the possible categories (c) complex examples where both models perform poorly.

	YouTube2Text		Entailment	
	Relev.	Coher.	Relev.	Coher.
Not Distinguish.	65.0%	93.0%	73.5%	94.5%
Baseline Wins	14.0%	1.0%	12.5%	1.5%
Multi-Task Wins	**21.0%**	**6.0%**	**15.0%**	**4.0%**

Table 5: Human evaluation on captioning and entailment.

Given Premise	Generated Entailment
a man on stilts is playing a tuba for money on the boardwalk	a man is playing an instrument
a girl looking through a large telescope on a school trip	a girl is looking at something
several young people sit at a table playing poker	people are playing a game
the stop sign is folded up against the side of the bus	the sign is not moving
a blue and silver monster truck making a huge jump over crushed cars	a truck is being driven

Table 6: Examples of our multi-task model's generated entailment hypotheses given a premise.

and coherence, and with similar improvements (2-7%) as the automatic metrics (shown in Table 1).

5.5 Analysis

Fig. 5 shows video captioning generation results on the YouTube2Text dataset where our final M-to-M multi-task model is compared with our strongest attention-based baseline model for three categories of videos: (a) complex examples where the multi-task model performs better than the baseline; (b) ambiguous examples (i.e., ground truth itself confusing) where multi-task model still correctly predicts one of the possible categories (c) complex examples where both models perform poorly. Overall, we find that the multi-task model generates captions that are better at both temporal action prediction and logical entailment (i.e., correct subset of full video premise) w.r.t. the ground truth captions. The supplementary also provides ablation examples of improvements by the 1-to-M video prediction based multi-task model alone, as well as by the M-to-1 entailment based multi-task model alone (over the baseline).

On analyzing the cases where the baseline is better than the final M-to-M multi-task model, we find that these are often scenarios where the multi-task model's caption is also correct but the baseline caption is a bit more specific, e.g., "a man is holding a gun" vs "a man is shooting a gun".

Finally, Table 6 presents output examples of our entailment generation multi-task model (Sec. 5.3), showing how the model accurately learns to produce logically implied subsets of the premise.

6 Conclusion

We presented a multimodal, multi-task learning approach to improve video captioning by incorporating temporally and logically directed knowledge via video prediction and entailment generation tasks. We achieve the best reported results (and rank) on three datasets, based on multiple automatic and human evaluations. We also show mutual multi-task improvements on the new entailment generation task. In future work, we are applying our entailment-based multi-task paradigm to other directed language generation tasks such as image captioning and document summarization.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was partially supported by a Google Faculty Research Award, an IBM Faculty Award, a Bloomberg Data Science Research Grant, and NVidia GPU awards.

References

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2007. Multi-task feature learning. In *NIPS*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer, pages 95–133.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 190–200.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325* .

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*. IEEE, pages 248–255.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *EACL*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2013. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *CVPR*. pages 2712–2719.

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

Haiqi Huang, Yueming Lu, Fangwei Zhang, and Songlin Sun. 2013. A multi-modal clustering method for web videos. In *International Conference on Trustworthy Computing and Services*. pages 163–169.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. 2014. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *In SemEval*. pages 732–742.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Vladyslav Kolesnyk, Tim Rocktäschel, and Sebastian Riedel. 2016. Generating natural language inference chains. *arXiv preprint arXiv:1606.01404* .

Abhishek Kumar and Hal Daumé III. 2012. Learning task grouping and overlap in multi-task learning. In *ICML*.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. *Proc. SemEval* 2:5.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 workshop*. volume 8.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. 2016a. Hierarchical recurrent neural encoder for video representation with application to captioning. In *CVPR*. pages 1029–1038.

Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. 2016b. Jointly modeling embedding and translation to bridge video and language. In *CVPR*. pages 4594–4602.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*. pages 311–318.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* .

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. 2015. Unsupervised learning of video representations using lstms. In *ICML*. pages 843–852.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. pages 3104–3112.

Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. In *CoRR*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*. pages 1–9.

Jesse Thomason, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Raymond J Mooney. 2014. Integrating language and vision to generate natural language descriptions of videos in the wild. In *COLING*.

Atousa Torabi, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Using descriptive video services to create a large data source for video annotation research. *arXiv preprint arXiv:1503.01070* .

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *CVPR*. pages 4566–4575.

Subhashini Venugopalan, Lisa Anne Hendricks, Raymond Mooney, and Kate Saenko. 2016. Improving lstm-based video description with linguistic knowledge mined from text. In *EMNLP*.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015a. Sequence to sequence-video to text. In *CVPR*. pages 4534–4542.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015b. Translating videos to natural language using deep recurrent neural networks. In *NAACL HLT*.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *CVPR*. pages 5288–5296.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *CVPR*. pages 4507–4515.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *CVPR*.

Enriching Complex Networks with Word Embeddings for Detecting Mild Cognitive Impairment from Speech Transcripts

Leandro B. dos Santos[1], Edilson A. Corrêa Jr[1], Osvaldo N. Oliveira Jr[2], Diego R. Amancio[1], Letícia L. Mansur[3], Sandra M. Aluísio[1]

[1] Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, São Paulo, Brazil
[2] São Carlos Institute of Physics, University of São Paulo, São Carlos, São Paulo, Brazil
[3] Department of Physiotherapy, Speech Pathology and Occupational Therapy, University of São Paulo, São Paulo, São Paulo, Brazil
{leandrobs,edilsonacjr,lamansur}@usp.br, chu@ifsc.usp.br
{diego,sandra}@icmc.usp.br

Abstract

Mild Cognitive Impairment (MCI) is a mental disorder difficult to diagnose. Linguistic features, mainly from parsers, have been used to detect MCI, but this is not suitable for large-scale assessments. MCI disfluencies produce non-grammatical speech that requires manual or high precision automatic correction of transcripts. In this paper, we modeled transcripts into complex networks and enriched them with word embedding (CNE) to better represent short texts produced in neuropsychological assessments. The network measurements were applied with well-known classifiers to automatically identify MCI in transcripts, in a binary classification task. A comparison was made with the performance of traditional approaches using Bag of Words (BoW) and linguistic features for three datasets: DementiaBank in English, and Cinderella and Arizona-Battery in Portuguese. Overall, CNE provided higher accuracy than using only complex networks, while Support Vector Machine was superior to other classifiers. CNE provided the highest accuracies for DementiaBank and Cinderella, but BoW was more efficient for the Arizona-Battery dataset probably owing to its short narratives. The approach using linguistic features yielded higher accuracy if the transcriptions of the Cinderella dataset were manually revised. Taken together, the results indicate that complex networks enriched with embedding is promising for detecting MCI in large-scale assessments.

1 Introduction

Mild Cognitive Impairment (MCI) can affect one or multiple cognitive domains (e.g. memory, language, visuospatial skills and executive functions), and may represent a pre-clinical stage of Alzheimer's disease (AD). The impairment that affects memory, referred to as amnestic MCI, is the most frequent, with the highest conversion rate for AD, at 15% per year versus 1 to 2% for the general population. Since dementias are chronic and progressive diseases, their early diagnosis ensures a greater chance of success to engage patients in non-pharmacological treatment strategies such as cognitive training, physical activity and socialization (Teixeira et al., 2012).

Language is one of the most efficient information sources to assess cognitive functions. Changes in language usage are frequent in patients with dementia and are normally first recognized by the patients themselves or their family members. Therefore, the automatic analysis of discourse production is promising in diagnosing MCI at early stages, which may address potentially reversible factors (Muangpaisan et al., 2012). Proposals to detect language-related impairment in dementias include machine learning (Jarrold et al., 2010; Roark et al., 2011; Fraser et al., 2014, 2015), magnetic resonance imaging (Dyrba et al., 2015), and data screening tests added to demographic information (Weakley et al., 2015). Discourse production (mainly narratives) is attractive because it allows the analysis of linguistic microstructures, including phonetic-phonological, morphosyntactic and semantic-lexical components, as well as semantic-pragmatic macrostructures.

Automated discourse analysis based on Natural Language Processing (NLP) resources and tools to diagnose dementias via machine learning methods has been used for English language (Lehr et al.,

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1284–1296
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1118

2012; Jarrold et al., 2014; Orimaye et al., 2014; Fraser et al., 2015; Davy et al., 2016) and for Brazilian Portuguese (Aluísio et al., 2016). A variety of features are required for this analysis, including Part-of-Speech (PoS), syntactic complexity, lexical diversity and acoustic features. Producing robust tools to extract these features is extremely difficult because speech transcripts used in neuropsychological evaluations contain disfluencies (repetitions, revisions, paraphasias) and patient's comments about the task being evaluated. Another problem in using linguistic knowledge is the high dependence on manually created resources, such as hand-crafted linguistic rules and/or annotated corpora. Even when traditional statistical techniques (Bag of Words or ngrams) are applied, problems still appear in dealing with disfluencies, because mispronounced words will not be counted together. Indeed, other types of disfluencies (repetition, amendments, patient's comments about the task) will be counted, thus increasing the vocabulary.

An approach applied successfully to several areas of NLP (Mihalcea and Radev, 2011), which may suffer less from the problems mentioned above, relies on the use of complex networks and graph theory. The word adjacency network model (i Cancho and Solé, 2001; Roxas and Tapang, 2010; Amancio et al., 2012a; Amancio, 2015b) has provided good results in text classification (de Arruda et al., 2016) and related tasks, namely author detection (Amancio, 2015a), identification of literary movements (Amancio et al., 2012c), authenticity verification (Amancio et al., 2013) and word sense discrimination (Amancio et al., 2012b).

In this paper, we show that speech transcripts (narratives or descriptions) can be modeled into complex networks that are enriched with word embedding in order to better represent short texts produced in these assessments. When applied to a machine learning classifier, the complex network features were able to distinguish between control participants and mild cognitive impairment participants. Discrimination of the two classes could be improved by combining complex networks with linguistic and traditional statistical features.

With regard to the task of detecting MCI from transcripts, this paper is, to the best of our knowledge, the first to: a) show that classifiers using features extracted from transcripts modeled into complex networks enriched with word embedding present higher accuracy than using only complex networks for 3 datasets; and b) show that for languages that do not have competitive dependency and constituency parsers to exploit syntactic features, e.g. Brazilian Portuguese, complex networks enriched with word embedding constitute a source to extract new, language independent features from transcripts.

2 Related Work

Detection of memory impairment has been based on linguistic, acoustic, and demographic features, in addition to scores of neuropsychological tests. Linguistic and acoustic features were used to automatically detect aphasia (Fraser et al., 2014); and AD (Fraser et al., 2015) or dementia (Orimaye et al., 2014) in the public corpora of DementiaBank[1]. Other studies distinguished different types of dementia (Garrard et al., 2014; Jarrold et al., 2014), in which speech samples were elicited using the Picnic picture of the Western Aphasia Battery (Kertesz, 1982). Davy et al. (2016) also used the Picnic scene to detect MCI, where the subjects were asked to write (by hand) a detailed description of the scene.

As for automatic detection of MCI in narrative speech, Roark et al. (2011) extracted speech features and linguistic complexity measures of speech samples obtained with the Wechsler Logical Memory (WLM) subtest (Wechsler et al., 1997), and Lehr et al. (2012) fully automatized the WLM subtest. In this test, the examiner tells a short narrative to a subject, who then retells the story to the examiner, immediately and after a 30-minute delay. WLM scores are obtained by counting the number of story elements recalled.

Tóth et al. (2015) and Vincze et al. (2016) used short animated films to evaluate immediate and delayed recalls in MCI patients who were asked to talk about the first film shown, then about their previous day, and finally about another film shown last. Tóth et al. (2015) adopted automatic speech recognition (ASR) to extract a phonetic level segmentation, which was used to calculate acoustic features. Vincze et al. (2016) used speech, morphological, semantic, and demographic features collected from their speech transcripts to automatically identify patients suffering from MCI.

For the Portuguese language, machine learning

[1] talkbank.org/DementiaBank/

algorithms were used to identify subjects with AD and MCI. Aluísio et al. (2016) used a variety of linguistic metrics, such as syntactic complexity, idea density (da Cunha et al., 2015), and text cohesion through latent semantics. NLP tools with high precision are needed to compute these metrics, which is a problem for Portuguese since no robust dependency or constituency parsers exist. Therefore, the transcriptions had to be manually revised; they were segmented into sentences, following a semantic-structural criterion and capitalization was applied. The authors also removed disfluencies and inserted omitted subjects when they were hidden, in order to reduce parsing errors. This process is obviously expensive, which has motivated us to use complex networks in the present study to model transcriptions and avoid a manual preprocessing step.

3 Modeling and Characterizing Texts as Complex Networks

The theory and concepts of complex networks have been used in several NLP tasks (Mihalcea and Radev, 2011; Cong and Liu, 2014), such as text classification (de Arruda et al., 2016), summarization (Antiqueira et al., 2009; Amancio et al., 2012a) and word sense disambiguation (Silva and Amancio, 2012). In this study, we used the word co-occurrence model (also called word adjacency model) because most of the syntactical relations occur among neighboring words (i Cancho et al., 2004). Each distinct word becomes a node and words that are adjacent in the text are connected by an edge. Mathematically, a network is defined as an undirected graph $G = \{V, E\}$, formed by a set $V = \{v_1, v_2, ..., v_n\}$ of nodes (words) and a set $E = \{e_1, e_2, ..., e_m\}$ of edges (co-occurrence) that are represented by an adjacency matrix A, whose elements A_{ij} are equal to 1 whenever there is an edge connecting nodes (words) i and j, and equal to 0 otherwise.

Before modeling texts into complex networks, it is often necessary to do some preprocessing in the raw text. Preprocessing starts with tokenization where each document/text is divided into tokens (meaningful elements, e.g., words and punctuation marks) and then *stopwords* and punctuation marks are removed, since they have little semantic meaning. One last step we decided to eliminate from the preprocessing pipeline is lemmatization, which transforms each word into its canonical

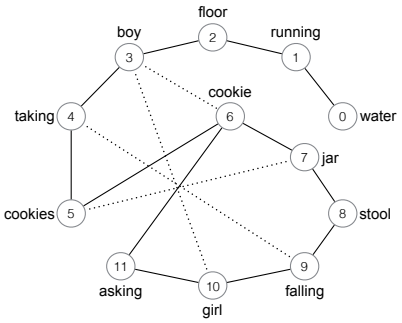
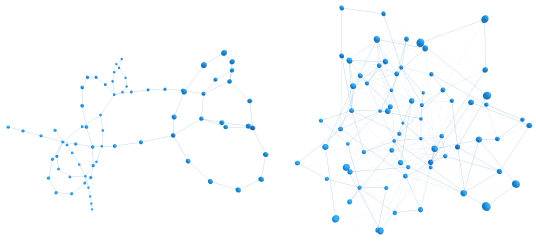


Figure 1: Example of co-occurrence network enriched with semantic information for the following transcription: "*The water's running on the floor. Boy's taking cookies out of cookie out of the cookie jar. The stool is falling over. The girl was asking for a cookie.*". The solid edges of the network represent co-occurrence edges and the dotted edges represent connections between words that had similarity higher than 0.5.

form. This decision was made based on two factors. First, a recent work has shown that lemmatization has little or no influence when network modeling is adopted in related tasks (Machicao et al., 2016). Second, the lemmatization process requires part-of-speech (POS) tagging that may introduce undesirable noises/errors in the text, since the transcriptions in our work contain disfluencies.

Another problem with transcriptions in our work is their size. As demonstrated by Amancio (2015c), classification of small texts using networks can be impaired, since short texts have almost linear networks, and the topological measures of these networks have little or no information relevant to classification. To solve this problem, we adapted the approach of inducing language networks from word embeddings, proposed by Perozzi et al. (2014) to enrich the networks with semantic information. In their work, language networks were generated from continuous word representations, in which each word is represented by a dense, real-valued vector obtained by training neural networks in the language model task (or variations, such as context prediction) (Bengio et al., 2003; Collobert et al., 2011; Mikolov et al., 2013a,b). This structure is known to capture syntactic and semantic information. Perozzi et al. (2014), in particular, take advantage of word embeddings to build networks where each word is



(a) (b)

Figure 2: Example of (a) co-occurrence network created for a transcript of the Cookie Theft dataset (see Supplementary Information, Section A) and (b) the same co-occurrence network enriched with semantic information. Note that (b) is a more informative network than (a), since (a) is practically a linear network.

a vertex and edges are defined by similarity between words established by the proximity of the word vectors.

Following this methodology, in our model we added new edges to the co-occurrence networks considering similarities between words, that is, for all pairs of words in the text that were not connected, an edge was created if their vectors (from word embedding) had a cosine similarity higher than a given threshold. Figure 1 shows an example of a co-occurrence network enriched by similarity links (the dotted edges). The gain in information by enriching a co-occurrence network with semantic information is readily apparent in Figure 2.

4 Datasets, Features and Methods

4.1 Datasets

The datasets[2] used in our study consisted of: (i) manually segmented and transcribed samples from the DementiaBank and Cinderella story and (ii) transcribed samples of Arizona Battery for Communication Disorders of Dementia (ABCD) automatically segmented into sentences, since we are working towards a fully automated system to detect MCI in transcripts and would like to evaluate a dataset which was automatically processed.

The DementiaBank dataset is composed of short English descriptions, while the Cinderella dataset contains longer Brazilian Portuguese narratives. ABCD dataset is composed of very short narratives, also in Portuguese. Below, we describe in further detail the datasets, participants, and the task in which they were used.

[2]All datasets are made available in the same representations used in this work, upon request to the authors.

4.1.1 The Cookie Theft Picture Description Dataset

The clinical dataset used for the English language was created during a longitudinal study conducted by the University of Pittsburgh School of Medicine on Alzheimer's and related dementia, funded by the National Institute of Aging. To be eligible for inclusion in the study, all participants were required to be above 44 years of age, have at least 7 years of education, no history of nervous system disorders nor be taking neuroleptic medication, have an initial Mini-Mental State Exam (MMSE) score of 10 or greater, and be able to give informed consent. The dataset contains transcripts of verbal interviews with AD and related Dementia patients, including those with MCI (for further details see (Becker et al., 1994)).

We used 43 transcriptions with MCI in addition to another 43 transcriptions sampled from 242 healthy elderly people to be used as the control group. Table 1 shows the demographic information for the two diagnostic groups.

Demographic	Control	MCI
Avg. Age (SD)	64.1 (7.2)	69.3 (8.2)
No. of Male/Female	23/20	27/16

Table 1: Demographic information of participants in the Cookie Theft dataset.

For this dataset, interviews were conducted in English and narrative speech was elicited using the Cookie Theft picture (Goodglass et al., 2001) (Figure 3 from Goodglass et al. (2001) in Section A.1). During the interview, patients were given the picture and were told to discuss everything they could see happening in the picture. The patients' verbal utterances were recorded and then transcribed into the CHAT (Codes for the Human Analysis of Transcripts) transcription format (MacWhinney, 2000).

We extracted the word-level transcript patient sentences from the CHAT files and discarded the annotations, as our goal was to create a fully automated system that does not require the input of a human annotator. We automatically removed filled pauses such as *uh*, *um* , *er* , and *ah* (e.g. *uh it seems to be summer out*), short false starts (e.g. *just t the ones*), and repetition (e.g. *mother's finished certain of the the dishes*), as in (Fraser et al.,

2015). The control group had an average of 9.58 sentences per narrative, with each sentence having an average of 9.18 words; while the MCI group had an average of 10.97 sentences per narrative, with 10.33 words per sentence in average.

4.1.2 The Cinderella Narrative Dataset

The dataset examined in this study included 20 subjects with MCI and 20 normal elderly control subjects, as diagnosed at the Medical School of the University of São Paulo (FMUSP). Table 2 shows the demographic information of the two diagnostic groups, which were also used in Aluísio et al. (2016).

Demographic	Control	MCI
Avg. Age (SD)	74.8 (11.3)	73.3 (5.9)
Avg. Years of Education (SD)	11.4 (2.6)	10.8 (4.5)
No. of Male/Female	27/16	29/14

Table 2: Demographic information of participants in the Cinderella dataset.

The criteria used to diagnose MCI came from Petersen (2004). Diagnostics were carried out by a multidisciplinary team consisting of psychiatrists, geriatricians, neurologists, neuropsychologists, speech pathologists, and occupational therapists, by a criterion of consensus. Inclusion criteria for the control group were elderlies with no cognitive deficits and preservation of functional capacity in everyday life. The exclusion criteria for the normal group were: poorly controlled clinical diseases, sensitive deficits that were not being compensated for and interfered with the performance in tests, and other neurological or psychiatric diagnoses associated with dementia or cognitive deficits and use of medications in doses that affected cognition.

Speech narrative samples were elicited by having participants tell the Cinderella story; participants were given as much time as they needed to examine a picture book illustrating the story (Figure 4 in Section A). When each participant had finished looking at the pictures, the examiner asked the subject to tell the story in their own words, as in Saffran et al. (1989). The time was recorded, but there was no limit imposed to the narrative length. If the participant had difficulty initiating or continuing speech, or took a long pause, an evaluator would use the stimulus question "What happens next ?", seeking to encourage the participant to continue his/her narrative. When the subject was unable to proceed with the narrative, the examiner asked if he/she had finished the story and had something to add. Each speech sample was recorded and then manually transcribed at the word level following the NURC/SP N. 338 EF and 331 D2 transcription norms[3].

Other tests were applied after the narrative, in the following sequence: phonemic verbal fluency test, action verbal fluency, Camel and Cactus test (Bozeat et al., 2000), and Boston Naming test (Kaplan et al., 2001), in order to diagnose the groups.

Since our ultimate goal is to create a fully automated system that does not require the input of a human annotator, we manually segmented sentences to simulate a high-quality ASR transcript with sentence segmentation, and we automatically removed the disfluencies following the same guidelines of TalkBank project. However, other disfluencies (revisions, elaboration, paraphasias and comments about the task) were kept. The control group had an average of 30.80 sentences per narrative, and each sentence averaged 12.17 words. As for the MCI group, it had an average of 29.90 sentences per narrative, and each sentence averaged 13.03 words.

We also evaluated a different version of the dataset used in Aluísio et al. (2016), where narratives were manually annotated and revised to improve parsing results. The revision process was the following: (i) in the original transcript, segments with hesitations or repetitions of more than one word or segment of a single word were annotated to become a feature and then removed from the narrative to allow the extraction of features from parsing; (ii) empty emissions, which were comments unrelated to the topic of narration or confirmations, such as "*né*" (alright), were also annotated and removed; (iii) prolongations of vowels, short pauses and long pauses were also annotated and removed; and (iv) omitted subjects in sentences were inserted. In this revised dataset, the control group had an average of 45.10 sentences per narrative, and each sentence averaged 8.17 words. The MCI group had an average of 31.40 sentences per narrative, with each sentence averaging 10.91 words.

4.1.3 The ABCD Dataset

The subtest of immediate/delayed recall of narratives of the ABCD battery was administered to 23

[3] albertofedel.blogspot.com.br/2010_11_01_archive.html

participants with a diagnosis of MCI and 20 normal elderly control participants, as diagnosed at the Medical School of the University of São Paulo (FMUSP).

MCI subjects produced 46 narratives while the control group produced 39 ones. In order to carry out experiments with a balanced corpus, as with the previous two datasets, we excluded seven transcriptions from the MCI group. We used the automatic sentence segmentation method referred to as DeepBond (Treviso et al., 2017) in the transcripts.

Table 3 shows the demographic information. The control group had an average of 5.23 sentences per narrative, with 11 words per sentence on average, and the MCI group had an average of 4.95 sentences per narrative, with an average of 12.04 words per sentence. Interviews were conducted in Portuguese and the subject listened to the examiner read a short narrative. The subject then retold the narrative to the examiner twice: once immediately upon hearing it and again after a 30-minute delay (Bayles and Tomoeda, 1991). Each speech sample was recorded and then manually transcribed at the word level following the NURC/SP N. 338 EF and 331 D2 transcription norms.

Demographic	Control	MCI
Avg. Age (SD)	61 (7.5)	72,0 (7.4)
Avg. Years of Education (SD)	16 (7.6)	13.3 (4.2)
No. of Male/Female	6/14	16/7

Table 3: Demographic information of participants in the ABCD dataset.

4.2 Features

Features of three distinct natures were used to classify the transcribed texts: topological metrics of co-occurrence networks, linguistic features and bag of words representations.

4.2.1 Topological Characterization of Networks

Each transcription was mapped into a co-occurrence network, and then enriched via word embeddings using the cosine similarity of words. Since the occurrence of out-of-vocabulary words is common in texts of neuropsychological assessments, we used the method proposed by Bojanowski et al. (2016) to generate word embeddings. This method extends the skip-gram model to use character-level information, with each word being represented as a bag of character n-grams. It provides some improvement in comparison with the traditional skip-gram model in terms of syntactic evaluation (Mikolov et al., 2013b) but not for semantic evaluation.

Once the network has been enriched, we characterize its topology using the following ten measurements:

1. **PageRank:** is a centrality measurement that reflects the relevance of a node based on its connections to other relevant nodes (Brin and Page, 1998);
2. **Betweenness:** is a centrality measurement that considers a node as relevant if it is highly accessed via shortest paths. The betweenness of a node v is defined as the fraction of shortest paths going through node v;
3. **Eccentricity:** of a node is calculated by measuring the shortest distance from the node to all other vertices in the graph and taking the maximum;
4. **Eigenvector centrality:** is a measurement that defines the importance of a node based on its connectivity to high-rank nodes;
5. **Average Degree of the Neighbors of a Node:** is the average of the degrees of all its direct neighbors;
6. **Average Shortest Path Length of a Node:** is the average distance between this node and all other nodes of the network;
7. **Degree:** is the number of edges connected to the node;
8. **Assortativity Degree:** or degree correlation measures the tendency of nodes to connect to other nodes that have similar degree;
9. **Diameter:** is defined as the maximum shortest path;
10. **Clustering Coefficient:** measures the probability that two neighbors of a node are connected.

Most of the measurements described above are local measurements, i.e. each node i possesses a value X_i, so we calculated the average $\mu(X)$, standard deviation $\sigma(X)$ and skewness $\gamma(X)$ for each measurement (Amancio, 2015b).

4.2.2 Linguistic Features

Linguistic features for classification of neuropsychological assessments have been used in several studies (Roark et al., 2011; Jarrold et al., 2014; Fraser et al., 2014; Orimaye et al., 2014; Fraser et al., 2015; Vincze et al., 2016; Davy et al., 2016). We used the Coh-Metrix[4](Graesser et al., 2004) tool to extract features from English transcripts, resulting in 106 features. The metrics are divided into eleven categories: Descriptive, Text Easability Principal Component, Referential Cohesion, Latent Semantic Analysis (LSA), Lexical Diversity, Connectives, Situation Model, Syntactic Complexity, Syntactic Pattern Density, Word Information, and Readability (Flesch Reading Ease, Flesch-Kincaid Grade Level, Coh-Metrix L2 Readability).

For Portuguese, Coh-Metrix-Dementia (Aluísio et al., 2016) was used. The metrics affected by constituency and dependency parsing were not used because they are not robust with disfluencies. Metrics based on manual annotation (such as proportion short pauses, mean pause duration, mean number of empty words, and others) were also discarded. The metrics of Coh-Metrix-Dementia are divided into twelve categories: Ambiguity, Anaphoras, Basic Counts, Connectives, Co-reference Measures, Content Word Frequencies, Hypernyms, Logic Operators, Latent Semantic Analysis, Semantic Density, Syntactical Complexity, and Tokens. The metrics used are shown in detail in Section A.2. In total, 58 metrics were used, from the 73 available on the website[5].

4.2.3 Bag of Words

The representation of text collections under the BoW assumption (i.e., with no information relating to word order) has been a robust solution for text classification. In this methodology, transcripts are represented by a table in which the columns represent the terms (or existing words) in the transcripts and the values represent frequency of a term in a document.

4.3 Classification Algorithms

In order to quantify the ability of the topological characterization of networks, linguistic metrics and BoW features were used to distinguish subjects with MCI from healthy controls. We employed four machine learning algorithms to induce classifiers from a training set. These techniques were the Gaussian Naive Bayes (GNB), k-Nearest Neighbor (k-NN), Support Vector Machine (SVM), linear and radial bases functions (RBF), and Random Forest (RF). We also combined these classifiers through ensemble and multi-view learning. In ensemble learning, multiple models/classifiers are generated and combined using a majority vote or the average of class probabilities to produce a single result (Zhou, 2012).

In multi-view learning, multiple classifiers are trained in different feature spaces and thus combined to produce a single result. This approach is an elegant solution in comparison to combining all features in the same vector or space, for two main reasons. First, combination is not a straightforward step and may lead to noise insertion since the data have different natures. Second, using different classifiers for each feature space allows for different weights to be given for each type of feature, and these weights can be learned by a regression method to improve the model. In this work, we used majority voting to combine different feature spaces.

5 Experiments and Results

All experiments were conducted using the Scikit-learn[6] (Pedregosa et al., 2011), with classifiers evaluated on the basis of classification accuracy i.e. the total proportion of narratives which were correctly classified. The evaluation was performed using 5-fold cross-validation instead of the well-accepted 10-fold cross-validation because the datasets in our study were small and the test set would have shrunk, leading to less precise measurements of accuracy. The threshold parameter was optimized with the best values being 0.7 in the Cookie Theft dataset and 0.4 in both the Cinderella and ABCD datasets.

We used the model proposed by Bojanowski et al. (2016) with default parameters (100 dimensional embeddings, context window equal to 5 and 5 epochs) to generate word embedding. We trained the models in Portuguese and English Wikipedia dumps from October and November 2016 respectively.

The accuracy in classification is given in Tables 4 through 6. CN, CNE, LM, and BoW denote, respectively, complex networks, complex network

[4] cohmetrix.com
[5] http://143.107.183.175:22380
[6] http://scikit-learn.org

enriched with embedding, linguistic metrics and Bag of Words, and CNE-LM, CNE-BoW, LM-BoW and CNE-LM-BoW refer to combinations of the feature spaces (multiview learning), using the majority vote. Cells with the "–" sign mean that it was not possible to apply majority voting because there were two classifiers. The last line represents the use of an ensemble of machine learning algorithms, in which the combination used was the majority voting in both ensemble and multiview learning.

In general, CNE outperforms the approach using only complex networks (CN), while SVM (Linear or RBF kernel) provides higher accuracy than other machine learning algorithms. The results for the three datasets show that characterizing transcriptions into complex networks is competitive with other traditional methods, such as the use of linguistic metrics. In fact, among the three types of features, using enriched networks (CNE) provided the highest accuracies in two datasets (Cookie Theft and original Cinderella). For the ABCD dataset, which contains short narratives, the small length of the transcriptions may have had an effect, since BoW features led to the highest accuracy. In the case of the revised Cinderella dataset, segmented into sentences and capitalized as reported in Aluísio et al. (2016), Table 7 shows that the manual revision was an important factor, since the highest accuracies were obtained with the approach based on linguistic metrics (LM). However, this process of manually removing disfluencies demands time; therefore it is not practical for large-scale assessments.

Ensemble and multi-view learning were helpful for the Cookie Theft dataset, in which multi-view learning achieved the highest accuracy (65% of accuracy for narrative texts, a 3% of improvement compared to the best individual classifier). However, neither multi-view or ensemble learning enhanced accuracy in the Cinderella dataset, where SVM-RBF with CNE space achieved the highest accuracy (65%). For the ABCD dataset, multiview CNE-LM-BoW with SVM-RBF and KNN classifiers improved the accuracy to 4% and 2%, respectively. Somewhat surprising were the results of SVM with linear kernel in BoW feature space (75% of accuracy).

6 Conclusions and Future Work

In this study, we employed metrics of topological properties of CN in a machine learning classification approach to distinguish between healthy patients and patients with MCI. To the best of our knowledge, these metrics have never been used to detect MCI in speech transcripts; CN were enriched with word embeddings to better represent short texts produced in neuropsychological assessments. The topological properties of CN outperform traditional linguistic metrics in individual classifiers' results. Linguistic features depend on grammatical texts to present good results, as can be seen in the results of the manually processed Cinderella dataset (Table 7). Furthermore, we found that combining machine and multi-view learning can improve accuracy. The accuracies found here are comparable to the values reported by other authors, ranging from 60% to 85% (Prud'hommeaux and Roark, 2011; Lehr et al., 2012; Tóth et al., 2015; Vincze et al., 2016), which means that it is not easy to distinguish between healthy subjects and those with cognitive impairments. The comparison with our results is not straightforward, though, because the databases used in the studies are different. There is a clear need for publicly available datasets to compare different methods, which would optimize the detection of MCI in elderly people.

In future work, we intend to explore other methods to enrich CN, such as the Recurrent Language Model, and use other metrics to characterize an adjacency network. The pursuit of these strategies is relevant because language is one of the most efficient information sources to evaluate cognitive functions, commonly used in neuropsychological assessments. As this work is ongoing, we will keep collecting new transcriptions of the ABCD retelling subtest to increase the corpus size and obtain more reliable results in our studies. Our final goal is to apply neuropsychological assessment batteries, such as the ABCD retelling subtest, to mobile devices, specifically tablets. This adaptation will enable large-scale applications in hospitals and facilitate the maintenance of application history in longitudinal studies, by storing the results in databases immediately after the test application.

Classifier	CN	CNE	LM	BoW	CNE-LM	CNE-BoW	LM-BoW	CNE-LM-BoW
SVM-Linear	52	55	56	59	–	–	–	60
SVM-RBF	56	**62**	**58**	**60**	–	–	–	**65**
k-NN	**59**	61	46	57	–	–	–	59
RF	52	47	45	48	–	–	–	50
G-NB	51	48	56	55	–	–	–	50
Ensemble	56	60	54	58	57	60	63	**65**

Table 4: Classification accuracy achieved on Cookie Theft dataset.

Classifier	CN	CNE	LM	BoW	CNE-LM	CNE-BoW	LM-BoW	CNE-LM-BoW
SVM-Linear	52	60	**52**	50	–	–	–	**52**
SVM-RBF	**57**	**65**	47	37	–	–	–	50
k-NN	47	50	47	37	–	–	–	37
RF	55	57	47	45	–	–	–	**52**
G-NB	47	52	47	**55**	–	–	–	**52**
Ensemble	52	60	50	37	57	52	50	47

Table 5: Classification accuracy achieved on Cinderella dataset.

Classifier	CN	CNE	LM	BoW	CNE-LM	CNE-BoW	LM-BoW	CNE-LM-BoW
SVM-Linear	56	**69**	51	**75**	–	–	–	**74**
SVM-RBF	54	57	66	67	–	–	–	71
k-NN	56	56	69	63	–	–	–	71
RF	54	62	**70**	64	–	–	–	69
G-NB	**61**	55	55	65	–	–	–	65
Ensemble	55	61	62	72	69	68	75	73

Table 6: Classification accuracy achieved on ABCD dataset.

Classifier	CN	CNE	LM	BoW
SVM-Linear	50	65	65	52
SVM-RBF	**57**	**67**	**72**	**55**
KNN	42	47	55	50
RF	52	47	70	45
G-NB	52	65	62	45
Ensemble	52	60	**72**	45

Table 7: Classification accuracy achieved on Cinderella dataset manually processed to revise non-grammatical sentences.

Acknowledgments

This work was supported by CAPES, CNPq, FAPESP, and Google Research Awards in Latin America. We would like to thank NVIDIA for their donation of GPU.

References

Sandra M. Aluísio, Andre L. da Cunha, and Carolina Scarton. 2016. Evaluating progression of alzheimer's disease by regression and classification methods in a narrative language test in portuguese. In João Silva, Ricardo Ribeiro, Paulo Quaresma, André Adami, and António Branco, editors, *International Conference on Computational Processing of the Portuguese Language*. Springer, pages 109–114. https://doi.org/10.1007/978-3-319-41552-9_10.

Diego R. Amancio. 2015a. Authorship recognition via fluctuation analysis of network topology and word intermittency. *Journal of Statistical Mechanics: Theory and Experiment* 2015(3):P03005. https://doi.org/10.1088/1742-5468/2015/03/P03005.

Diego R. Amancio. 2015b. A complex network approach to stylometry. *PloS one* 10(8):e0136076. https://doi.org/10.1371/journal.pone.0136076.

Diego R. Amancio. 2015c. Probing the topological properties of complex networks modeling short written texts. *PloS one* 10(2):1–17. https://doi.org/10.1371/journal.pone.0118394.

Diego R. Amancio, Eduardo G. Altmann, Diego Rybski, Osvaldo N. Oliveira Jr., and Luciano da F. Costa. 2013. Probing the statistical properties of unknown texts: Application to the voynich manuscript. *PLOS ONE* 8(7):1–10. https://doi.org/10.1371/journal.pone.0067310.

Diego R. Amancio, Maria G. V. Nunes, Osvaldo N. Oliveira Jr., and Luciano F. Costa. 2012a. Extractive summarization using complex networks and syntactic dependency. *Physica A: Statistical Me-*

chanics and its Applications 391(4):1855–1864. https://doi.org/10.1016/j.physa.2011.10.015.

Diego R. Amancio, O.N. Oliveira Jr., and Luciano da F. Costa. 2012b. Unveiling the relationship between complex networks metrics and word senses. *EPL (Europhysics Letters)* 98(1):18002. https://doi.org/10.1209/0295-5075/98/18002.

Diego R. Amancio, Osvaldo N. Oliveira Jr., and Luciano F. Costa. 2012c. Identification of literary movements using complex networks to represent texts. *New Journal of Physics* 14(4):043029. https://doi.org/10.1088/1367-2630/14/4/043029.

Lucas Antiqueira, Osvaldo N. Oliveira Jr., Luciano da Fontoura Costa, and Maria das Graças Volpe Nunes. 2009. A complex network approach to text summarization. *Information Sciences* 179(5):584 – 599.

Kathryn A. Bayles and Cheryl K. Tomoeda. 1991. *ABCD: Arizona Battery for Communication Disorders of Dementia*. Tucson, AZ: Canyonlands Publishing.

James T. Becker, François Boiler, Oscar L. Lopez, Judith Saxton, and Karen L. McGonigle. 1994. The natural history of alzheimer's disease: description of study cohort and accuracy of diagnosis. *Archives of Neurology* 51(6):585–594. https://doi.org/10.1001/archneur.1994.00540180063015.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *journal of machine learning research* 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Sasha Bozeat, Matthew A. Ralph, Karalyn Patterson, Peter Garrard, and John R. Hodges. 2000. Non-verbal semantic impairment in semantic dementia. *Neuropsychologia* 38(9):1207–1215. https://doi.org/10.1016/S0028-3932(00)00034-8.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *International Conference on World Wide Web*. Elsevier, pages 107–117.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Jin Cong and Haitao Liu. 2014. Approaching human language with complex networks. *Physics of Life Reviews* 11(4):598 – 618. https://doi.org/10.1016/j.plrev.2014.04.004.

Andre L. da Cunha, Lucilene B. de Sousa, Letícia L. Mansur, and Sandra M. Aluísio. 2015. Automatic proposition extraction from dependency trees: Helping early prediction of alzheimer's disease from narratives. In *Proceedings of the 28th International Symposium on Computer-Based Medical Systems*. Institute of Electrical and Electronics Engineers, pages 127–130. https://doi.org/10.1109/CBMS.2015.19.

Weissenbacher Davy, Johnson A. Travis, Wojtulewicz Laura, Dueck Amylou, Locke Dona, Caselli Richard, and Gonzalez Graciela. 2016. Towards automatic detection of abnormal cognitive decline and dementia through linguistic analysis of writing samples. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1198–1207. https://doi.org/10.18653/v1/N16-1143.

Henrique F. de Arruda, Luciano F. Costa, and Diego R. Amancio. 2016. Using complex networks for text classification: Discriminating informative and imaginative documents. *EPL (Europhysics Letters)* 113(2):28007. https://doi.org/10.1209/0295-5075/113/28007.

Martin Dyrba, Frederik Barkhof, Andreas Fellgiebel, Massimo Filippi, Lucrezia Hausner, Karlheinz Hauenstein, Thomas Kirste, and Stefan J. Teipel. 2015. Predicting prodromal alzheimer's disease in subjects with mild cognitive impairment using machine learning classification of multimodal multicenter diffusion-tensor and magnetic resonance imaging data. *Journal of Neuroimaging* 25(5):738–747. https://doi.org/10.1111/jon.12214.

Kathleen C. Fraser, Jed A. Meltzer, Naida L. Graham, Carol Leonard, Graeme Hirst, Sandra E. Black, and Elizabeth Rochon. 2014. Automated classification of primary progressive aphasia subtypes from narrative speech transcripts. *Cortex* 55:43–60. https://doi.org/10.1016/j.cortex.2012.12.006.

Kathleen C. Fraser, Jed A. Meltzer, and Frank Rudzicz. 2015. Linguistic features identify alzheimer's disease in narrative speech. *Journal of Alzheimer's Disease* 49(2):407–422. https://doi.org/10.3233/JAD-150520.

Peter Garrard, Vassiliki Rentoumi, Benno Gesierich, Bruce Miller, and Maria L. Gorno-Tempini. 2014. Machine learning approaches to diagnosis and laterality effects in semantic dementia discourse. *Cortex* 55:122–129. https://doi.org/10.1016/j.cortex.2013.05.008.

Harold Goodglass, Edith Kaplan, and Barbara Barresi. 2001. *The Assessment of Aphasia and Related Disorders*. The Assessment of Aphasia and Related Disorders. Lippincott Williams & Wilkins.

Arthur C. Graesser, Danielle S. McNamara, Max M. Louwerse, and Zhiqiang Cai. 2004.

Coh-metrix: Analysis of text on cohesion and language. *Behavior research methods, instruments, & computers* 36(2):193–202. https://doi.org/10.3758/BF03195564.

Ramon F. i Cancho, Ricard V. Solé, and Reinhard Köhler. 2004. Patterns in syntactic dependency networks. *Physical Review E* 69(5):051915. https://doi.org/10.1103/PhysRevE.69.051915.

Ramon F. i Cancho and Richard V. Solé. 2001. The small world of human language. *Proceedings of the Royal Society of London B: Biological Sciences* 268(1482):2261–2265. https://doi.org/10.1098/rspb.2001.1800.

William L. Jarrold, Bart Peintner, David Wilkins, Dimitra Vergryi, Colleen Richey, Maria L. Gorno-Tempini, and Jennifer Ogar. 2014. Aided diagnosis of dementia type through computer-based analysis of spontaneous speech. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics Workshop on Computational Linguistics and Clinical Psychology*. Association for Computational Linguistics, pages 27–36.

William L. Jarrold, Bart Peintner, Eric Yeh, Ruth Krasnow, Harold S. Javitz, and Gary E. Swan. 2010. Language analytics for assessing brain health: Cognitive impairment, depression and pre-symptomatic alzheimer's disease. In Yiyu Yao, Ron Sun, Tomaso Poggio, Jiming Liu, Ning Zhong, and Jimmy Huang, editors, *Proceedings of International Conference on Brain Informatics (BI 2010)*, Springer Berlin Heidelberg, pages 299–307. https://doi.org/10.1007/978-3-642-15314-3_28.

Edith Kaplan, Harold Googlass, and Sandra Weintrab. 2001. *Boston naming test*. Lippincott Williams & Wilkins.

A. Kertesz. 1982. *Western Aphasia Battery test manual*. Grune & Stratton.

Maider Lehr, Emily T. Prud'hommeaux, Izhak Shafran, and Brian Roark. 2012. Fully automated neuropsychological assessment for detecting mild cognitive impairment. In *Proceedings of the 13th Annual Conference of the International Speech Communication Association*. pages 1039–1042.

Jeaneth Machicao, Edilson A. Corrêa Jr, Gisele H. B. Miranda, Diego R. Amancio, and Odemir M. Bruno. 2016. Authorship attribution based on life-like network automata. *arXiv preprint arXiv:1610.06498* .

Brian MacWhinney. 2000. *The CHILDES Project: Tools for analyzing talk*. Lawrence Erlbaum Associates, 3 edition.

Rada Mihalcea and Dragomir Radev. 2011. *Graph-based natural language processing and information retrieval*. Cambridge University Press.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. pages 3111–3119.

Weerasak Muangpaisan, Chonachan Petcharat, and Varalak Srinonprasert. 2012. Prevalence of potentially reversible conditions in dementia and mild cognitive impairment in a geriatric clinic. *Geriatrics & gerontology international* 12(1):59–64. https://doi.org/10.1111/j.1447-0594.2011.00728.x.

Sylvester O. Orimaye, Jojo Wong, and K. Jennifer Golden. 2014. Learning predictive linguistic features for alzheimer's disease and related dementias using verbal utterances. In *Proceedings of the 1st Workshop on Computational Linguistics and Clinical Psychology (CLPsych)*. Association for Computational Linguistics, pages 78–87. www.aclweb.org/anthology/W/W14/W14-3210.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Bryan Perozzi, Rami Al-Rfou, Vivek Kulkarni, and Steven Skiena. 2014. Inducing language networks from continuous space word representations. In *Proceedings of the 5th Workshop on Complex Networks CompleNet 2014*, Springer, pages 261–273. https://doi.org/10.1007/978-3-319-05401-8_25.

Ronald C. Petersen. 2004. Mild cognitive impairment as a diagnostic entity. *Journal of internal medicine* 256(3):183–194. https://doi.org/10.1111/j.1365-2796.2004.01388.x.

Emily T. Prud'hommeaux and Brian Roark. 2011. Alignment of spoken narratives for automated neuropsychological assessment. In *Proceedings of Workshop on Automatic Speech Recognition & Understanding,ASRU*. Institute of Electrical and Electronics Engineers, pages 484–489. https://doi.org/10.1109/ASRU.2011.6163979.

Brian Roark, Margaret Mitchell, John-Paul Hosom, Kristy Hollingshead, and Jeffrey Kaye. 2011. Spoken language derived measures for detecting mild cognitive impairment. *Transactions on Audio, Speech, and Language Processing, Institute of Electrical and Electronics Engineers* 19(7):2081–2090. https://doi.org/10.1109/TASL.2011.2112351.

Ranzivelle M. Roxas and Giovanni Tapang. 2010. Prose and poetry classification and boundary detec-

tion using word adjacency network analysis. *International Journal of Modern Physics C* 21(04):503–512. https://doi.org/10.1142/S0129183110015257.

Eleanor M. Saffran, Rita S. Berndt, and Myrna F. Schwartz. 1989. The quantitative analysis of agrammatic production: Procedure and data. *Brain and language* 37(3):440–479. https://doi.org/10.1016/0093-934X(89)90030-8.

Thiago C. Silva and Diego R. Amancio. 2012. Word sense disambiguation via high order of learning in complex networks. *EPL (Europhysics Letters)* 98(5):58001.

Camila V. Teixeira, Lilian T. Gobbi, Danilla I. Corazza, Florindo Stella, José L. Costa, and Sebastião Gobbi. 2012. Non-pharmacological interventions on cognitive functions in older people with mild cognitive impairment (mci). *Archives of gerontology and geriatrics* 54(1):175–180. https://doi.org/10.1016/j.archger.2011.02.014.

László Tóth, Gábor Gosztolya, Veronika Vincze, Ildikó Hoffmann, and Gréta Szatlóczki. 2015. Automatic detection of mild cognitive impairment from spontaneous speech using asr. In *Proceedings of the 16th Annual Conference of the International Speech Communication Association*. International Speech and Communication Association, pages 2694–2698.

Marcos V. Treviso, Christopher Shulby, and Sandra M. Aluísio. 2017. Sentence segmentation in narrative transcripts from neuropsycological tests using recurrent convolutional neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1–10. https://arxiv.org/abs/1610.00211.

Veronika Vincze, Gábor Gosztolya, László Tóth, Ildikó Hoffmann, and Gréta Szatlóczki. 2016. Detecting mild cognitive impairment by exploiting linguistic information from transcripts. In *Proceedings of the 54th Annual Meeting of the Association Computer Linguistics*. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-2030.

Alyssa Weakley, Jennifer A. Williams, Maureen Schmitter-Edgecombe, and Diane J. Cook. 2015. Neuropsychological test selection for cognitive impairment classification: A machine learning approach. *Journal of clinical and experimental neuropsychology* 37(9):899–916. https://doi.org/10.1080/13803395.2015.1067290.

David Wechsler et al. 1997. *Wechsler memory scale (WMS-III)*. Psychological Corporation.

Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. Chapman & Hall/CRC, 1st edition.

A Supplementary Material

Figure 3 is Cookie Theft picture, which was used in DementiaBank project.

Figure 4 is a sequence of pictures from the Cinderella story, which were used to elicit speech narratives.

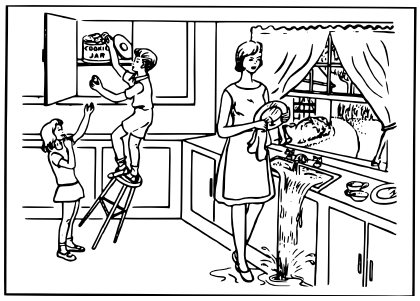


Figure 3: The Cookie Theft Picture, taken from the Boston Diagnostic Aphasia Examination (Goodglass et al., 2001).

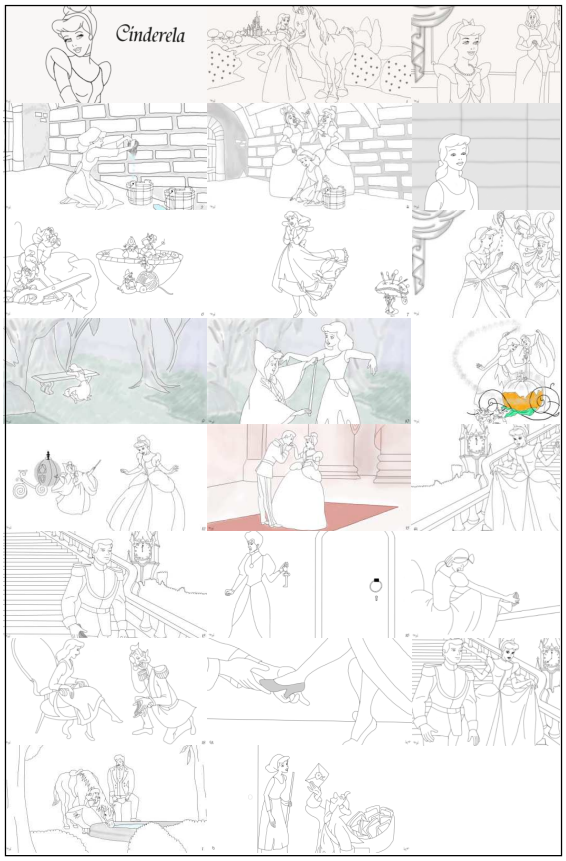


Figure 4: Sequence of Pictures of the of Cinderella story.

A.1 Examples of transcriptions

Below follows an example of a transcript of the Cookie Theft dataset.

You just want me to start talking ? Well the little girl is asking her brother we 'll say for a cookie . Now he 's getting the cookie one for him and one for her . He unbalances the step the little stool and he 's about to fall . And the lid 's off the cookie jar . And the mother is drying the dishes abstractly so she 's left the water running in the sink and it is spilling onto the floor . And there are two there 's look like two cups and a plate on the sink and board . And that boy 's wearing shorts and the little girl is in a short skirt . And the mother has an apron on . And she 's standing at the window . The window 's opened . It must be summer or spring . And the curtains are pulled back . And they have a nice walk around their house . And there 's this nice shrubbery it appears and grass . And there 's a big picture window in the background that has the drapes pulled off . There 's a not pulled off but pulled aside . And there 's a tree in the background . And the house with the kitchen has a lot of cupboard space under the sink board and under the cabinet from which the cookie you know cookies are being removed .

Below follows an excerpt of a transcript of the Cinderella dataset.

Original transcript in Portuguese:

ela morava com a madrasta as irmã né e ela era diferenciada das três era maltratada ela tinha que fazer limpeza na casa toda no castelo alias e as irmãs não faziam nada até que um dia chegou um convite do rei ele ia fazer um baile e a madrasta então é colocou que todas as filhas elas iam menos a cinderela bom como ela não tinha o vestido sapato as coisas tudo então ela mesmo teve que fazer a roupa dela começou a fazer ...

Translation of the transcript in English:

she lived with the stepmother the sister right and she was differentiated from the three was mistreated she had to do the cleaning in the entire house actually in the castle and the sisters didn't do anything until one day the king's invitation arrived he would invite everyone to a ball and then the stepmother is said that all the daughters they would go except for cinderella well since she didn't have a dress shoes all the things she had to make her own clothes she started to make them ...

A.2 Coh-Metrix-Dementia metrics

1. **Ambiguity**: verb ambiguity, noun ambiguity, adjective ambiguity, adverb ambiguity;
2. **Anaphoras**: adjacent anaphoric references, anaphoric references;
3. **Basic Counts**: Flesch index, number of word, number of sentences, number of paragraphs, words per sentence, sentences per paragraph, syllables per content word, verb incidence, noun incidence, adjective incidence, adverb incidence, pronoun incidence, content word incidence, function word incidence;
4. **Connectives**: connectives incidence, additive positive connectives incidence, additive negative connectives incidence, temporal positive connectives incidence, temporal negative connectives incidence, casual positive connectives incidence, casual negative connectives incidence, logical positive connectives incidence, logical negative connectives incidence;
5. **Co-reference Measures**: adjacent argument overlap, argument overlap, adjacent stem overlap, stem overlap, adjacent content word overlap;
6. **Content Word Frequencies**: Content words frequency, minimum among content words frequency;
7. **Hypernyms**: Mean hypernyms per verb;
8. **Logic Operators**: Logic operators incidence, and incidence, or incidence, if incidence, negation incidence;
9. **Latent Semantic Analysis (LSA)**: Average and standard deviation similarity between pairs of adjacent sentences in the text, Average and standard deviation similarity between all sentence pairs in the text, Average and standard deviation similarity between pairs of adjacent paragraphs in the text, Givenness average and standard deviation of each sentence in the text;
10. **Semantic Density**: content density;
11. **Syntactical Complexity**: only cross entropy;
12. **Tokens**: personal pronouns incidence, type-token ratio, Brunet index, Honoré Statistics.

Adversarial Adaptation of Synthetic or Stale Data

Young-Bum Kim[†] **Karl Stratos**[‡] **Dongchan Kim**[†]

[†]Microsoft AI and Research
[‡]Bloomberg L. P.
{ybkim, dongchan.kim}@microsoft.com
me@karlstratos.com

Abstract

Two types of data shift common in practice are 1. transferring from synthetic data to live user data (a deployment shift), and 2. transferring from stale data to current data (a temporal shift). Both cause a distribution mismatch between training and evaluation, leading to a model that overfits the flawed training data and performs poorly on the test data. We propose a solution to this mismatch problem by framing it as domain adaptation, treating the flawed training dataset as a source domain and the evaluation dataset as a target domain. To this end, we use and build on several recent advances in neural domain adaptation such as adversarial training (Ganin et al., 2016) and domain separation network (Bousmalis et al., 2016), proposing a new effective adversarial training scheme. In both supervised and unsupervised adaptation scenarios, our approach yields clear improvement over strong baselines.

1 Introduction

Spoken language understanding (SLU) systems analyze various aspects of a user query by classifying its domain, intent, and semantic slots. For instance, the query `how is traffic to target in bellevue` has domain `PLACES`, intent `CHECK_ROUTE_TRAFFIC`, and slots `PLACE_NAME: target` and `ABSOLUTE_LOCATION: bellevue`.

We are interested in addressing two types of data shift common in SLU applications. The first data shift problem happens when we transfer from synthetic data to live user data (a deployment shift). This is also known as the "cold-start" problem; a model cannot be trained on the real usage data prior to deployment simply because it does not exist. A common practice is to generate a large quantity of synthetic training data that mimics the expected user behavior. Such synthetic data is crafted using domain-specific knowledge and can be time-consuming. It is also flawed in that it typically does not match the live user data generated by actual users; the real queries submitted to these systems are different from what the model designers expect to see.

The second data shift problem happens when we transfer from stale data to current data (a temporal shift). In our use case, we have one set of training data from 2013 and wish to handle data from 2014–2016. This is problematic since the content of the user queries changes over time (e.g., new restaurant or movie names may be added). Consequently, the model performance degrades over time.

Both shifts cause a distribution mismatch between training and evaluation, leading to a model that overfits the flawed training data and performs poorly on the test data. We propose a solution to this mismatch problem by framing it as domain adaptation, treating the flawed training dataset as a source domain and the evaluation dataset as a target domain. To this end, we use and build on several recent advances in neural domain adaptation such as adversarial training (Ganin et al., 2016) and domain separation network (Bousmalis et al., 2016), proposing a new adversarial training scheme based on randomized predictions.

We consider both supervised and unsupervised adaptation scenarios (i.e., absence/presence of labeled data in the target domain). We find that unsupervised DA can greatly improve performance without requiring additional annotation. Super-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1297–1307
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1119

vised DA with a small amount of labeled data gives further improvement on top of unsupervised DA. In experiments, we show clear gains in both deployment and temporal shifts across 5 test domains, yielding average error reductions of 74.04% and 41.46% for intent classification and 70.33% and 32.0% for slot tagging compared to baselines without adaptation.

2 Related Work

2.1 Domain Adaptation

Our work builds on the recent success of DA in the neural network framework. Notably, Ganin et al. (2016) propose an adversarial training method for unsupervised DA. They partition the model parameters into two parts: one inducing domain-specific (or private) features and the other domain-invariant (or shared) features. The domain-invariant parameters are adversarially trained using a gradient reversal layer to be poor at domain classification; as a consequence, they produce representations that are domain agnostic. This approach is motivated by a rich literature on the theory of DA pioneered by Ben-David et al. (2007). We describe our use of adversarial training in Section 3.2.3. A special case of Ganin et al. (2016) is developed independently by Kim et al. (2016c) who motivate the method as a generalization of the feature augmentation method of Daumé III (2009).

Bousmalis et al. (2016) extend the framework of Ganin et al. (2016) by additionally encouraging the private and shared features to be mutually exclusive. This is achieved by minimizing the dot product between the two sets of parameters and simultaneously reconstructing the input (for all domains) from the features induced by these parameters.

Both Ganin et al. (2016) and Bousmalis et al. (2016) discuss applications in computer vision. Zhang et al. (2017) apply the method of Bousmalis et al. (2016) to tackle transfer learning in NLP. They focus on transfer learning between classification tasks over the same domain ("aspect transfer"). They assume a set of keywords associated with each aspect and use these keywords to inform the learner of the relevance of each sentence for that aspect.

2.2 Spoken Language Understanding

Recently, there has been much investment on the personal digital assistant (PDA) technology in industry (Sarikaya, 2015; Sarikaya et al., 2016). Apples Siri, Google Now, Microsofts Cortana, and Amazons Alexa are some examples of personal digital assistants. Spoken language understanding (SLU) is an important component of these examples that allows natural communication between the user and the agent (Tur, 2006; El-Kahky et al., 2014). PDAs support a number of scenarios including creating reminders, setting up alarms, note taking, scheduling meetings, finding and consuming entertainment (i.e. movie, music, games), finding places of interest and getting driving directions to them (Kim et al., 2016a).

Naturally, there has been an extensive line of prior studies for domain scaling problems to easily scale to a larger number of domains: pre-training (Kim et al., 2015c), transfer learning (Kim et al., 2015d), constrained decoding with a single model (Kim et al., 2016a), multi-task learning (Jaech et al., 2016), neural domain adaptation (Kim et al., 2016c), domainless adaptation (Kim et al., 2016b), a sequence-to-sequence model (Hakkani-Tür et al., 2016), domain attention (Kim et al., 2017) and zero-shot learning(Chen et al., 2016; Ferreira et al., 2015).

There are also a line of prior works on enhancing model capability and features: jointly modeling intent and slot predictions (Jeong and Lee, 2008; Xu and Sarikaya, 2013; Guo et al., 2014; Zhang and Wang, 2016; Liu and Lane, 2016a,b), modeling SLU models with web search click logs (Li et al., 2009; Kim et al., 2015a) and enhancing features, including representations (Anastasakos et al., 2014; Sarikaya et al., 2014; Celikyilmaz et al., 2016, 2010; Kim et al., 2016d) and lexicon (Liu and Sarikaya, 2014; Kim et al., 2015b).

All the above works assume that there are no any data shift issues which our work try to solve.

3 Method

3.1 BiLSTM Encoder

We use an LSTM simply as a mapping $\phi : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ that takes an input vector x and a state vector h to output a new state vector $h' = \phi(x, h)$. See Hochreiter and Schmidhuber (1997) for a detailed description.

Let $\mathcal{C}$ denote the set of character types and $\mathcal{W}$ the set of word types. Let $\oplus$ denote the vector concatenation operation. We encode an utterance using the wildly successful architecture given by bidirectional LSTMs (BiLSTMs) (Schuster and

Paliwal, 1997; Graves, 2012). The model parameters Θ associated with this BiLSTM layer are

- Character embedding $e_c \in \mathbb{R}^{25}$ for each $c \in \mathcal{C}$
- Character LSTMs $\phi^{\mathcal{C}}_f, \phi^{\mathcal{C}}_b : \mathbb{R}^{25} \times \mathbb{R}^{25} \to \mathbb{R}^{25}$
- Word embedding $e_w \in \mathbb{R}^{100}$ for each $w \in \mathcal{W}$
- Word LSTMs $\phi^{\mathcal{W}}_f, \phi^{\mathcal{W}}_b : \mathbb{R}^{150} \times \mathbb{R}^{100} \to \mathbb{R}^{100}$

Let $w_1 \dots w_n \in \mathcal{W}$ denote a word sequence where word w_i has character $w_i(j) \in \mathcal{C}$ at position j. First, the model computes a character-sensitive word representation $v_i \in \mathbb{R}^{150}$ as

$$\begin{aligned} f^{\mathcal{C}}_j &= \phi^{\mathcal{C}}_f\left(e_{w_i(j)}, f^{\mathcal{C}}_{j-1}\right) && \forall j = 1 \dots |w_i| \\ b^{\mathcal{C}}_j &= \phi^{\mathcal{C}}_b\left(e_{w_i(j)}, b^{\mathcal{C}}_{j+1}\right) && \forall j = |w_i| \dots 1 \\ v_i &= f^{\mathcal{C}}_{|w_i|} \oplus b^{\mathcal{C}}_1 \oplus e_{w_i} \end{aligned}$$

for each $i = 1 \dots n$.[1] Next, the model computes

$$\begin{aligned} f^{\mathcal{W}}_i &= \phi^{\mathcal{W}}_f\left(v_i, f^{\mathcal{W}}_{i-1}\right) && \forall i = 1 \dots n \\ b^{\mathcal{W}}_i &= \phi^{\mathcal{W}}_b\left(v_i, b^{\mathcal{W}}_{i+1}\right) && \forall i = n \dots 1 \end{aligned}$$

and induces a character- and context-sensitive word representation $h_i \in \mathbb{R}^{200}$ as

$$h_i = f^{\mathcal{W}}_i \oplus b^{\mathcal{W}}_i \tag{1}$$

for each $i = 1 \dots n$. For convenience, we write the entire operation as a mapping BiLSTM_Θ:

$$(h_1 \dots h_n) \leftarrow \text{BiLSTM}_\Theta(w_1 \dots w_n)$$

3.2 Unsupervised DA

In unsupervised domain adaptation, we assume labeled data for the source domain but not the target domain. Our approach closely follows the previous work on unsupervised neural domain adaptation by Ganin et al. (2016) and Bousmalis et al. (2016). We have three BiLSTM encoders described in Section 3.1:

1. Θ^{src}: induces source-specific features
2. Θ^{tgt}: induces target-specific features
3. Θ^{shd}: induces domain-invariant features

We now define a series of loss functions defined by these encoders.

[1] For simplicity, we assume some random initial state vectors such as $f^{\mathcal{C}}_0$ and $b^{\mathcal{C}}_{|w_i|+1}$ when we describe LSTMs.

3.2.1 Source Side Tagging Loss

The most obvious objective is to minimize the model's error on labeled training data for the source domain. Let $w_1 \dots w_n \in \mathcal{W}$ be an utterance in the source domain annotated with labels $y_1 \dots y_n \in \mathcal{L}$. We induce

$$\begin{aligned} (h^{\text{src}}_1 \dots h^{\text{src}}_n) &\leftarrow \text{BiLSTM}_{\Theta^{\text{src}}}(w_1 \dots w_n) \\ (h^{\text{shd}}_1 \dots h^{\text{shd}}_n) &\leftarrow \text{BiLSTM}_{\Theta^{\text{shd}}}(w_1 \dots w_n) \end{aligned}$$

Then we define the probability of tag $y \in \mathcal{L}$ for the i-th word as

$$\begin{aligned} z_i &= W^2_{\text{tag}} \tanh\left(W^1_{\text{tag}} \bar{h}_i + b^1_{\text{tag}}\right) + b^2_{\text{tag}} \\ p(y|h_i) &\propto \exp\left([z_i]_y\right) \end{aligned}$$

where $\bar{h}_i = h^{src}_i \oplus h^{\text{shd}}_i$ and $\Theta^{\text{tag}} = \{W^1_{\text{tag}}, W^2_{\text{tag}}, b^1_{\text{tag}}, b^2_{\text{tag}}\}$ denotes additional feed-foward parameters. The tagging loss is given by the negative log likelihood

$$L^{\text{tag}}\left(\Theta^{\text{src}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) = -\sum_i \log p\left(y_i|\bar{h}_i\right)$$

where we iterate over annotated words (w_i, y_i) on the source side.

3.2.2 Reconstruction Loss

Following previous works, we ground feature learning by reconstructing encoded utterances. Both Bousmalis et al. (2016) and Zhang et al. (2017) use mean squared errors for reconstruction, the former of image pixels and the latter of words in a context window. In contrast, we use an attention-based LSTM that fully re-generates the input utterance and use its log loss.

More specifically, let $w_1 \dots w_n \in \mathcal{W}$ be an utterance in domain $d \in \{\text{src}, \text{tgt}\}$. We first use the relevant encoders as before

$$\begin{aligned} (h^d_1 \dots h^d_n) &\leftarrow \text{BiLSTM}_{\Theta^d}(w_1 \dots w_n) \\ (h^{\text{shd}}_1 \dots h^{\text{shd}}_n) &\leftarrow \text{BiLSTM}_{\Theta^{\text{shd}}}(w_1 \dots w_n) \end{aligned}$$

The concatenated vectors $\bar{h}_i = h^d_i \oplus h^{\text{shd}}_i$ are fed into the standard attention-based decoder (Bahdanau et al., 2014) to define the probability of word w at each position i with state vector μ_{i-1} (where $\mu_0 = \bar{h}_n$):

$$\begin{aligned} \alpha_j &\propto \exp\left(\mu^\top_{i-1} \bar{h}_j\right) \quad \forall j \in \{1 \dots n\} \\ \tilde{h}_i &= \sum_{j=1}^{n} \alpha_j \bar{h}_j \\ \mu_i &= \phi^{\mathcal{R}}(\mu_{i-1} \oplus \tilde{h}_i, \mu_{i-1}) \\ p(w|\mu_i) &\propto \exp\left([W^1_{\text{rec}} \mu_i + b^1_{\text{rec}}]_w\right) \end{aligned}$$

where $\Theta^{\text{rec}} = \{\phi^{\mathcal{R}}, W^1_{\text{rec}}, b^1_{\text{rec}}\}$ denotes additional parameters. The reconstruction loss is given by the negative log likelihood

$$L^{\text{rec}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{rec}}\right) = -\sum_i \log p\left(w_i|\mu_i\right)$$

where we iterate over words w_i in both the source and target utterances.

3.2.3 Adversarial Domain Classification Loss

Ganin et al. (2016) propose introducing an adversarial loss to make shared features domain-invariant. This is motivated by a theoretical result of Ben-David et al. (2007) who show that the generalization error on the target domain depends on how "different" the source and the target domains are. This difference is approximately measured by

$$2\left(1 - 2\inf_{\Theta} \text{error}(\Theta)\right) \tag{2}$$

where $\text{error}(\Theta)$ is the domain classification error using model Θ. It is assumed that the source and target domains are balanced so that $\inf_\Theta \text{error}(\Theta) \leq 1/2$ and the difference lies in $[0, 2]$. In other words, we want to make $\text{error}(\Theta)$ as large as possible in order to generalize well to the target domain. The intuition is that the more domain-invariant our features are, the easier it is to benefit from the source side training when testing on the target side. It can also be motivated as a regularization term (Ganin et al., 2016).

Let $w_1 \dots w_n \in \mathcal{W}$ be an utterance in domain $d \in \{\text{src}, \text{tgt}\}$. We first use the shared encoder

$$(h^{\text{shd}}_1 \dots h^{\text{shd}}_n) \leftarrow \text{BiLSTM}_{\Theta^{\text{shd}}}(w_1 \dots w_n)$$

It is important that we only use the shared encoder for this loss. Then we define the probability of domain d for the utterance as

$$z_i = W^2_{\text{adv}} \tanh\left(W^1_{\text{adv}} \sum_{i=1}^{n} h^{\text{shd}}_i + b^1_{\text{adv}}\right) + b^2_{\text{adv}}$$

$$p(d|h_i) \propto \exp\left([z_i]_d\right)$$

where $\Theta^{\text{adv}} = \{W^1_{\text{adv}}, W^2_{\text{adv}}, b^1_{\text{adv}}, b^2_{\text{adv}}\}$ denotes additional feedfoward parameters. The adversarial domain classification loss is given by the *positive* log likelihood

$$L^{\text{adv}}\left(\Theta^{\text{shd}}, \Theta^{\text{adv}}\right) = \sum_i \log p\left(d^{(i)}|w^{(i)}\right)$$

where we iterate over domain-annotated utterances $(w^{(i)}, d^{(i)})$.

Random prediction training While past work only consider using a negative gradient (Ganin et al., 2016; Bousmalis et al., 2016) or positive log likelihood (Zhang et al., 2017) to perform adversarial training, it is unclear whether these approaches are optimal for the purpose of "confusing" the domain predictor. For instance, minimizing log likelihood can lead to a model accurately predicting the *opposite* domain, compromising the goal of inducing domain-invariant representations. Thus we propose to instead optimize the shared parameters for *random domain predictions*. Specifically, the above loss is replaced with

$$L^{\text{adv}}\left(\Theta^{\text{shd}}, \Theta^{\text{adv}}\right) = -\sum_i \log p\left(d^{(i)}|w^{(i)}\right)$$

where $d^{(i)}$ is set to be `src` with probability 0.5 and `tgt` with probability 0.5. By optimizing for random predictions, we achieve the desired effect: the shared parameters are trained to induce features that cannot discriminate between the source and the target domains.

3.2.4 Non-Adversarial Domain Classification Loss

In addition to the adversarial loss for domain-invariant parameters, we also introduce a non-adversarial loss for domain-specific parameters. Given $w_1 \dots w_n \in \mathcal{W}$ in domain $d \in \{\text{src}, \text{tgt}\}$, we use the private encoder

$$(h^{\text{d}}_1 \dots h^{\text{d}}_n) \leftarrow \text{BiLSTM}_{\Theta^{\text{d}}}(w_1 \dots w_n)$$

It is important that we only use the private encoder for this loss. Then we define the probability of domain d for the utterance as

$$z_i = W^2_{\text{nadv}} \tanh\left(W^1_{\text{nadv}} \sum_{i=1}^{n} h^{\text{d}}_i + b^1_{\text{nadv}}\right) + b^2_{\text{nadv}}$$

$$p(d|h_i) \propto \exp\left([z_i]_d\right)$$

where $\Theta^{\text{nadv}} = \{W^1_{\text{nadv}}, W^2_{\text{nadv}}, b^1_{\text{nadv}}, b^2_{\text{nadv}}\}$ denotes additional feedfoward parameters. The non-adversarial domain classification loss is given by the negative log likelihood

$$L^{\text{nadv}}\left(\Theta^{\text{d}}, \Theta^{\text{nadv}}\right) = \sum_i \log p\left(d^{(i)}|w^{(i)}\right)$$

where we iterate over domain-annotated utterances $(w^{(i)}, d^{(i)})$.

3.2.5 Orthogonality Loss

Finally, following Bousmalis et al. (2016), we further encourage the domain-specific features to be mutually exclusive with the shared features by imposing soft orthogonality constraints. This is achieved as follows. Given an utterance $w_1 \dots w_n \in \mathcal{W}$ in domain $d \in \{\text{src}, \text{tgt}\}$. We compute

$$(h_1^d \dots h_n^d) \leftarrow \text{BiLSTM}_{\Theta^d}(w_1 \dots w_n)$$
$$(h_1^{\text{shd}} \dots h_n^{\text{shd}}) \leftarrow \text{BiLSTM}_{\Theta^{\text{shd}}}(w_1 \dots w_n)$$

The orthogonality loss for this utterance is given by

$$L^{\text{orth}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}\right) = \sum_i (h_i^d)^\top h_i^{\text{shd}}$$

where we iterate over words i in both the source and target utterances.

3.2.6 Joint Objective

For unsupervised DA, we optimize

$$\begin{aligned} L^{\text{unsup}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}, \Theta^{\text{rec}}, \Theta^{\text{adv}}\right) = \\ L^{\text{tag}}\left(\Theta^{\text{src}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) + \\ L^{\text{rec}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{rec}}\right) + \\ L^{\text{adv}}\left(\Theta^{\text{shd}}, \Theta^{\text{adv}}\right) + \\ L^{\text{nadv}}\left(\Theta^{\text{src}}, \Theta^{\text{nadv}}\right) + \\ L^{\text{nadv}}\left(\Theta^{\text{tgt}}, \Theta^{\text{nadv}}\right) + \\ L^{\text{orth}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}\right) \end{aligned}$$

with respect to all model parameters. In an online setting, given an utterance we compute its reconstruction, adversarial, orthogonality, and tagging loss if in the source domain, and take a gradient step on the sum of these losses.

3.3 Supervised DA

In supervised domain adaptation, we assume labeled data for both the source domain and the target domain. We can easily incorporate supervision in the target domain by adding $L^{\text{tag}}\left(\Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right)$ to the unsupervised DA objective:

$$\begin{aligned} L^{\text{sup}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}, \Theta^{\text{rec}}, \Theta^{\text{adv}}\right) = \\ L^{\text{unsup}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}, \Theta^{\text{rec}}, \Theta^{\text{adv}}\right) + \\ L^{\text{tag}}\left(\Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) \end{aligned} \quad (3)$$

We mention that the approach by Kim et al. (2016c) is a special case of this objective; they optimize

$$\begin{aligned} L^{\text{sup2}}\left(\Theta^{\text{src}}, \Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) = L^{\text{tag}}\left(\Theta^{\text{src}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) + \\ L^{\text{tag}}\left(\Theta^{\text{tgt}}, \Theta^{\text{shd}}, \Theta^{\text{tag}}\right) \end{aligned} \quad (4)$$

which is motivated as a neural extension of the feature augmentation method of Daumé III (2009).

4 Experiments

In this section, we conducted a series of experiments to evaluate the proposed techniques on datasets obtained from real usage.

4.1 Test Domains and Tasks

We test our approach on a suite of 5 Microsoft Cortana domains with 2 separate tasks in spoken language understanding: (1) intent classification and (2) slot (label) tagging. The intent classification task is a multi-class classification problem with the goal of determining to which one of the n intents a user utterance belongs conditioning on the given domain. The slot tagging task is a sequence labeling problem with the goal of identifying entities and chunking of useful information snippets in a user utterance. For example, a user could say `reserve a table at joeys grill for thursday at seven pm for five people`. Then the goal of the first task would be to classify this utterance as `MAKE_RESERVATION` intent given the domain `PLACES`, and the goal of the second task would be to tag `joeys grill` as `RESTAURANT`, `thursday` as `DATE`, `seven pm` as `TIEM`, and `five` as `NUMBER_PEOPLE`.

Table 1 gives a summary of the 5 test domains. We note that the domains have various levels of label granularity.

Domain	Intent	Slot	Description
calendar	23	43	Set appointments in calendar
comm.	38	45	Make calls & send messages
places	35	64	Find locations & directions
reminder	14	35	Remind tasks in a to-do list
weather	13	19	Get weather information

Table 1: The number of intents, the number of slots and a short description of the test domains.

4.2 Experimental Setup

We consider 2 possible domain adaptation (DA) scenarios: (1) adaptation of an engineered dataset to a live user dataset and (2) adaptation of an old

dataset to a new dataset. For the first DA scenario, we test whether our approach can effectively make a system adapt from experimental, engineered data to real-world, live data. We use synthetic data which domain experts manually create based on a given domain schema[2] before the system goes live as the engineered data. We use transcribed dataset from users' speech input as the live user data. For the second scenario, we test whether our approach can effectively make a system adapt over time. A large number of users will quickly generate a large amount of data, and the usage pattern could also change. We use annotation data over 1 month in 2013 (more precisely August of 2013) as our old dataset, and use the whole data between 2014 and 2016 as our new dataset regardless of whether the data type is engineered or live user.

As we describe in the earlier sections, we consider both supervised and unsupervised DA. We apply our DA approach with labeled data in the target domain for the supervised setting and with unlabeled data for the unsupervised one. We give details of the baselines and variants of our approach below.

Unsupervised DA baselines and variants:

- *SRC*: a single LSTM model trained on a source domain without DA techniques
- DA^W: an unsupervised DA model with a word-level decoder (i.e., re-generate each word independently)
- DA^S: an unsupervised DA model with a sentence-level decoder described in Section 3.2

Supervised DA baselines and variants:

- *SRC*: a single LSTM model trained only on a source domain
- *TGT*: a single LSTM model trained only on a target domain
- *Union*: a single LSTM model trained on the union of source and target domains.
- DA: a supervised DA model described in Section 3.3
- DA^A: *DA* with adversary domain training
- DA^U: *DA* with reasonably sufficient unlabeled data

In our experiments, all the models were implemented using Dynet (Neubig et al., 2017) and were trained using Stochastic Gradient Descent (SGD) with Adam (Kingma and Ba, 2015)—an adaptive learning rate algorithm. We used the initial learning rate of 4×10^{-4} and left all the other hyper parameters as suggested in Kingma and Ba (2015). Each SGD update was computed without a minibatch with Intel MKL (Math Kernel Library)[3]. We used the dropout regularization (Srivastava et al., 2014) with the keep probability of 0.4.

We encode user utterances with BiLSTMs as described in Section 3.1. We initialize word embeddings with pre-trained embeddings used by Lample et al. (2016). In the following sections, we report intent classification results in accuracy percentage and slot results in F1-score. To compute slot F1-score, we used the standard CoNLL evaluation script[4]

4.3 Results: Unsupervised DA

We first show our results in the unsupervised DA setting where we have a labeled dataset in the source domain, but only unlabeled data in the target domain. We assume that the amount of data in both datasets is sufficient. Dataset statistics are shown in Table 2.

The performance of the baselines and our model variants are shown in Table 3. The left side of the table shows the results of the DA scenario of adapting from engineered data to live user data, and the baseline which trained only on the source domain (SRC) show a poor performance, yielding on average 48.5% on the intent classification and 42.7% F1-score on the slot tagging. Using our DA approach with a word-level decoder (DA^W) shows a significant increase in performance in all 5 test domains, yielding on average 82.2% intent accuracy and 80.5% slot F1-score. The performance increases further using the DA approach with a sentence-level decoder DA^S, yielding on average 85.6% intent accuracy and 83.0% slot F1-score.

The right side of the table shows the results of the DA scenario of adapting from old to new data, and the baseline trained only on SRC also show

[2]This is a semantic template that defines a set of intents and slots for each domain according to the intended functionality of the system.

[3]https://software.intel.com/en-us/articles/intelr-mkl-and-c-template-libraries

[4]http://www.cnts.ua.ac.be/conll2000/chunking/output.html

	Engineered → Live User				Old → New			
Domain	Train	Train*	Dev	Test	Train	Train*	Dev	Test
calendar	16904	50000	1878	10k	13165	13165	1463	10k
communication	32072	50000	3564	10k	12631	12631	1403	10k
places	23410	50000	2341	10k	21901	21901	2433	10k
reminder	19328	50000	1933	10k	16245	16245	1805	10k
weather	20794	50000	2079	10k	15575	15575	1731	10k
AVG	23590	50000	2359	10k	15903	15903	1767	10k

Table 2: Data statistics for unsupervised domain adaptation; In the first row, the columns are adaptation of engineered dataset to live user dataset, and and adaptation of old dataset to new dataset. In the second row, columns are domain, size of labeled training, unlabeled training, development and test sets. * denotes unlabeled data

		Engineered → User Live			Old → New		
Task	Domain	SRC	DA^W	DA^S	SRC	DA^W	DA^S
Intent	calendar	47.5	82.0	**84.6**	50.7	85.7	**88.8**
	communication	45.8	75.3	**81.2**	49.4	83.2	**86.2**
	places	48.5	83.7	**86.3**	51.7	88.1	**91.1**
	reminder	50.7	83.9	**88.7**	53.3	88.8	**92.8**
	weather	50.3	86.3	**87.1**	53.4	89.1	**92.2**
	AVG	48.5	82.2	**85.6**	51.7	86.9	**90.2**
Slot	calendar	42.4	79.4	**81.7**	42.2	84.7	**87.9**
	communication	41.1	75.3	**79.1**	41.5	85.3	**89.1**
	places	40.2	81.6	**83.8**	44.1	85.4	**88.7**
	reminder	42.6	83.5	**85.7**	47.4	87.6	**91.2**
	weather	47.2	82.8	**84.7**	43.2	85.6	**89.5**
	AVG	42.7	80.5	**83.0**	43.7	85.7	**89.3**

Table 3: Intent classification accuracy (%) and slot tagging F1-score (%) for the unsupervised domain adaptation. The results that perform in each domain are in bold font.

	Engineered → Live User				Old → New			
Domain	Train	Train*	Dev	Test	Train	Train*	Dev	Test
calendar	16904	1000	100	10k	13165	1000	100	10k
communication	32072	1000	100	10k	12631	1000	100	10k
places	23410	1000	100	10k	21901	1000	100	10k
reminder	19328	1000	100	10k	16245	1000	100	10k
weather	20794	1000	100	10k	15575	1000	100	10k
AVG	23590	1000	100	10k	15903	1000	100	10k

Table 4: Data statistics for supervised domain adaptation

		Engineered → User Live						Old → New					
	Domain	SRC	TGT	Union	DA	DA^A	DA^U	SRC	TGT	Union	DA	DA^A	DA^U
I	calendar	47.5	69.2	48.3	80.7	80.5	**82.4**	50.7	69.2	49.9	74.4	75.4	**75.8**
	comm.	45.8	67.4	47.0	77.5	78.0	**79.7**	49.4	65.8	50.0	70.2	70.7	**71.9**
	places	48.5	71.2	48.5	82.0	82.4	**83.2**	51.7	69.6	52.2	75.8	76.4	**77.3**
	reminder	50.7	75.0	49.9	83.9	84.1	**87.3**	53.3	72.3	53.9	77.2	78.0	**78.5**
	weather	50.3	73.8	49.6	84.3	84.7	**85.6**	53.4	71.4	52.7	76.9	78.1	**79.2**
	AVG	48.5	71.3	48.7	81.7	81.9	**83.6**	51.7	69.7	51.7	74.9	75.7	**76.5**
S	calendar	42.4	64.9	43.0	76.1	76.7	**77.1**	42.2	61.8	41.6	68.0	66.9	**69.3**
	comm.	41.1	62.0	40.4	73.3	72.1	**73.8**	41.5	61.1	44.9	67.2	66.3	**68.4**
	places	40.2	61.8	39.0	72.1	72.0	**72.9**	44.1	64.6	47.7	70.1	68.7	**72.5**
	reminder	42.6	65.1	42.6	76.8	75.7	**80.0**	47.4	70.9	44.2	78.4	76.2	**78.9**
	weather	47.2	71.2	46.4	82.6	83.0	**84.4**	43.2	64.1	44.7	71.0	69.0	**70.2**
	AVG	42.7	65.0	42.3	76.2	75.9	**77.6**	43.7	64.5	44.6	71.0	69.4	**71.9**

Table 5: Intent classification accuracy (%) and slot tagging F1-score (%) for the supervised domain adaptation.

a similar poor performance, yielding on average 51.7% accuracy and 43.7% F1-score. DA^W approach shows a significant performance increase in all 5 test domains, yielding on average 86.9% intent accuracy and 85.7% slot F1-score. Similarly, the performance increases further with the DA^S with 90.2% intent accuracy and 89.3% F1-score.

Our DA approach variants yield average error reductions of 72.04% and 79.71% for intent classification and 70.33% and 80.99% for slot tagging. The results suggest that our DA approach can quickly make a model adapt from synthetic data to real-world data and from old data to new data with the additional use of only 2 to 2.5 more data from the target domain. Aside from the performance boost itself, the approach shows even more power since the new data from the target down do not need to be labeled and it only requires collecting a little more data from the target domain. We note that the model development sets were created only from the source domain for a fully unsupervised setting. But having the development set from the target domain shows even more boost in performance although not shown in the results, and labeling only the development set from the target domain is relatively less expensive than labeling the whole dataset.

4.4 Results: Supervised DA

Second, we show our results in the supervised DA setting where we have a sufficient amount of labeled data in the source domain but relatively insufficient amount of labeled data in the target domain. Having more labeled data in the target domain would most likely help with the performance, but we intentionally made the setting more disadvantageous for our DA approach to better simulate real-world scenarios where there is usually lack of resources and time to label a large amount of new data. For each personal assistant test domain, we only used 1000 training utterances to simulate scarcity of newly labeled data, and dataset statistics are shown in Table 2. Unlike the unsupervised DA scenario, here we used the development sets created from the target domain shown in Table 4.

The left side of Table 5 shows the results of the supervised DA approach of adapting from engineered data to live user data. The baseline trained only on the source (SRC) shows on average 48.5% intent accuracy and 42.7% slot F1-score. Training only on the target domain (TGT) increases the performance to 71.3% and 65.0%, but training on the union of the source and target domains (Union) again brings the performance down to 48.7% and 42.3%. As shown in the unsupervised setting, using our DA approach (DA) shows significant performance increase in all 5 test domains, yielding on average 81.7% intent accuracy and 76.2% slot tagging. The DA approach with adversary domain training (DA^A) shows similar performance compared to that of DA, and performance shows more increase when using our DA approach with sufficient unlabeled data[5] (DA^U), yielding on average 83.6% and 77.6%. For the second scenario of adapting from old to new dataset, the results show a very similar trend in performance.

The results show that our supervised DA (DA) approach also achieves a significant performance gain in all 5 test domains, yielding average error reductions of 68.18% and 51.35% for intent classification and 60.90% and 50.09% for slot tagging. The results suggest that an effective domain adaptation can be done using the supervised DA by having only a handful more data of 1k newly labeled data points. In addition, having both a small amount of newly labeled data combined with sufficient unlabeled data can help the models perform even better. The poor performance of using the union of both source and target domain data might be due to the relatively very small size of the target domain data, overwhelmed by the data in the source domain.

4.5 Results: Adversarial Domain Classification Loss

Task	Domain	Eng. → User RAND	Eng. → User ADV
Intent	calendar	**84.6**	81.1
	communication	**81.2**	77.9
	places	**86.3**	83.5
	reminder	**88.7**	85.8
	weather	**87.1**	84.2
	AVG	**85.6**	82.5
Slot	calendar	**81.7**	78.7
	communication	**79.1**	75.7
	places	**83.8**	80.6
	reminder	**85.7**	82.4
	weather	**84.7**	81.7
	AVG	**83.0**	79.8

Table 6: Intent classification accuracy (%) and slot tagging F1-score (%) for the unsupervised domain adaptation with two different adversarial classification losses – our claimed random domain predictions (RAND) and adversarial loss (ADVR) of Ganin et al. (2016) as explained in 3.2.3.

[5]This data is used for unsupervised DA experiments (Table 2).

The impact on the performance of two different adversarial classification losses are shown in Table 6. RAND represents the unsupervised DA model with sentence-level decoder (DA^S) using random prediction loss. The ADV shows the performance of same model using the adversarial loss of Ganin et al. (2016) as described in 3.2.3. Unfortunately, in the deployment shift scenario, using the adversarial loss fails to provide any improvement on intent classification accuracy and slot tagging F1 score, achieving 82.5% intent accuracy and 79.8% slot F1 score. These results align with our hypothesis that the adversarial loss using does not confuse the classifier sufficiently.

4.6 Proxy $\mathcal{A}$-distance

	Eng. → User	Old → New
Domain	$d_{\mathcal{A}}$	$d_{\mathcal{A}}$
calendar	0.58	0.43
comm.	0.54	0.44
places	0.68	0.62
reminder	0.54	0.57
weather	0.57	0.54
AVG	0.58	0.52

Table 7: Proxy $\mathcal{A}$-distance of resulting models: (1) engineered and live user dataset and (2) old and new dataset.

The results in shown in Table 7 show Proxy $\mathcal{A}$-distance(Ganin et al., 2016) to check if our adversary domain training generalize well to the target domain. The distance between two datasets is computed by

$$\hat{d}_A = 2(1 - 2\min\{\varepsilon, 1-\varepsilon\}) \quad (5)$$

where ε is a generalization error in discriminating between the source and target datasets.

The range of $\hat{d}_A$ distance is between 0 and 2.0. 0 is the best case where adversary training successfully fake shared encoder to predict domains. In other words, thanks to adversary training our model make the domain-invariant features in shared encoder in order to generalize well to the target domain.

4.7 Vocabulary distance between engineered data and live user data

The results in shown in Table 8 show the discrepancy between two datasets. We measure the degree of overlap between vocabulary V employed by the two datasets. We simply take the Jaccard coefficient between the two sets of such vocabulary:

$$d_V(s,t) = 1 - JC(V_s, V_t),$$

where V_s is the set of vocabulary in source s domain, and V_t is the corresponding set for target t domain and $JC(A,B) = \frac{|A \cap B|}{|A \cup B|}$ is the Jaccard coefficient, measuring the similarity of two sets. The distance d_V is the high it means that they are not shared with many words. Overall, the distance between old and new dataset are still far and the number of overlapped are small, but better than live user case.

	Eng. → User	Old → New
Domain	d_V	d_V
calendar	0.80	0.72
comm.	0.80	0.93
places	0.82	0.72
reminder	0.89	0.71
weather	0.72	0.73
AVG	0.80	0.76

Table 8: Distance between different datasets: (1) engineered and live user dataset and (2) old and new dataset.

5 Conclusion

In this paper, we have addressed two types of data shift common in SLU applications: 1. transferring from synthetic data to live user data (a deployment shift), and 2. transferring from stale data to current data (a temporal shift). Our method is based on domain adaptation, treating the flawed training dataset as a source domain and the evaluation dataset as a target domain. We use and build on several recent advances in neural domain adaptation such as adversarial training and domain separation network, proposing a new effective adversarial training scheme based on randomized predictions. In both supervised and unsupervised adaptation scenarios, our approach yields clear improvement over strong baselines.

References

Tasos Anastasakos, Young-Bum Kim, and Anoop Deoras. 2014. Task specific continuous word representations for mono and multi-lingual spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. 2007. Analysis of representations for domain adaptation. *Advances in neural information processing systems* 19:137.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*. pages 343–351.

Asli Celikyilmaz, Ruhi Sarikaya, Minwoo Jeong, and Anoop Deoras. 2016. An empirical investigation of word class-based features for natural language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24(6).

Asli Celikyilmaz, Silicon Valley, and Dilek Hakkani-Tur. 2010. Convolutional neural network based semantic tagging with entity embeddings. *genre* .

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815* .

Ali El-Kahky, Derek Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2014. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. IEEE, Proceedings of the ICASSP.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015. Zero-shot semantic parser for spoken language understanding. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research* 17(59):1–35.

Alex Graves. 2012. Neural networks. In *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8).

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain adaptation of recurrent neural networks for natural language understanding. *arXiv preprint arXiv:1604.00117* .

Minwoo Jeong and Gary Geunbae Lee. 2008. Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing* 16(7).

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015a. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *Proceedings of the NAACL*. Association for Computational Linguistics.

Young-Bum Kim, Alexandre Rochette, and Ruhi Sarikaya. 2016a. Natural language model reusability for scaling to different domains. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics* .

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Domain attention with an ensemble of experts. In *Annual Meeting of the Association for Computational Linguistics*.

Young-Bum Kim, Karl Stratos, Xiaohu Liu, and Ruhi Sarikaya. 2015b. Compact lexicon selection with spectral methods. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2015c. Pre-training of hidden-unit crfs. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016b. Domainless adaptation by constrained decoding on a schema lattice. *Proceedings of the 26th International Conference on Computational Linguistics (COLING)* .

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016c. Frustratingly easy neural domain adaptation. *Proceedings of the 26th International Conference on Computational Linguistics (COLING)* .

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016d. Scalable semi-supervised query classification using matrix sketching. In *The 54th Annual Meeting of the Association for Computational Linguistics*. page 8.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015d. New transfer learning techniques for disparate label sets. *ACL. Association for Computational Linguistics* .

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*. .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* .

Xiao Li, Ye-Yi Wang, and Alex Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*.

Bing Liu and Ian Lane. 2016a. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016*.

Bing Liu and Ian Lane. 2016b. Joint online spoken language understanding and language modeling with recurrent neural networks. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Los Angeles.

Xiaohu Liu and Ruhi Sarikaya. 2014. A discriminative model based entity dictionary weighting approach for spoken language understanding. In *Spoken Language Technology Workshop (SLT)*. IEEE, pages 195–199.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Ruhi Sarikaya. 2015. *The technology powering personal digital assistants*. Keynote at Interspeech, Dresden, Germany.

Ruhi Sarikaya, Asli Celikyilmaz, Anoop Deoras, and Minwoo Jeong. 2014. Shrinkage based features for slot tagging with conditional random fields. In *INTERSPEECH*.

Ruhi Sarikaya, Paul Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiuahu Liu, et al. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *IEEE Workshop on Spoken Language Technology*.

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

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1).

Gokhan Tur. 2006. Multitask learning for spoken language understanding. In *In Proceedings of the ICASSP*. Toulouse, France.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. IJCAI.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *arXiv preprint arXiv:1701.00188* .

Chat Detection in an Intelligent Assistant: Combining Task-oriented and Non-task-oriented Spoken Dialogue Systems

Satoshi Akasaki*
The University of Tokyo
akasaki@tkl.iis.u-tokyo.ac.jp

Nobuhiro Kaji
Yahoo Japan Corporation
nkaji@yahoo-corp.jp

Abstract

Recently emerged intelligent assistants on smartphones and home electronics (*e.g.*, Siri and Alexa) can be seen as novel hybrids of domain-specific task-oriented spoken dialogue systems and open-domain non-task-oriented ones. To realize such hybrid dialogue systems, this paper investigates determining whether or not a user is going to have a chat with the system. To address the lack of benchmark datasets for this task, we construct a new dataset consisting of 15,160 utterances collected from the real log data of a commercial intelligent assistant (and will release the dataset to facilitate future research activity). In addition, we investigate using tweets and Web search queries for handling open-domain user utterances, which characterize the task of chat detection. Experiments demonstrated that, while simple supervised methods are effective, the use of the tweets and search queries further improves the F_1-score from 86.21 to 87.53.

1 Introduction

1.1 Chat detection

Conventional studies on spoken dialogue systems (SDS) have investigated either domain-specific task-oriented SDS[1] (Williams and Young, 2007) or open-domain non-task-oriented SDS (*a.k.a.*, chatbots or chat-oriented SDS) (Wallace, 2009). The former offers convenience by helping users complete tasks in specific domains, while the latter offers entertainment through open-ended chatting (or smalltalk) with users. Although the functionalities offered by the two types of SDS are complementary to each other, little practical effort has been made to combine them. This unfortunately has limited the potential of SDS.

This situation is now being changed by the emergence of voice-activated intelligent assistants on smartphones and home electronics (*e.g.*, Siri[2] and Alexa[3]). These intelligent assistants typically perform various tasks (*e.g.*, Web search, weather checking, and alarm setting) while being able to have chats with users. They can be seen as a novel hybrid of multi-domain task-oriented SDS and open-domain non-task-oriented SDS.

To realize such hybrid SDS, we have to determine whether or not a user is going to have a chat with the system. For example, if a user says "*What is your hobby?*" it is considered that she is going to have a chat with the system. On the other hand, if she says "*Set an alarm at 8 o'clock*," she is probably trying to operate her smartphone. We refer to this task as *chat detection* and treat it as a binary classification problem.

Chat detection has not been explored enough in past studies. This is primarily because little attempts have been made to develop hybrids of task-oriented and non-task-oriented SDS (see Section 2 for related work). Although task-oriented and non-task-oriented SDS have long research histories, both of them do not require chat detection. Typically, users of task-oriented SDS do not have chats with the systems and users of non-task-oriented SDS always have chats with the systems.

1.2 Summary of this paper

In this work, we construct a new dataset for chat detection. As we already discussed, chat detection

*Work done during internship at Yahoo Japan Corporation.

[1] They can be classified as single-domain or multi-domain task-oriented SDS.

[2] http://www.apple.com/ios/siri

[3] https://developer.amazon.com/alexa

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1308–1319
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1120

has not been explored enough, and thus there exist no benchmark datasets available. To address this situation, we collected $15,160$ user utterances from real log data of a commercial intelligent assistant, and recruited crowd workers to annotate those utterances with whether or not the users are going to have chats with the intelligent assistant. The resulting dataset will be released to facilitate future studies.

The technical challenge in chat detection is that we have to handle open-ended utterances of intelligent assistant users. Commercial intelligent assistants have a vast amount of users and they talk about a wide variety of topics especially when chatting with the assistants. It consequently becomes labor-intensive to collect a sufficiently large amount of annotated data for training accurate chat detectors.

We develop supervised binary classifiers to perform chat detection. We address the open-ended user utterances, which characterize chat detection, by using unlabeled external resources. We specifically utilize tweets (*i.e.*, Twitter posts) and Web search queries to enhance the supervised classifiers.

Experimental results demonstrated that, while simple supervised methods are effective, the external resources are able to further improve them. The results demonstrated that the use of the external resources increases over 1 point of F_1-score (from 86.21 to 87.53).

2 Related Work

2.1 Previous studies on combining task-oriented and non-task-oriented SDS

Task-oriented and non-task-oriented SDS have long been investigated independently, and little attempts have been made to develop hybrids of the two types of SDS. As a consequence, previous studies have not investigated chat detection without only a few exceptions.[4]

Niculescu and Banchs (2015) explored using non-task-oriented SDS as a back-off mechanism for task-oriented SDS. They, however, did not propose any concrete methods of automatically determining when to switch to non-task-oriented SDS.

[4]Unfortunately, we cannot discuss little about chat detection in existing commercial intelligent assistants since most of their technical details have not been disclosed. We make the best effort to compensate for it by comparing the proposed methods with our in-house intelligent assistant in the experiment.

Lee et al. (2007) proposed an example-based dialogue manager to combine task-oriented and non-task-oriented SDS. In such a framework, however, it is difficult to flexibly utilize state-of-the-art supervised classifiers as a component.

Other studies proposed machine-learning-based frameworks for combining multi-domain task-oriented SDS and non-task-oriented SDS (Wang et al., 2014; Sarikaya, 2017). These assume that several components including a chat detector are already available, and explore integrating those components. They discuss little on how to develop each of the components. On the other hand, the focus of this work is to develop one of those components, a chat detector. Although it lies outside the scope of this paper to explore how to exploit chat detection method in a full dialogue system, the chat detection method is considered to serve, for example, as one component within those frameworks.

2.2 Intent and domain determination

Chat detection is related to, but different from, intent and domain determination that have been studied in the field of SDS (Guo et al., 2014; Xu and Sarikaya, 2014; Ravuri and Stolcke, 2015; Kim et al., 2016; Zhang and Wang, 2016).

Both intent and domain determination have been investigated in domain-specific task-oriented SDS. Intent determination aims to determine the type of information a user is seeking in single-domain task-oriented SDS. For example, in the ATIS dataset, which is collected from an airline travel information service system, the information type includes flight, city, and so on (Tur et al., 2010). On the other hand, domain determination aims to determine which domain is relevant to a given user utterance in multi-domain task-oriented SDS (Xu and Sarikaya, 2014). Note that it is possible that domain determination is followed by intent determination.

Unlike intent and domain determination, chat detection targets hybrid systems of multi-domain task-oriented SDS and open-domain non-task-oriented SDS, and aims to determine whether the non-task-oriented component is responsible to a given user utterance or not (*i.e.*, the user is going to have a chat or not). Therefore, the objective of chat detection is different from intent and domain determination.

It may be possible to see chat detection as a spe-

cific problem of domain determination (Sarikaya, 2017). We, nevertheless, discuss it as a different problem because of the uniqueness of the "chat domain." It greatly differs from ordinary domains in that it plays a role of combining the two different types of SDS that have long been studied independently, rather than combining multiple SDS of the same types. In addition, we discuss the use of external resources, especially tweets, for chat detection. This approach is unique to chat detection and is not considered effective for ordinary domain determination.

It is interesting to note that chat detection is not followed by slot-filling unlike intent and domain determination, as far as we use a popular response generator such as seq2seq model (Sutskever et al., 2014) or an information retrieval based approach (Yan et al., 2016). Although joint intent (or domain) determination and slot-filling has been widely studied to improve accuracy (Guo et al., 2014; Zhang and Wang, 2016), the same approach is not feasible in chat detection.

2.3 Intelligent assistant

Previous studies on intelligent assistants have not investigated chat detection. Their research topics are centered around those on user behaviors including the prediction of user satisfaction and engagement (Jiang et al., 2015; Kobayashi et al., 2015; Sano et al., 2016; Kiseleva et al., 2016a,b) and gamification (Otani et al., 2016). For example, Jiang et al. (2015) investigated predicting whether users are satisfied with the responses of intelligent assistants by combining diverse features including clicks and utterances. Sano et al. (2016) explored predicting whether users will keep using the intelligent assistants in the future by using long-term usage histories.

Some earlier works used the Cortana dataset as a benchmark of domain determination (Guo et al., 2014; Xu and Sarikaya, 2014; Kim et al., 2016) or proposed a development framework for Cortana (Crook et al., 2016). Those studies, however, regarded the intelligent assistant as merely one example of multi-domain task-oriented SDS and did not explore chat detection.

2.4 Non-task-oriented SDS

Non-task-oriented SDS have long been studied in the research community. While early studies adopted rule-based methods (Weizenbaum, 1966; Wallace, 2009), statistical approaches have recently gained much popularity (Ritter et al., 2011; Vinyals and Le, 2015). This research direction was pioneered by Ritter et al. (2011), who applied a phrase-based SMT model to the response generation. Later, Vinyals and Le (2015) used the seq2seq model (Sutskever et al., 2014). To date, a number follow-up studies have been made to improve on the response quality (Hasegawa et al., 2013; Shang et al., 2015; Sordoni et al., 2015; Li et al., 2016a,b; Gu et al., 2016; Yan et al., 2016). Those studies assume that users always want to have chats with systems and investigate only methods of generating appropriate responses to given utterances. Chat detection is required for integrating those response generators into intelligent assistants.

2.5 Use of conversational data

The recent explosion of conversational data on the Web, especially tweets, have triggered a variety of dialogue studies. Those typically used tweets either for training response generators (*c.f.*, Section 2.4) or for discovering dialogue acts in an unsupervised fashion (Ritter et al., 2010; Higashinaka et al., 2011). This treatment of tweets differs from that in our work.

3 Chat Detection Dataset

In this section we explain how we constructed the new benchmark dataset for chat detection. We then analyze the data to provide insights into the actual user behavior.

3.1 Construction procedure

We sampled $15,160$ unique utterances[5] (*i.e.*, automatic speech recognition results) from the real log data of a commercial intelligent assistant, Yahoo! Voice Assist.[6] The log data were collected between Jan. and Aug. 2016. In the log data, some utterances such as "Hello" appear frequently. To construct a dataset containing both high and low frequency utterances, we set frequency thresholds[7] to divide the utterances into three groups (high, middle, and low frequency) and then randomly sampled the same number of utterances

[5]The utterances are all in Japanese. Example utterances given in this paper are English translations.

[6]`https://v-assist.yahoo.co.jp`

[7]We cannot disclose the exact threshold values so as to keep the detailed statistics of the original log data confidential.

Label	Example	No. of votes
CHAT	Let's talk about something.	5
	What is your hobby?	7
	I don't have any holidays this month.	5
	I'm walking around now.	6
	Do you like cats?	5
	You are a serious geek.	7
NONCHAT	Show me a picture of Mt. Fuji.	6
	What's the highest building in the world?	5
	A nice restaurant near here.	7
	Wake me up at 9:10.	7
	Brighten the screen.	6
	Turn off the alarm.	7

Table 1: Example utterances and the numbers of votes. NONCHAT utterances are further divided into information seeking (top) and device control (bottom) to facilitate readers' understanding.

#Votes	No. of utterances
4	1701
5	2670
6	4978
7	5811

Table 2: Distribution of the numbers of votes.

from each of the three groups. During the data collection, we ensured privacy by manually removing utterances that included the full name of a person or detailed address information.

Next, we recruited crowd workers to annotate the $15,160$ utterances with two labels, CHAT and NONCHAT. The workers annotated the CHAT label when users were going to have chats with the intelligent assistant and annotated the NONCHAT label when users were seeking some information (*e.g.*, searching the Web or checking the weather) or were trying to operate the smartphones (*e.g.*, setting alarms or controlling volume). Note that our intelligent assistant works primarily on smartphones and thus the NONCHAT utterances include many operational instructions such as alarm setting. Example utterances are given in Table 1.

Seven workers were assigned to each utterance, and the final labels were obtained by majority vote to address the quality issue inherent in crowdsourcing. The last column in Table 1 shows the number of votes that the majority label obtained. For example, five workers provided the CHAT label (and the other two provided the NONCHAT label) to the first utterance "Let's talk about something."

3.2 Data analysis

The construction process described above yielded a dataset made up of $4,833$ CHAT and $10,327$ NONCHAT utterances.

We investigated the annotation agreement among the crowd workers. Table 2 shows the distribution of the numbers of votes that the majority labels obtained. The annotation given by the seven workers agreed perfectly in $5,811$ of the $15,160$ utterances (38%). Also, at least six workers agreed in the majority of cases, $10,789$ $(= 4,978 + 5,811)$ utterances (71%). This indicates high agreement among the workers and the reliability of the annotation results.

During the data construction, we found that a typical confusing case arises when the utterance can be interpreted as an implicit information request. For example, the utterance "*I am hungry*" can be seen as the user trying to have a chat with the assistant, but it might be the case that she is looking for a local restaurant. Similar examples include "*I have a backache*" and so on. One solution in this case might be to ask the user a clarification question (Schlöder and Fernandez, 2015). Such an exploration is left for our future research.

Additionally, we manually classified the CHAT utterances according to their dialogue acts to figure out how real users have chats with the intelligent assistant (Table 3). The set of dialogue acts was designed by referring to (Meguro et al., 2010). As shown in Table 3, while some of the utterances are boilerplates (*e.g.*, those in the GREETING act) and thus have limited variety, the majority of the utterances exhibit tremendous diversity. We see

Dialogue act (No. of Utter.)	Example
GREETING (206)	Hello. Merry Christmas.
SELFDISCLOSURE (1164)	I am free today. I have a sore throat.
ORDER (716)	Please cheer me up. Give me a song!
QUESTION (1551)	Do you have emotions? Are you angry?
INVITATION (130)	Let's play with me! Let's go to karaoke next time.
INFORMATION (214)	My cat is acting strange. It snowed a lot.
THANKS (126)	Thank you. You are cool!
CURSE (172)	You're an idiot. You are useless.
APOLOGY (9)	I'm sorry. I mistook, sorry.
INTERJECTION (151)	Whoof. Yeah, yeah.
MISC (394)	May the force be with you. Cock-a-doodle-doo.

Table 3: Distribution over dialogue acts and example utterances.

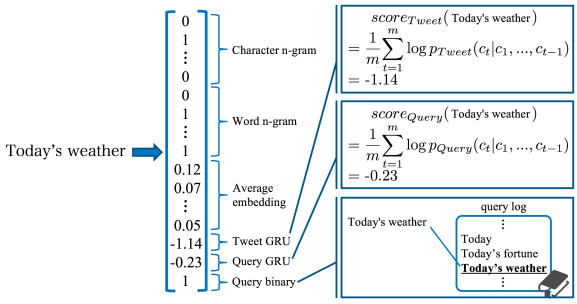


Figure 1: Feature vector representation of the example utterance "Today's weather." The upper three parts of the vector represent the features described in Section 4.1 (character n-gram, word n-gram, and average of the word embeddings). The three additional features explained in Section 4.2 are added as two real-valued features (Tweet GRU and Query GRU) and one binary feature (Query binary).

a wide variety of topics including private issues (*e.g.*, "*I am free today*") and questions to the assistant (*e.g.*, "*Are you angry?*"). Also, we even see a movie quote ("*May the force be with you*") and a rooster crow ("*Cock-a-doodle-doo*") in the MISC act. These clearly represent the open-domain nature of the user utterances in intelligent assistants.

Interestingly, some users curse at the intelligent assistant probably because it failed to make appropriate responses (see the CURSE act). Although such user behavior would not be observed from paid research participants, we observe a certain amount of curse utterances in the real data.

4 Detection Method

We formulate chat detection as a binary classification problem to train supervised classifiers. In this section, we first explain the two types of classifiers explored in this paper, and then investigate the use of external resources for enhancing those classifiers.

4.1 Base classifiers

The first classifier utilizes SVM for its popularity and efficiency. It uses character and word n-gram ($n = 1$ and 2) features. It also uses word embedding features (Turian et al., 2010). A skip-gram model (Mikolov et al., 2013) is trained on the entire intelligent assistant log[8] to learn word embeddings. The embeddings of the words in the utterance are then averaged to produce additional features.

The second classifier uses a convolutional neural network (CNN) because it has recently proven to perform well on text classification problems (Kim, 2014; Johnson and Zhang, 2015a,b). We follow (Kim, 2014) to develop a simple CNN that has a single convolution and max-pooling layer followed by the soft-max layer. We use a rectified linear unit (ReLU) as the non-linear activation function. The same word embeddings as SVM are used for the pre-training.

4.2 Using external resources

We next investigate using external resources for enhancing the base classifiers. Thanks to the rapid evolution of the Web in the past decade, a variety of textual data including not only conversational (*i.e.*, chat-like) but also non-conversational ones are abundantly available nowadays. These data offer an effective way of enhancing the base classifiers. We specifically use tweets and Web search queries as conversational and non-conversational text data, respectively.

We train character-based[9] language models on

[8] We used the same log data used in Section 3. The detailed statistics is confidential.

[9] We also trained word-based language models in prelim-

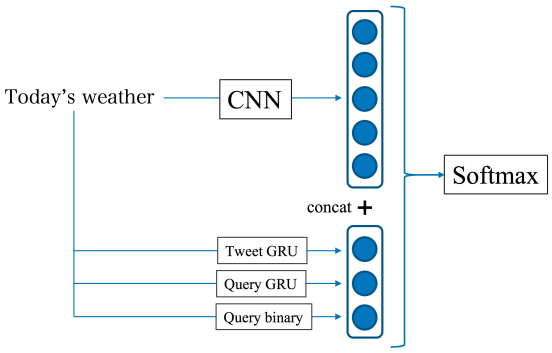


Figure 2: Architecture of our CNN-based classifier when the input utterance is "Today's weather." The output layer of CNN and the three additional features explained in Section 4.2 are concatenated. The resulting vector is fed to the soft-max function.

tweets and Web search queries, and use their scores (*i.e.*, the normalized log probabilities of the utterance) as two additional features. Let $u = c_1, c_2, \ldots, c_m$ be an utterance made up of m characters. Then, the score $score_r(u)$ of the language model trained on the external resource $r \in \{\text{tweet}, \text{query}\}$ is defined as

$$score_r(u) = \frac{1}{m} \sum_{t=1}^{m} \log p_r(c_t \mid c_1, \ldots, c_{t-1}).$$

The GRU language model is adopted for its superior performance (Cho et al., 2014; Chung et al., 2014). Let $\mathbf{x}_t$ be the embedding of t-th character and $\mathbf{h}_t$ be the t-th hidden state. GRU computes the hidden state as

$$\begin{aligned}
\mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \\
\mathbf{z}_t &= \sigma(\mathbf{W}^{(z)}\mathbf{z}_t + \mathbf{U}^{(z)}\mathbf{h}_{t-1}) \\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}^{(h)}\mathbf{x}_t + \mathbf{U}^{(h)}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \\
\mathbf{r}_t &= \sigma(\mathbf{W}^{(r)}\mathbf{x}_t + \mathbf{U}^{(r)}\mathbf{h}_{t-1})
\end{aligned}$$

where $\odot$ is the element-wise multiplication, σ is the sigmoid and $\tanh$ is the hyperbolic tangent. $\mathbf{W}^{(z)}$, $\mathbf{U}^{(z)}$, $\mathbf{W}^{(h)}$, $\mathbf{U}^{(h)}$, $\mathbf{W}^{(r)}$, and $\mathbf{U}^{(r)}$ are weight matrices. The hidden states are fed to the soft-max to predict the next word.

We also use a binary feature indicating whether the utterance appears in the Web search query log or not. We observe that some NONCHAT utterances are made up of single entities such as location and product names. Such utterances are considered to be seeking information on those entities. We therefore use the query log as an entity dictionary to derive a feature indicating whether the utterance is likely to be a single entity.

The resulting three features are incorporated into the SVM-based classifier straightforwardly (Figure 1). For the CNN-based classifier, they are provided as additional inputs to the soft-max layer (Figure 2).

5 Experimental Results

We empirically evaluate the proposed methods on the chat detection dataset.

5.1 Experimental settings

We performed 10-fold cross validation on the chat detection dataset to train and evaluate the proposed classifiers. In each fold, we used 80%, 10%, and 10% of the data for the training, development, and evaluation, respectively.

We used `word2vec`[10] to learn 300 dimensional word embeddings. They were used to induce the additional 300 features for SVM. They were also used as the pre-trained word embeddings for CNN.

We used the `faster-rnn` toolkit[11] to train the GRU language models. The size of the embedding and hidden layer was set to 256. Noise contrastive estimation (Gutmann and Hyvärinen, 2010) was used to train the soft-max function and the number of noise samples was set to 50. Maximum entropy 4-gram models were also trained to yield a combined model (Mikolov et al., 2011).

The language models were trained on 100 millions tweets collected between Apr. and July 2016 and 100 million Web search queries issued between Mar. and Jun. 2016. The tweets were sampled from those received replies to collect only conversational tweets (Ritter et al., 2011). The same Web search queries were used to derive the binary feature. Although it is difficult to release those data, we plan to make the feature values available together with the benchmark dataset.

We used `liblinear`[12] to train L_2-regularized L_2-loss SVM. The hyperparameter c was tuned

inary experiments and found that character-based ones perform consistently better.

[10] https://code.google.com/archive/p/word2vec
[11] https://github.com/yandex/faster-rnnlm
[12] https://www.csie.ntu.edu.tw/~cjlin/liblinear

Model	Acc.	P	R	F_1
Majority	68.12	N/A	N/A	N/A
Tweet GRU	72.07	54.54	74.40	62.94
In-house IA	78.31	62.57	79.51	70.03
SVM	90.51	86.42	83.45	84.91
SVM+embed.	91.35	87.62	84.88	86.21
SVM+embed.+tweet-query	**92.15**	**88.61**	**86.50**	**87.53**
CNN	85.16	83.40	68.12	74.41
CNN+pre-train.	90.84	87.03	83.80	85.36
CNN+pre-train.+tweet-query	91.48	87.78	85.18	86.56

Table 4: Chat detection results.

over $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$.

The CNN was implemented with `chainer`.[13] We tuned the number of feature maps over $\{100, 150\}$, and filter region sizes over $\{\{2\}, \{3\}, \{1,2\}, \{2,3\}, \{3,4\}, \{1,2,3\}, \{2,3,4\}\}$. The mini-batch size was set to 32. The dropout rate was set to 0.5. We used Adam ($\alpha = 0.001$, $\beta 1 = 0.9$, $\beta 2 = 0.999$, and $\epsilon = 10^{-8}$) to perform stochastic gradient descent (Kingma and Ba, 2015).

5.2 Baselines

The following baseline methods were implemented for comparison:

Majority Utterances are always classified as the majority class, NONCHAT.

Tweet GRU Utterances are classified as CHAT if the score of the GRU language model trained on the tweets exceeds a threshold. We used exactly the same GRU language model as the one that was used for deriving the feature. The threshold was calibrated on the development data by maximizing the F_1-score of the CHAT class.

In-house IA Our in-house intelligent assistant system, which adopts a hybrid of rule-based and example-based approaches. Since we cannot disclose its technical details, the result is presented just for reference.

5.3 Result

Table 4 gives the precision, recall, F_1-score (for the CHAT class), and overall classification accuracy results. We report only accuracy for **Majority** baseline. **+embed.** and **+pre-train.** represent using the word embedding features for SVM and the pre-trained word embeddings for CNN, respectively. **+tweet-query** represents using the three features derived from the tweets and Web search query.

Table 4 represents that both of the classifiers, SVM and CNN, perform accurately. We see that both **+embed.** and **+pre-train.** improve the results. The best performing method, **SVM+embed.+tweet-query**, achieves 92% accuracy and 87% F_1-score, outperforming all of the baselines. CNN performed worse than SVM contrary to results reported by recent studies (Kim, 2014). We think this is because the architecture of our CNN is rather simplistic. It might be possible to improve the CNN-based classifier by adopting more complex network, although it is likely to come at the cost of extra training time. Another reason would be that our SVM classifier uses carefully designed features beyond word 1-grams.

Table 4 also represents that the external resources are effective, improving F_1-scores almost 1 points in both SVM and CNN. Table 5 illustrates example utterances and their language model scores. We see that the language models trained on the tweets and queries successfully provide the CHAT utterances with high and low scores, respectively. Table 6 shows chat detection results when each of the three features derived from the external resources is added to **SVM+embed.** The results represent that they are all worse than **SVM+embed.+tweet-query** and thus it is crucial to combine all of them for achieving the best performance.

Table 7 shows examples of feature weights of **SVM+embed.+tweet-query**. Tweet GRU and query GRU denote the language model score features. The others are word n-gram features. We see that the language model scores have the large

[13] http://chainer.org

Score (tweet/query)		Label	Utterance
−0.964	−1.427	CHAT	Halloween has already finished.
−0.957	−1.610	CHAT	Let's sleep.
−1.233	−0.562	NONCHAT	Pokemon Go install.
−1.837	−0.682	NONCHAT	Weekly weather forecast.

Table 5: Examples of the language model scores. The first two columns represent the scores provided by the GRU language models trained on the tweets and Web search queries, respectively. The third and fourth columns represent the label and utterance.

Feature	Acc.	P	R	F_1
tweet GRU	91.53	87.62	85.49	86.53
query GRU	91.38	87.55	85.06	86.28
query binary	91.42	87.56	85.21	86.36

Table 6: Effect of the three features derived from the tweets and Web search queries.

Feature	Weight	Feature	Weight
tweet GRU	1.128	query GRU	−0.771
I	0.215	call to	−0.217
Sing	0.195	volume	−0.196

Table 7: Examples feature weights of **SVM+embed+tweet-query**.

positive and negative weights, respectively. This indicates that effectiveness of the language models. We also see that the first person has a large positive weight, while terms related to device controlling ("call to" and "volume") have large negative weights.

Table 8 represents chat detection results of **SVM+embd.+tweet-query** across the numbers of votes that the majority label obtained. As expected, we see that all metrics get higher as the number of agreement among the crowd workers becomes larger. In fact, we see as much as 98% accuracy when all seven workers agree. This implies that utterances easy for humans to classify are also easy for the classifiers.

5.4 Training data size

We next investigate the effect of the training data size on the classification accuracy.

Figure 3 illustrates the learning curve. It represents that the classification accuracy improves almost monotonically as the training data size increases. Although our training data is by no means small, the shape of the learning curve nevertheless suggests that further improvement would be achieved by adding more training data. This implies that a very large amount of training data are required for covering open-domain utterances in intelligent assistants.

#Votes	#Utter.	Acc.	P	R	F_1
4	1701	66.67	55.41	59.81	57.53
5	2670	87.72	80.46	83.01	81.72
6	4978	96.02	92.73	93.87	93.30
7	5811	98.33	96.73	97.68	97.20

Table 8: Chat detection results across the numbers of votes that the majority label obtained.

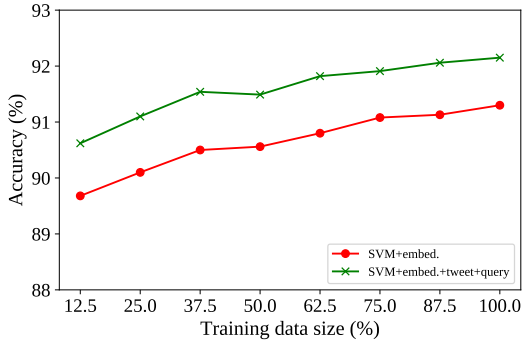


Figure 3: Learning curve of the proposed methods. The horizontal axis represents what percentage of the training portion is used in each fold of the cross validation. The vertical axis represents the classification accuracy.

The figure at the same time represents the usefulness of the external resources. We see that **SVM+embed.+tweet-query** trained on about 25% of the training data is able to achieve comparable accuracy with **SVM+embed.** trained on the entire training data. This result suggests that the external resources are able to compensate for the scarcity of annotated data.

5.5 Utterance length

We finally investigate how the utterance length correlates with the classification accuracy. Fig-

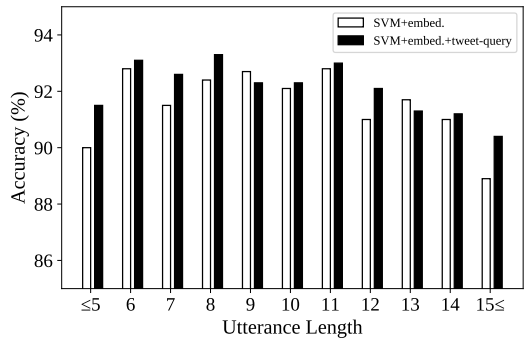


Figure 4: Classification accuracy across utterance lengths in the number of characters.

ure 4 illustrates the classification accuracies of **SVM+embed.** and **SVM+embed.+tweet-query** for each utterance length in the number of characters.

Figure 4 reveals that the difference between the two proposed methods is evident in short utterances (*i.e.*, ≤ 5). This is because those utterances are too short to contain sufficient information required for classification, and the additional features are helpful. We note that Japanese writing system uses ideograms and thus even five characters is enough to represent a simple sentence.

We also see a clear difference in longer utterances (*i.e.*, $15 \leq$) as well. We consider those long utterances are difficult to classify because some words in the utterances are irrelevant for the classification and the n-gram and embedding features include those irrelevant ones. On the other hand, we consider that the language model scores are good at capturing stylistic information irrespective of the utterance length.

6 Future Work

As discussed in Section 3.2, some user utterances such as "*I am hungry*" are ambiguous in nature and thus are difficult to handle in the current framework. An important future work is to develop a sophisticated dialogue manager to handle such utterances, for example, by making clarification questions (Schlöder and Fernandez, 2015).

We manually investigated the dialogue acts in the chat detection dataset (*c.f.*, Section 3.2). It is interesting to automatically determine the dialogue acts to help producing appropriate system responses. Some related studies exist in such a research direction (Meguro et al., 2010).

Although we used only text data to perform chat detection, we can also utilize contextual information such as the previous utterances (Xu and Sarikaya, 2014), the acoustic information (Jiang et al., 2015), and the user profile (Sano et al., 2016). It is an interesting research topic to use such contextual information beyond text. It is considered promising to make use of a neural network for integrating such heterogeneous information.

An automatic speech recognition (ASR) error is a popular problem in SDS, and previous studies have proposed sophisticated techniques, including re-ranking (Morbini et al., 2012) and POMDP (Williams and Young, 2007), for addressing the ASR errors. Incorporating these techniques into our methods is also an important future work.

Although the studies on non-task-oriented SDS have made substantial progress in the past few years, it unfortunately remains difficult for the systems to fluently chat with users (Higashinaka et al., 2015). Further efforts on improving non-task-oriented dialogue systems is an important future work.

7 Conclusion

This paper investigated chat detection for combining domain-specific task-oriented SDS and open-domain non-task-oriented SDS. To address the scarcity of benchmark datasets for this task, we constructed a new benchmark dataset from the real log data of a commercial intelligent assistant. In addition, we investigated using the external resources, tweets and Web search queries, to handle open-domain user utterances, which characterize the task of chat detection. The empirical experiment demonstrated that the off-the-shelf supervised methods augmented with the external resources perform accurately, outperforming the baseline approaches. We hope that this study contributes to remove the long-standing boundary between task-oriented and non-task-oriented SDS.

To facilitate future research, we are going to release the dataset together with the feature values derived from the tweets and Web search queries.[14]

Acknowledgments

We thank Manabu Sassano, Chikara Hashimoto, Naoki Yoshinaga, and Masashi Toyoda for fruitful discussions and comments. We also thank the anonymous reviewers.

[14] https://research-lab.yahoo.co.jp/en/software

References

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*. pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555.

Paul Crook, Alex Marin, Vipul Agarwal, Khushboo Aggarwal, Tasos Anastasakos, Ravi Bikkula, Daniel Boies, Asli Celikyilmaz, Senthilkumar Chandramohan, Zhaleh Feizollahi, Roman Holenstein, Minwoo Jeong, Omar Khan, Young-Bum Kim, Elizabeth Krawczyk, Xiaohu Liu, Danko Panic, Vasiliy Radostev, Nikhil Ramesh, Jean-Phillipe Robichaud, Alexandre Rochette, Logan Stromberg, and Ruhi Sarikaya. 2016. Task completion platform: A self-serve multi-domain goal oriented dialogue platform. In *Proceedings of NAACL (Demonstrations)*. pages 47–51. http://www.aclweb.org/anthology/N16-3010.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*. pages 1631–1640. http://www.aclweb.org/anthology/P16-1154.

Daniel (Zhaohan) Guo, Gokhan Tur, Scott Wen tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *Proceedings of IEEE SLT Workshop*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of AISTATS*. pages 297–304.

Takayuki Hasegawa, Nobuhiro Kaji, Naoki Yoshinaga, and Masashi Toyoda. 2013. Predicting and eliciting addressee's emotion in online dialogue. In *Proceedings of ACL*. pages 964–972. http://www.aclweb.org/anthology/P13-1095.

Ryuichiro Higashinaka, Kotaro Funakoshi, Masahiro Araki, Hiroshi Tsukahara, Yuka Kobayashi, and Masahiro Mizukami. 2015. Towards taxonomy of errors in chat-oriented dialogue systems. In *Proceedings of SIGDIAL*. pages 87–95. http://aclweb.org/anthology/W15-4611.

Ryuichiro Higashinaka, Noriaki Kawamae, Kugatsu Sadamitsu, Yasuhiro Minami, Toyomi Meguro, Kohji Dohsaka, and Hirohito Inagaki. 2011. Building a conversational model from two-tweets. In *Proceedings of ASRU*. pages 330–335.

Jiepu Jiang, Ahmed Hassan Awadallah, Rosie Jones, Umut Ozertem, Imed Zitouni, Ranjitha Gurunath Kulkarni, and Omar Zia Khan. 2015. Automatic online evaluation of intelligent assistants. In *Proceedings of WWW*. pages 506–516.

Rie Johnson and Tong Zhang. 2015a. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of NAACL*. pages 103–112. http://www.aclweb.org/anthology/N15-1011.

Rie Johnson and Tong Zhang. 2015b. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in NIPS*, pages 919–927.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016. Intent detection using semantically enriched word embeddings. In *Proceedings of IEEE SLT Workshop*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*. pages 1746–1751. http://www.aclweb.org/anthology/D14-1181.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Julia Kiseleva, Kyle Williams, Ahmed Hassan Awadallah, Aidan Crook, Imed Zitouni, and Tasos Anastasakos. 2016a. Predicting user satisfaction with intelligent assistants. In *Proceedings of SIGIR*. pages 45–54.

Julia Kiseleva, Kyle Williams, Ahmed Hassan Awadallah, Aidan C. Crook, Imed Zitouni, and Tasos Anastasakos. 2016b. Understanding user satisfaction with intelligent assistants. In *Proceedings of SIGCHIIR*. pages 121–130.

Hayato Kobayashi, Kaori Tanio, and Manabu Sassano. 2015. Effects of game on user engagement with spoken dialogue system. In *Proceedings of SIGDIAL*. pages 422–426. http://aclweb.org/anthology/W15-4656.

Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. 2007. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication* 51(5):466–484.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL*. pages 110–119. http://www.aclweb.org/anthology/N16-1014.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of ACL*. pages 994–1003. http://www.aclweb.org/anthology/P16-1094.

Toyomi Meguro, Ryuichiro Higashinaka, Yasuhiro Minami, and Kohji Dohsaka. 2010. Controlling listening-oriented dialogue using partially observable markov decision processes. In *Proceedings of Coling*. pages 761–769. http://www.aclweb.org/anthology/C10-1086.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky. 2011. Strategies for training large scale neural network language models. In *Proceedings of ASRU*. pages 196–201.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*. pages 3111–3119.

Fabrizio Morbini, Kartik Audhkhasi, Ron Artstein, Maarten Van Segbroeck, Kenji Sagae, Panayiotis Georgiou, David R. Traum, and Shri Narayanan. 2012. A reranking approach for recognition and classification of speech input in conversational dialogue systems. In *Proceedings of SLT*. pages 49–54.

Andreea I. Niculescu and Rafael E. Banchs. 2015. Strategies to cope with errors in human-machine speech interactions: using chatbots as back-off mechanism for task-oriented dialogues. In *Proceedings of ERRARE*.

Naoki Otani, Daisuke Kawahara, Sadao Kurohashi, Nobuhiro Kaji, and Manabu Sassano. 2016. Large-scale acquisition of commonsense knowledge via a quiz game on a dialogue system. In *Proceedings of OKBQA*. pages 11–20. http://aclweb.org/anthology/W16-4402.

Suman Ravuri and Andreas Stolcke. 2015. A comparative study of neural network models for lexical intent classification. In *In Proceedings of ASRU*. pages 368–374.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *In Proceedings of NAACL*. pages 172–180. http://www.aclweb.org/anthology/N10-1020.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of EMNLP*. pages 583–593. http://www.aclweb.org/anthology/D11-1054.

Shumpei Sano, Nobuhiro Kaji, and Manabu Sassano. 2016. Prediction of prospective user engagement with intelligent assistants. In *Proceedings of ACL*. pages 1203–1212. http://www.aclweb.org/anthology/P16-1114.

Ruhi Sarikaya. 2017. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine* 34(1):67–81.

Julian J. Schlöder and Raquel Fernandez. 2015. Clarifying intentions in dialogue: A corpus study. In *Proceedings of the 11th International Conference on Computational Semantics*. pages 46–51. http://www.aclweb.org/anthology/W15-0106.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of ACL*. pages 1577–1586. http://www.aclweb.org/anthology/P15-1152.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL*. pages 196–205. http://www.aclweb.org/anthology/N15-1020.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in NIPS*, pages 3104–3112.

Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? In *Proceedings of IEEE SLT Workshop*. pages 19–24.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*. pages 384–394. http://www.aclweb.org/anthology/P10-1040.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of Deep Learning Workshop*.

Richard S. Wallace. 2009. *The Anatomy of A.L.I.C.E.*, Springer, pages 181–210.

Zhuoran Wang, Hongliang Chen, Guanchun Wang, Hao Tian, Hua Wu, and Haifeng Wang. 2014. Policy learning for domain selection in an extensible multi-domain spoken dialogue system. In *Proceedings of EMNLP*. pages 57–67. http://www.aclweb.org/anthology/D14-1007.

Joseph Weizenbaum. 1966. Eliza–a computer program for the study of natural language communication between man and machine. *Communications of the ACM* 9(1):36–45.

Jason D. Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language* 21(2):393–422.

Puyang Xu and Ruhi Sarikaya. 2014. Contextual domain classification in spoken language understanding systems using recurrent neural network. In *Proceedings of ICASSP*. pages 136–140.

Zhao Yan, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. 2016. Docchat: An information retrieval approach for chatbot engines using unstructured documents. In *Proceedings of ACL*. pages 516–525. http://www.aclweb.org/anthology/P16-1049.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *Proceedings of IJCAI*. pages 2993–2999.

A Neural Local Coherence Model

Dat Tien Nguyen*
Informatics Institute
University of Amsterdam
t.d.nguyen@uva.nl

Shafiq Joty
Qatar Computing Research Institute
HBKU, Qatar Foundation
sjoty@hbku.edu.qa

Abstract

We propose a local coherence model based on a convolutional neural network that operates over the entity grid representation of a text. The model captures long range entity transitions along with entity-specific features without loosing generalization, thanks to the power of distributed representation. We present a pairwise ranking method to train the model in an end-to-end fashion on a task and learn task-specific high level features. Our evaluation on three different coherence assessment tasks demonstrates that our model achieves state of the art results outperforming existing models by a good margin.

1 Introduction and Motivation

What distinguishes a coherent text from a random sequence of sentences is that it binds the sentences together to express a meaning as a whole — the interpretation of a sentence usually depends on the meaning of its neighbors. *Coherence models* that can distinguish a coherent from incoherent texts have a wide range of applications in text generation, summarization, and coherence scoring.

Several formal theories of coherence have been proposed (Mann and Thompson, 1988a; Grosz et al., 1995; Asher and Lascarides, 2003), and their principles have inspired development of many existing coherence models (Barzilay and Lapata, 2008; Lin et al., 2011; Li and Hovy, 2014). Among these models, the *entity grid* (Barzilay and Lapata, 2008), which is based on Centering Theory (Grosz et al., 1995), is arguably the most popular, and has seen a number of improvements over the years. As shown in Figure 1, the entity grid model represents a text by a grid that captures how grammatical roles of different entities change from sentence to sentence. The grid is then converted into a feature vector containing probabilities of local entity transitions, which enables machine learning models to learn the degree of text coherence. Extensions of this basic grid model incorporate entity-specific features (Elsner and Charniak, 2011), multiple ranks (Feng and Hirst, 2012), and coherence relations (Feng et al., 2014).

While the entity grid and its extensions have been successful in many applications, they are limited in several ways. First, they use discrete representation for grammatical roles and features, which prevents the model from considering sufficiently long transitions (Bengio et al., 2003). Second, feature vector computation in existing models is decoupled from the target task, which limits the model's capacity to learn task-specific features.

In this paper, we propose a *neural* architecture for coherence assessment that can capture long range entity transitions along with arbitrary entity-specific features. Our model obtains generalization through *distributed representations* of entity transitions and entity features. We also present an end-to-end training method to learn task-specific high level features automatically in our model.

We evaluate our approach on three different evaluation tasks: discrimination, insertion, and summary coherence rating, proposed previously for evaluating coherence models (Barzilay and Lapata, 2008; Elsner and Charniak, 2011). Discrimination and insertion involve identifying the right order of the sentences in a text with different levels of difficulty. In the summary coherence rating task, we compare the rankings, given by the model, against human judgments of coherence.

The experimental results show that our neural models consistently improve over the non-neural counterparts (i.e., existing entity grid models) yielding absolute gains of about 4% on dis-

*Both authors contributed equally to this work.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1320–1330
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1121

crimination, up to 2.5% on insertion, and more than 4% on summary coherence rating. Furthermore, our model achieves state of the art results in all these tasks. We have released our source code for research purposes.[1]

The remainder of this paper is organized as follows. We describe entity grid, its extensions, and its limitations in Section 2. In Section 3, we present our neural model. We describe evaluation tasks and results in Sections 4 and 5. We give a brief account of related work in Section 6. Finally, we conclude with future directions in Section 7.

2 Entity Grid and Its Extensions

Motivated by Centering Theory (Grosz et al., 1995), Barzilay and Lapata (2008) proposed an entity-based model for representing and assessing text coherence. Their model represents a text by a two-dimensional array called *entity grid* that captures transitions of discourse entities across sentences. As shown in Figure 1, the rows of the grid correspond to sentences, and the columns correspond to discourse entities appearing in the text. They consider noun phrases (NP) as entities, and employ a coreference resolver to detect mentions of the same entity (e.g., *Obama*, *the president*). Each entry $G_{i,j}$ in the entity grid represents the syntactic role that entity e_j plays in sentence s_i, which can be one of: subject (**S**), object (**O**), or other (**X**). In addition, entities not appearing in a sentence are marked by a special symbol (**-**). If an entity appears more than once with different grammatical roles in the same sentence, the role with the highest rank (S $\succ$ O $\succ$ X) is considered.

To represent the entity grid using a feature vector, Barzilay and Lapata (2008) compute probability for each local entity transition of length k (i.e., $\{S, O, X, -\}^k$), and represent each grid by a vector of 4^k transitions probabilities. To distinguish between transitions of important entities from unimportant ones, they consider the *salience* of the entities, which they quantify by their occurrence frequency in the document. Assessment of text coherence is then formulated as a ranking problem in an SVM preference ranking framework (Joachims, 2002).

Subsequent studies proposed to extend the basic entity grid model. Filippova and Strube (2007) attempted to improve the model by grouping entities based on semantic relatedness, but did not get significant improvement. Elsner and Charniak (2011) proposed a number of improvements. They initially show significant improvement by including non-head nouns (i.e., nouns that do not head NPs) as entities in the grid.[2] Then, they extend the grid to distinguish between entities of different types by incorporating entity-specific features like named entity, noun class, modifiers, etc. These extensions led to the best results reported so far.

	UNIT	PRODUCTS	RESEARCH	COMPANY	PARTS	CONTROLS	INDUSTRY	ELECTRONICS	TERM	CONCERN	AEROSPACE	EMPLOYEES	SERVICES	LOS ANGELES	EATON
s_0	O	–	X	X	–	–	–	–	–	–	–	X	–	–	X
s_1	–	–	–	–	–	–	–	–	S	–	–	–	–	–	–
s_2	–	O	–	–	–	–	X	–	–	–	–	O	O	X	–
s_3	–	–	–	–	X	X	–	X	–	O	X	–	–	–	S

s_0: Eaton Corp. said it sold its Pacific Sierra Research unit to a company formed by employees of that unit.

s_1: Terms were not disclosed.

s_2: Pacific Sierra, based in Los Angeles, has 200 employees and supplies professional services and advanced products to industry.

s_3: Eaton is an automotive parts, controls and aerospace electronics concern.

Figure 1: Entity grid representation (top) for a document (below) from WSJ (id: 0079).

The Entity grid and its extensions have been successfully applied to many downstream tasks including coherence rating (Barzilay and Lapata, 2008), essay scoring (Burstein et al., 2010), story generation (McIntyre and Lapata, 2010), and readability assessment (Pitler et al., 2010; Barzilay and Lapata, 2008). They have also been critical components in state-of-the-art sentence ordering models (Soricut and Marcu, 2006; Elsner and Charniak, 2011; Lin et al., 2011).

2.1 Limitations of Entity Grid Models

Despite its success, existing entity grid models are limited in several ways.

• Existing models use discrete representation for grammatical roles and features, which leads to the so-called **curse of dimensionality** problem (Bengio et al., 2003). In particular, to model transitions of length k with $\mathcal{R}$ different grammatical roles, the basic entity grid model needs to compute $\mathcal{R}^k$ tran-

[1] https://github.com/datienguyen/cnn_coherence/

[2] They match the nouns to detect coreferent entities.

sition probabilities from a grid. One can imagine that the estimated distribution becomes sparse as k increases. This prevents the model from considering longer transitions – existing models use $k \leq 3$. This problem is exacerbated when we want to include entity-specific features, as the number of parameters grows exponentially with the number of features (Elsner and Charniak, 2011).

• Existing models compute feature representations from entity grids in a task-agnostic way. In other words, feature extraction is decoupled from the target downstream tasks. This can limit the model's capacity to learn task-specific features. Therefore, models that can be trained in an end-to-end fashion on different target tasks are desirable.

In the following section, we present a neural architecture that allows us to capture long range entity transitions along with arbitrary entity-specific features without loosing generalization. We also present an end-to-end training method to learn task-specific features automatically.

3 The Neural Coherence Model

Figure 2 summarizes our neural architecture for modeling local coherence, and how it can be trained in a pairwise fashion. The architecture takes a document as input, and first extracts its entity grid.[3] The first layer of the neural network transforms each grammatical role in the grid into a distributed representation, a real-valued vector. The second layer computes high-level features by going over each column (transitions) of the grid. The following layer selects the most important high-level features, which are in turn used for coherence scoring. The features computed at different layers of the network are automatically trained by backpropagation to be relevant to the task. In the following, we elaborate on the layers of the neural network model.

(I) Transforming grammatical roles into feature vectors: Grammatical roles are fed to our model as indices taken from a finite vocabulary $\mathcal{V}$. In the simplest scenario, $\mathcal{V}$ contains $\{S, O, X, -\}$. However, we will see in Section 3.1 that as we include more entity-specific features, $\mathcal{V}$ can contain more symbols. The first layer of our network maps each of these indices into a distributed representation $\mathbb{R}^d$ by looking up a shared embedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d}$. We consider E a model parameter to be learned by backpropagation on a given task. We can initialize E randomly or using pretrained vectors trained on a general coherence task.

Given an entity grid G with columns representing entity transitions over sentences in a document, the lookup layer extracts a d-dimensional vector for each entry $G_{i,j}$ from E. More formally,

$$\mathcal{L}(G) = \Big\langle E(G_{1,1}) \ \cdots \ E(G_{i,j}) \ \cdots \ E(G_{m,n}) \Big\rangle \tag{1}$$

where $E(G_{i,j})$ refers to the row in E that corresponds to the grammatical role $G_{i,j} \in \mathcal{V}$; m is the total number of sentences and n is the total number of entities in the document. The output $\mathcal{L}(G)$ is a tensor in $\mathbb{R}^{m \times n \times d}$, which is fed to the next layer of the network as we describe below.

(II) Modeling entity transitions: The vectors produced by the lookup layer are combined by subsequent layers of the network to generate a coherence score for the document. To compose higher-level features from the embedding vectors, we make the following modeling assumptions:

• Similar to existing entity grid models, we assume there is no spatio-temporal relation between the entities in a document. In other words, columns in a grid are treated independently.

• We are interested in modeling entity transitions of arbitrary lengths in a *location-invariant* way. This means, we aim to compose local patches of entity transitions into higher-level representations, while treating the patches independently of their position in the entity grid.

Under these assumptions, the natural choice to tackle this problem is to use a *convolutional* approach, used previously to solve other NLP tasks (Collobert et al., 2011; Kim, 2014).

Convolution layer: A convolution operation involves applying a *filter* $\mathbf{w} \in \mathbb{R}^{k.d}$ (i.e., a vector of weight parameters) to each entity transition of length k to produce a new abstract feature

$$h_t = f(\mathbf{w}^T \mathcal{L}_{t:t+k-1,j} + b_t) \tag{2}$$

where $\mathcal{L}_{t:t+k-1,j}$ denotes the concatenation of k vectors in the lookup layer representing a transition of length k for entity e_j in the grid, b_t is a bias

[3] For clarification, pairwise input as shown in the figure is required only to train the model.

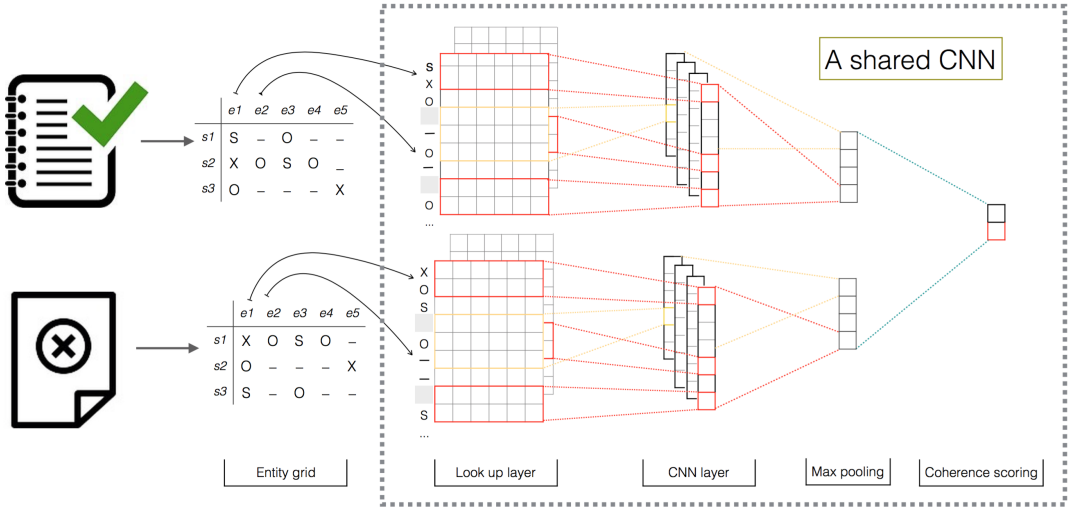


Figure 2: Neural architecture for modeling local coherence and the pairwise training method.

term, and f is a nonlinear activation function, e.g., ReLU (Nair and Hinton, 2010) in our model.

We apply this filter to each possible k-length transitions of different entities in the grid to generate a *feature map*, $\mathbf{h}^i = [h_1, \cdots, h_{m.n+k-1}]$. We repeat this process N times with N different filters to get N different feature maps (Figure 2). Notice that we use a *wide* convolution (Kalchbrenner et al., 2014), as opposed to *narrow*, to ensure that the filters reach entire columns of a grid, including the boundary entities. This is done by performing *zero-padding*, where out-of-range (i.e., for $t < 0$ or $t > \{m, n\}$) vectors are assumed to be zero.

Convolutional filters learn to compose local transition features of a grid into higher-level representations automatically. Since it operates over the distributed representation of grid entries, compared to traditional grid models, the transition length k can be sufficiently large (e.g., $5 - 8$ in our experiments) to capture long-range transitional dependencies without overfitting on the training data. Moreover, unlike existing grid models that compute transition probabilities from a single document, embedding vectors and convolutional filters are learned from all training documents, which helps the neural framework to obtain better generalization and robustness.

Pooling layer: After the convolution, we apply a *max-pooling* operation to each feature map.

$$\mathbf{m} = [\mu_p(\mathbf{h}^1), \cdots, \mu_p(\mathbf{h}^N)] \quad (3)$$

where $\mu_p(\mathbf{h}^i)$ refers to the max operation applied to each non-overlapping[4] window of p features in the feature map $\mathbf{h}^i$. Max-pooling reduces the output dimensionality by a factor of p, and it drives the model to capture the most salient local features from each feature map in the convolutional layer.

Coherence scoring: Finally, the max-pooled features are used in the output layer of the network to produce a coherence score $y \in \mathbb{R}$.

$$y = \mathbf{v}^T \mathbf{m} + b \quad (4)$$

where $\mathbf{v}$ is the weight vector and b is a bias term.

Why it works: Intuitively, each filter detects a specific transition pattern (e.g., 'SS-O-X' for a coherent text), and if this pattern occurs somewhere in the grid, the resulting feature map will have a large value for that particular region and small values for other regions. By applying max pooling on this feature map, the network then discovers that the transition appeared in the grid.

3.1 Incorporating Entity-Specific Features

Our model as described above neuralizes the basic entity grid model that considers only entity transitions without distinguishing between types of the entities. However, as Elsner and Charniak (2011) pointed out entity-specific features could be crucial for modeling local coherence. One simple way to incorporate entity-specific features into our model is to attach the feature value (e.g., named entity type) with the grammatical role in the grid.

[4] We set the *stride size* to be the same as the pooling length p to get non-overlapping regions.

For example, if an entity e_j of type PERSON appears as a subject (S) in sentence s_i, the grid entry $G_{i,j}$ can be encoded as PERSON-S.

3.2 Training

Our neural model assigns a coherence score to an input document d based on the degree of local coherence observed in its entity grid G. Let $y = \phi(G|\theta)$ define our model that transforms an input grid G to a coherence score y through a sequence of lookup, convolutional, pooling, and linear projection layers with parameter set θ. The parameter set θ includes the embedding matrix E, the filter matrix W, the weight vector $\mathbf{v}$, and the biases. We use a *pairwise ranking* approach (Collobert et al., 2011) to learn θ.

The training set comprises *ordered* pairs (d_i, d_j), where document d_i exhibits a higher degree of coherence than document d_j. As we will see in Section 4 such orderings can be obtained automatically or through manual annotation. In training, we seek to find θ that assigns a higher coherence score to d_i than to d_j. We minimize the following ranking objective with respect to θ:

$$\mathcal{J}(\theta) = \max\{0, 1 - \phi(G_i|\theta) + \phi(G_j|\theta)\} \quad (5)$$

where G_i and G_j are the entity grids corresponding to documents d_i and d_j, respectively. Notice that (also shown in Figure 2) the network shares its layers (and hence θ) to obtain $\phi(G_i|\theta)$ and $\phi(G_j|\theta)$ from a pair of input grids (G_i, G_j).

Barzilay and Lapata (2008) adopted a similar ranking criterion using an SVM preference kernel learner as they argue coherence assessment is best seen as a ranking problem as opposed to classification (*coherent* vs. *incoherent*). Also, the ranker gives a scoring function ϕ that a text generation system can use to compare alternative hypotheses.

4 Evaluation Tasks

We evaluate the effectiveness of our coherence models on two different evaluation tasks: sentence ordering and summary coherence rating.

4.1 Sentence Ordering

Following Elsner and Charniak (2011), we evaluate our models on two sentence ordering tasks: discrimination and insertion.

In the *discrimination* task (Barzilay and Lapata, 2008), a document is compared to a random permutation of its sentences, and the model is considered correct if it scores the original document higher than the permuted one. We use 20 permutations of each document in the test set in accordance with previous work.

In the *insertion* task (Elsner and Charniak, 2011), we evaluate models based on their ability to locate the original position of a sentence previously removed from a document. To measure this, each sentence in the document is removed in turn, and an insertion place is located for which the model gives the highest coherence score to the document. The insertion score is then computed as the average fraction of sentences per document reinserted in their actual position.

Discrimination can be easier for longer documents, since a random permutation is likely to be different than the original one. Insertion is a much more difficult task since the candidate documents differ only by the position of one sentence.

	Sections	# Doc.	# Pairs	Avg. # Sen.
TRAIN	00-13	1,378	26,422	21.5
TEST	14-24	1,053	20,411	22.3

Table 1: Statistics on the WSJ dataset.

Dataset: For sentence ordering tasks, we use the Wall Street Journal (WSJ) portion of Penn Treebank, as used by Elsner and Charniak (2008, 2011); Lin et al. (2011); Feng et al. (2014). Table 1 gives basic statistics about the dataset. Following previous works, we use 20 random permutations of each article, and we exclude permutations that match the original document.[5] The fourth column (# Pairs) in Table 1 shows the resulting number of (*original*, *permuted*) pairs used for training our model and for testing in the discrimination task.

Some previous studies (Barzilay and Lapata, 2008; Li and Hovy, 2014) used the AIRPLANES and the EARTHQUAKES corpora, which contain reports on airplane crashes and earthquakes, respectively. Each of these corpora contains 100 articles for training and 100 articles for testing. The average number of sentences per article in these two corpora is 10.4 and 11.5, respectively.

We preferred the WSJ corpus for several reasons. First and most importantly, the WSJ corpus is larger than other corpora (see Table 1). A large training set is crucial for learning effective

[5] Short articles may produce many matches.

deep learning models (Collobert et al., 2011), and a large enough test set is necessary to make a general comment about model performance. Second, as Elsner and Charniak (2011) pointed out, texts in AIRPLANES and EARTHQUAKES are constrained in style, whereas WSJ documents are more like normal informative articles. Third, we could reproduce results on this dataset for the competing systems (e.g., entity grid and its extensions) using the publicly available Brown coherence toolkit.[6]

4.2 Summary Coherence Rating

We further evaluate our models on the summary coherence rating task proposed by Barzilay and Lapata (2008), where we compare rankings given by a model to a pair of summaries against rankings elicited from human judges.

Dataset: The summary dataset was extracted from the Document Understanding Conference (DUC'03), which contains 6 clusters of multi-document summaries produced by human experts and 5 automatic summarization systems. Each cluster has 16 summaries of a document with pairwise coherence rankings given by humans judges; see (Barzilay and Lapata, 2008) for details on the annotation method. There are 144 pairs of summaries for training and 80 pairs for testing.

5 Experiments

In this section, we present our experiments — the models we compare, their settings, and the results.

5.1 Models Compared

We compare our coherence model against a random baseline and several existing models.

Random: The Random baseline makes a random decision for the evaluation tasks.

Graph-based Model: This is the graph-based unsupervised model proposed by Guinaudeau and Strube (2013). We use the implementation from the cohere[7] toolkit (Smith et al., 2016), and run it on the test set with syntactic projection (command line option 'projection=3') for graph construction. This setting yielded best scores for this model.

Distributed Sentence Model: Li and Hovy (2014) proposed this neural model for measuring text coherence. The model first encodes each sentence in a document into a fixed-length vector using a recurrent or a recursive neural network. Then it computes the coherence score of the document by aggregating the scores estimated for each window of three sentences in the document. We used the implementation made publicly available by the authors.[8] We trained the model on our WSJ corpus with 512, 1024 and 1536 minibatch sizes for a maximum of 25 epochs.[9] The model that used minibatch size of 512 and completed 23 epochs achieved the best accuracy on the DEV set. We applied this model to get the scores on the TEST set.

Grid-all nouns (E&C): This is the simple extension of the original entity grid model, where all nouns are considered as entities. Elsner and Charniak (2011) report significant gains by considering all nouns as opposed to only head-nouns. Results for this model were obtained by training the baseline entity grid model (command line option '-n') in the Brown coherence toolkit on our dataset.

Extended grid (E&C): This represents the extended entity grid model of Elsner and Charniak (2011) that uses 9 entity-specific features; 4 of them were computed from external corpora. This model considers all nouns as entities. For this system, we train the extended grid model (command line option '-f') in the Brown coherence toolkit.

Grid-CNN: This is our proposed neural extension of the basic entity grid (all nouns), where we only consider entity transitions as input.

Extended Grid-CNN: This corresponds to our neural model that incorporates entity-specific features following the method described in Section 3.1. To keep the model simple, we include only three entity-specific features from (Elsner and Charniak, 2011) that are easy to compute and do not require any external corpus. The features are: (*i*) *named entity* type, (*ii*) *salience* as determined by occurrence frequency of the entity, and

[6] https://bitbucket.org/melsner/browncoherence

[7] https://github.com/karins/CoherenceFramework

[8] http://cs.stanford.edu/ bdlijiwei/code/

[9] Our WSJ corpus is about 14 times larger than their ACCIDENT or EARTHQUAKE corpus (1378 vs. 100 training articles), and the articles in our corpus are generally longer than the articles in their corpus (on average 22 vs. 10 sentences per article). Also, the vocabulary in our corpus is much larger than their vocabulary (45462 vs. 4758). Considering these factors and the fact that their Java-based implementation does not support GPU and parallelization, it takes quite long to train and to validate their model on our dataset. In our experiments, depending on the minibatch size, it took approximately 3-5 days to complete only one epoch of training!

	Batch	Emb.	Dropout	Filter	Win.	Pool
Grid-CNN	128	100	0.5	150	6	6
Ext. Grid-CNN	32	100	0.5	150	5	6

Table 2: Optimal hyper-parameter setting for our neural models based on development set accuracy.

(*iii*) whether the entity has a *proper* mention.

5.2 Settings for Neural Models

We held out 10% of the training documents to form a development set (DEV) on which we tune the hyper-parameters of our neural models. For discrimination and insertion tasks, the resulting DEV set contains 138 articles and 2,678 pairs after removing the permutations that match the original documents. For the summary rating task, DEV contains 14 pairs of summaries.

We implement our models in Theano (Theano Development Team, 2016). We use rectified linear units (ReLU) as activations (f). The embedding matrix is initialized with samples from uniform distribution $\mathcal{U}(-0.01, 0.01)$, and the weight matrices are initialized with samples from glorot-uniform distribution (Glorot and Bengio, 2010).

We train the models by optimizing the pairwise ranking loss in Equation 5 using the gradient-based online learning algorithm RMSprop with parameters (ρ and ϵ) set to the values suggested by Tieleman and Hinton (2012).[10] We use up to 25 epochs. To avoid overfitting, we use dropout (Srivastava et al., 2014) of hidden units, and do *early stopping* by observing accuracy on the DEV set – if the accuracy does not increase for 10 consecutive epochs, we exit with the best model recorded so far. We search for optimal mini**batch** size in $\{16, 32, 64, 128\}$, **emb**edding size in $\{80, 100, 200\}$, **dropout** rate in $\{0.2, 0.3, 0.5\}$, **filter** number in $\{100, 150, 200, 300\}$, **win**dow size in $\{2, 3, 4, 5, 6, 7, 8\}$, and **pool**ing length in $\{3, 4, 5, 6, 7\}$. Table 2 shows the optimal hyper-parameter setting for our models. The best model on DEV is then used for the final evaluation on the TEST set. We run each experiment five times, each time with a different random seed, and we report the average of the runs to avoid any randomness in results. Statistical significance tests are done using an *approximate randomization* test based on the accuracy. We used SIGF V.2 (Padó, 2006) with 10,000 iterations.

[10]Other adaptive algorithms, e.g., ADAM (Kingma and Ba, 2014), ADADELTA (Zeiler, 2012) gave similar results.

	Discr. Acc	**Discr.** F_1	**Ins.**
Random	50.0	50.0	12.60
Graph-based (G&S)	64.23	65.01	11.93
Dist. sentence (L&H)	77.54	77.54	19.32
Grid-all nouns (E&C)	81.58	81.60	22.13
Extended Grid (E&C)	84.95	84.95	23.28
Grid-CNN	85.57†	85.57†	23.12
Extended Grid-CNN	**88.69**†	**88.69**†	**25.95**†

Table 3: Coherence evaluation results on **Discr**imination and **Ins**ertion tasks. † indicates a neural model is significantly superior to its non-neural counterpart with p-value < 0.01.

5.3 Results on Sentence Ordering

Table 3 shows the results on discrimination and insertion tasks. The graph-based model gets the lowest scores. This is not surprising considering that this model works in an unsupervised way. The distributed sentence model surprisingly performed poorly on our dataset. Among the existing models, the grid models get the best scores on both tasks. This demonstrates that entity transition, as a method to capture local coherence, is more effective than the sentence representation method.

Neuralization of the existing grid models yields significant improvements in most cases. The Grid-CNN model delivers absolute improvements of about 4% in discrimination and 1% in insertion over the basic grid model. When we compare our Extended Grid-CNN with its non-neural counterpart Extended Grid, we observe similar gains in discrimination and more gains (2.5%) in insertion. Note that the Extended Grid-CNN yields these improvements considering only a subset of the Extended Grid features. This demonstrates the effectiveness of distributed representation and convolutional feature learning method.

Compared to the discrimination task, gain in the insertion task is less verbose. There could be two reasons for this. First, as mentioned before, insertion is a harder task than discrimination. Second, our models were not trained specifically on the insertion task. The model that is trained to distinguish an original document from its random permutation may learn features that are not specific enough to distinguish documents when only one sentence differs. In the future, it will be interesting

	Acc	F_1
Random	50.0	50.0
Graph-based (G&S)	80.0	81.5
Grid (B&L)	83.8	-
Grid-CNN	85.0	85.0
Extended Grid-CNN	86.3	86.3
Pre-trained Grid-CNN	86.3	86.3
Pre-trained Ext. Grid-CNN	**87.5**	**87.5**

Table 4: Evaluation results on the **Summary Coherence Rating** task.

to see how the model performs when it is trained on the insertion task directly.

5.4 Results on Summary Coherence Rating

Table 4 presents the results on the summary coherence rating task, where we compare our models with the *reported* results of the graph-based method (Guinaudeau and Strube, 2013) and the initial entity grid model (Barzilay and Lapata, 2008) on the same experimental setting.[11] The extended grid model does not use pairwise training, therefore could not be trained on the summarization dataset. Since there are not many training instances, our neural models may not learn well for this task. Therefore, we also present versions of our model, where we use pre-trained models from discrimination task on WSJ corpus (last two rows in the table). The pre-trained models are then *fine-tuned* on the summary rating task.

We can observe that even without pre-training our models outperform existing models, and pre-training gives further improvements. Specifically, Pre-trained Grid-CNN gives an improvement of 2.5% over the Grid model, and including entity features pushes the improvement further to 3.7%.

6 Related Work

Barzilay and Lapata (2005, 2008) introduced the entity grid representation of discourse to model local coherence that captures the distribution of discourse entities across sentences in a text. They also introduced three tasks to evaluate the performance of coherence models: discrimination, summary coherence rating, and readability.

[11] Since we do not have access to the output of their systems, we could not do a significance test for this task.

A number of extensions of the basic entity grid model has been proposed. Elsner and Charniak (2011) included entity-specific features to distinguish between entities. Feng and Hirst (2012) used the basic grid representation, but improved its learning to rank scheme. Their model learns not only from original document and its permutations but also from ranking preferences among the permutations themselves. Guinaudeau and Strube (2013) convert a standard entity grid into a bipartite graph representing entity occurrences in sentences. To model local entity transition, the method constructs a directed projection graph representing the connection between adjacent sentences. Two sentences have a connected edge if they share at least one entity in common. The coherence score of the document is then computed as the average out-degree of sentence nodes.

In addition, there are some approaches that model text coherence based on coreferences and discourse relations. Elsner and Charniak (2008) proposed the discourse-new model by taking into account mentions of all referring expression (i.e., NPs) whether they are first mention (*discourse-new*) or subsequent (*discourse-old*) mentions. Given a document, they run a maximum-entropy classifier to detect each NP as a label $L_{np} \in \{new, old\}$. The coherence score of the document is then estimated by $\prod_{np:NPs} P(L_{np}|np)$. In this work, they also estimate text coherence through pronoun coreference modeling. Lin et al. (2011) assume that a coherent text has certain discourse relation patterns. Instead of modeling entity transitions, they model discourse role transitions between sentences. In a follow up work, Feng et al. (2014) trained the same model but using features derived from deep discourse structures annotated with Rhetorical Structure Theory or RST (Mann and Thompson, 1988b) relations. Louis and Nenkova (2012) introduced a coherence model based on syntactic patterns in text by assuming that sentences in a coherent discourse should share the same structural syntactic patterns.

In recent years, there has been a growing interest in neuralizing traditional NLP approaches – language modeling (Bengio et al., 2003), sequence tagging (Collobert et al., 2011), syntactic parsing (Socher et al., 2013), and discourse parsing (Li et al., 2014), etc. Following this tradition, in this paper we propose to neuralize the popular entity grid models. Li and Hovy (2014) also proposed a

neural framework to compute the coherence score of a document by estimating coherence probability for every window of L sentences (in their experiments, $L = 3$). First, they use a recurrent or a recursive neural network to compute the representation for each sentence in L from its words and their pre-trained embeddings. Then the concatenated vector is passed through a non-linear hidden layer, and finally the output layer decides if the window of sentences is a coherent text or not. Our approach is fundamentally different from their approach; our model operates over entity grids, and we use convolutional architecture to model sufficiently long entity transitions.

7 Conclusion and Future Work

We presented a local coherence model based on a convolutional neural network that operates over the distributed representation of entity transitions in the grid representation of a text. Our architecture can model sufficiently long entity transitions, and can incorporate entity-specific features without loosing generalization power. We described a pairwise ranking approach to train the model on a target task and learn task-specific features. Our evaluation on discrimination, insertion and summary coherence rating tasks demonstrates the effectiveness of our approach yielding the best results reported so far on these tasks.

In future, we would like to include other sources of information in our model. Our initial plan is to include rhetorical relations, which has been shown to benefit existing grid models (Feng et al., 2014). We would also like to extend our model to other forms of discourse, especially, asynchronous conversations, where participants communicate with each other at different times (e.g., forum, email).

Acknowledgments

We thank Regina Barzilay and Mirella Lapata for making their summarization data available and Micha Elsner for making his coherence toolkit publicly available. We also thank the three anonymous ACL reviewers and the program chairs for their insightful comments on the paper.

References

N. Asher and A. Lascarides. 2003. *Logics of Conversation*, Cambridge University Press.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Ann Arbor, Michigan, ACL '05, pages 141–148.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1):1–34. http://www.aclweb.org/anthology/J08-1001.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3. http://dl.acm.org/citation.cfm?id=944919.944966.

Jill Burstein, Joel Tetreault, and Slava Andreyev. 2010. Using entity-based features to model coherence in student essays. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, HLT '10, pages 681–684.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537.

Micha Elsner and Eugene Charniak. 2008. Coreference-inspired coherence modeling. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*. Association for Computational Linguistics, Columbus, Ohio, HLT-Short '08, pages 41–44.

Micha Elsner and Eugene Charniak. 2011. Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. Association for Computational Linguistics, Portland, Oregon, HLT '11, pages 125–129.

Vanessa Wei Feng and Graeme Hirst. 2012. Extending the entity-based coherence model with multiple ranks. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Avignon, France, EACL '12, pages 315–324.

Vanessa Wei Feng, Ziheng Lin, and Graeme Hirst. 2014. The impact of deep hierarchical discourse structures in the evaluation of text coherence. In *COLING*.

Katja Filippova and Michael Strube. 2007. Extending the entity-grid coherence model to semantically related entities. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*. Association for Computational Linguistics, Germany, ENLG '07, pages 139–142.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*. Sardinia, Italy, volume 9, pages 249–256.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Comput. Linguist.* 21(2):203–225.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*. pages 93–103.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Edmonton, Alberta, Canada, KDD '02, pages 133–142.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 2039–2048. http://www.aclweb.org/anthology/D14-1218.

Jiwei Li, Rumeng Li, and Eduard H Hovy. 2014. Recursive deep models for discourse parsing. In *EMNLP*. pages 2061–2069.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Association for Computational Linguistics, Portland, Oregon, HLT '11, pages 997–1006.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP-CoNLL '12, pages 1157–1168. http://dl.acm.org/citation.cfm?id=2390948.2391078.

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

William C Mann and Sandra A Thompson. 1988b. Rhetorical structure theory: Toward a functional theory of text organization. *Text* 8(3):243–281.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, ACL '10, pages 1562–1572.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In Johannes Frnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Omnipress, pages 807–814. http://www.icml2010.org/papers/432.pdf.

Sebastian Padó. 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

Emily Pitler, Annie Louis, and Ani Nenkova. 2010. Automatic evaluation of linguistic quality in multi-document summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, ACL '10, pages 544–554.

Karin Sim Smith, Wilker Aziz, and Lucia Specia. 2016. Cohere: A toolkit for local coherence. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Portoroz, Slovenia.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 455–465. http://www.aclweb.org/anthology/P13-1045.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*. Association for Computational Linguistics, Sydney, Australia, COLING-ACL '06, pages 803–810.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.

T. Tieleman and G Hinton. 2012. *RMSprop*, COURSERA: Neural Networks

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Data-Driven Broad-Coverage Grammars for Opinionated Natural Language Generation (ONLG)

Tomer Cagan
School of Computer Science
The Interdisciplinary Center
Herzeliya, Israel
cagan.tomer@idc.ac.il

Stefan L. Frank
Centre for Language Studies
Radboud University
Nijmegen, The Netherlands
s.frank@let.ru.nl

Reut Tsarfaty
Mathematics and Computer Science
The Open University of Israel
Ra'anana, Israel
reutts@openu.ac.il

Abstract

Opinionated natural language generation (ONLG) is a new, challenging, NLG task in which we aim to automatically generate human-like, subjective, responses to opinionated articles online. We present a data-driven architecture for ONLG that generates subjective responses triggered by users' agendas, based on automatically acquired wide-coverage generative grammars. We compare three types of grammatical representations that we design for ONLG. The grammars interleave different layers of linguistic information, and are induced from a new, enriched dataset we developed. Our evaluation shows that generation with Relational-Realizational (Tsarfaty and Sima'an, 2008) inspired grammar gets better language model scores than lexicalized grammars à la Collins (2003), and that the latter gets better human-evaluation scores. We also show that conditioning the generation on topic models makes generated responses more relevant to the document content.

1 Introduction

Interaction in social media has become increasingly prevalent nowadays. It fundamentally changes the way businesses and consumers behave (Qualman, 2012), it is instrumental to the success of individuals and businesses (Haenlein and Kaplan, 2009) and it also affects political regimes (Howard et al., 2011; Lamer, 2012). In particular, *automatic* interaction in natural language in social media is now a common theme, as seen in the rapid popularization of chat applications, chat-bots, and "smart agents" aiming to conduct human-like interactions in natural language.

So far, generation of human-like interaction in general has been addressed mostly commercially, where there is a movement towards online response automation (Owyang, 2012; Mah, 2012), and movement away from script-based interaction towards interactive chat bots (Mori et al., 2003; Feng et al., 2006). These efforts provide an automated one-size-fits-all type of interaction, with no particular expression of particular sentiments, topics, or opinions. In academia, work on generating human-like interaction focused so far on generating responses to tweets (Ritter et al., 2011; Hasegawa et al., 2013) or taking turns in short dialogs (Li et al., 2017). However, the architectures assumed in these studies implement *sequence to sequence* (seq2seq) mappings, which do not take into account topics, sentiments or agendas of the intended responders.

Many real-world tasks and applications would benefit from automatic interaction that is generated intendedly based on a certain user profile or agenda. For instance, this can help promoting a political candidate or a social idea in social media, aiding people forming and expressing opinions on specific topics, or, in *human-computer interfaces* (HCI), making the computer-side generated utterances more meaningful, and ultimately more human-like (assuming that human-like interaction is very often affected by opinion, agenda, style, etc.).

In this work we address the *opinionated natural language generation* (ONLG) task, in which we aim to automatically generate human-like responses to opinionated articles. These responses address particular topics and reflect diverse sentiments towards them, in accordance to predefined user agendas. This is an open-ended and unstructured generation challenge, which is closely tied to the communicative goals of actual human responders.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1331–1341
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1122

In previous work we addressed the ONLG challenge using a *template-based* approach (Cagan et al., 2014). The proposed system generated subjective responses to articles, driven by *user agendas*. While the evaluation showed promising results in human-likeness and relevance ratings, the template-based system suffers from low output variety, which leads to a learning effect that reduced the perceived human-likeness of generated responses over time.

In this work we tackle ONLG from a data-driven perspective, aiming to circumvent such learning effects and repetitive patterns in template-based generation. Here, we approach generation via automatically inducing *broad-coverage generative grammars* from a large corpus, and using them for response generation. More specifically, we define a grammar-based generation architecture and design different grammatical representations suitable for the ONLG task. Our grammars interleave different layers of linguistic information — including phrase-structure and dependency labels, lexical items, and levels of sentiment — with the goal of making responses both human-like and relevant. In classical NLG terms, these grammars offer the opportunity for both *micro-planning* and *surface realization* (Reiter and Dale, 1997) to unfold together. We implement a generator and a search strategy to carry out the generation, and sort through possible candidates to get the best ones.

We evaluate the generated responses and the underlying grammars using automated metrics as well as human evaluation inspired by the Turing test (cf. Cagan et al. (2014) and Li et al. (2017)). Our evaluation shows that while *relational realizational* (RR) inspired grammars (Tsarfaty and Sima'an, 2008) get good language model scores, simple head-driven *lexicalized* grammars à la Collins (2003) get better human rating and are more sensitive to sentiment. Furthermore, we show that incorporating topic models into the grammar-based generation makes the generated responses more relevant to the document content. Finally, our human evaluations results show no learning effect. That is, human raters are unable to discover in the generated responses typical structures that would lead them to consider the responses machine-generated.

The remainder of this paper is organized as follows. In Section 2 we discuss the formal model, and in Section 3 we present the proposed end-to-end ONLG architecture. In Section 4 we introduce the grammars we define, and we describe how we use them for generation in Section 5. We follow that with our empirical evaluation in Section 6. In Section 7 we discuss related and future work, and in Section 8 we summarize conclude.

2 The Formal Model

Task Definition. Let d be a document containing a single article, and let a be a user agenda as in Cagan et al. (2014). Specifically, a user agenda a can consist of one or more pairs of a *topic* (represented by a weighted bag-of-words) and an associated *sentiment*. Let c be an analysis function on documents such that $c(d)$ yields a set of *content elements* which are also pairings of topics and sentiments. The operation $\otimes$ represents the intersection of the sets of content elements in the document and in the user agenda. We cast ONLG as a prediction function which maps the intersection $a \otimes c(d)$ to a sentence $y \in \Sigma^*$ in natural language (in our case, Σ is the vocabulary of English):

$$f_{response}(a \otimes c(d)) = y \tag{1}$$

For any non-empty intersection, a response is generated which is related to the topic of the intersection and the sentiments defined towards this topic. The relation between the sentiment in the user agenda and the sentiment reflected in the document is a simple *xor* function: when the user and the author share a sentiment toward a topic the response is positive, else it is negative.

Objective Function. Let G be a formal generative grammar and let T be the set of trees strongly generated by G. In our proposed data-driven, grammar-based, generation architecture, we define $f_{response}$ as a function selecting a most probable tree $t \in T$ derived by G, given the intersection of document content and user agenda.

$$f_{response}(a \otimes c(d)) = \underset{\{w|w=yield(t),t\in T\}}{\operatorname{argmax}} P(w,t|a \otimes c(d)) \tag{2}$$

Here, $w = yield(t)$ is the sequence of terminals that defines the leaves of the tree, which is then picked as the generated response.

Assuming that G is a context-free grammar, we can spell out the probabilistic expression in Equation (2) as a history-based probabilistic model where $root(t)$ selects a starting point for the

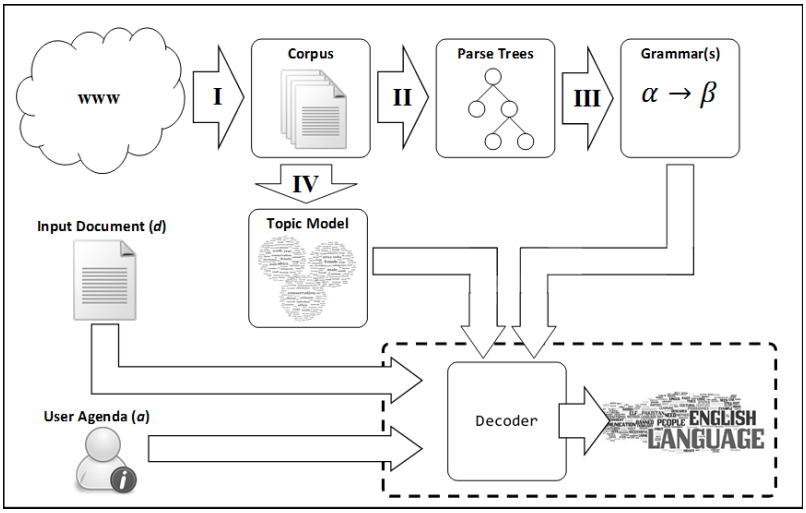


Figure 1: The end-to-end, data-driven, grammar-based generation architecture.

derivation, $der(t)$ selects the sequence of syntactic rules to be applied, and $yield(t)$ selects the sequence of terminals that forms the response all conditioned on the derivation history.

$$P(w,t|.) = P(root(t)|a \otimes c(d)) \quad (3a)$$
$$\times P(der(t)|root(t), a \otimes c(d)) \quad (3b)$$
$$\times P(yield(t)|root(t), der(t), a \otimes c(d)) \quad (3c)$$

Using standard independence assumptions, Eq. (3) may be re-written as a chain of local decisions, conditioned on selected aspects of the generation history, marked here by the function Φ.

$$P(w,t|.) \approx P(root|\Phi(a \otimes c(d))) \times \quad (4a)$$
$$\prod_{rule_j \in der(t)} P(rule_j|\Phi(root, a \otimes c(d))) \times \quad (4b)$$
$$\prod_{w_i \in yield(t)} P(w_i|\Phi(t, a \otimes c(d))) \quad (4c)$$

In words, the probability of the starting rule (4a) is multiplied with the probability of each of the rules in the derivation (4b) and the probability of each of the terminal nodes in the tree (4c). Each decision may be conditioned on previously generated part(s) of the structure, as well as the intersection of the input document content and user agenda.

3 The Architecture

A bird's-eye view of the architecture we propose is depicted in Figure 1. The process consists of an offline component containing (I) *corpus collection*, (II) *automatic annotation*, (III) *grammar induction*, and (IV) *topic-model training*. The induced grammar along with a predefined user agenda and the pre-trained topic model are provided as input to the online generation component, which is marked with the dashed box in Figure 1.

In (I) *corpus collection*, we collect a set of documents D with corresponding user comments. The documents in the corpus are used for training a topic model (IV), which is used for topic inference given a new input document d. The collected comments are used for inducing a wide-coverage grammar G for response generation.

To realize the goal of ONLG, we aim to jointly model opinion, structure and lexical decisions in our induced grammars. To this end, in (II) *automatic annotation* we enrich the user comments with annotations that reflect different levels of linguistic information, as detailed in Section 4.

In (III) *grammar induction* we induce a generative grammar G from the annotated corpus, following the common methodology of inducing PCFGs from syntactically annotated corpora (Charniak, 1995; Collins, 2003). We traverse the annotated trees from (III) and use maximum likelihood estimation for learning rule probabilities. No smoothing is done, and in order to filter noise from possibly erroneous parses, we use a frequency cap to define which rules can participate in derivations.

We finally define and implement an efficient grammar-based generator, termed here the *decoder*, which carries out the generation and calculates the objective function in Eq. (4). The algorithm is described in Section 5.

4 The Grammars

Base Grammar. A central theme in this research is generating sentences that express a certain sentiment. Our base grammatical representation is inspired by the Stanford sentiment classification parser (Socher et al., 2013) which annotates every non-ternminal node with one of five sentiment classes $s \in \{-2, -1, 0, 1, 2\}$.

Formally, each non-terminal in our base grammar includes a constituency category C and a sentiment class label s. The derivation of depth-1 trees with a parent node p and two daughters d_1, d_2 will thus appear as follows:

$$C_p[s_p] \rightarrow C_{d1}[s_{d1}] \;\; C_{d2}[s_{d2}]$$

The generative story imposed by this grammar is quite simple: each non-terminal node annotated with a sentiment can generate either a sequence of non-terminal daughters, or a single terminal node.

An example of a subtree and its generation sequence is given in Figure 2(Base). Here we see a positive NP which generates two daughters: a neutral DT and a positive NX. The positive NX generates a neutral noun NN and a positive modifying adjective JJ on its left. Such a derivation can yield NP terms such as "the good wife" or "an awesome movie", but will not generate "some terrible words". In this grammar, lexical realization is generated conditioned on local pre-terminals only, and independently of the syntactic structure.

While the generative story is simple, this grammar can capture complex interactions of sentiment. Such interactions take place in tree structures that include elements that may affect polarity, such as negation, modal verbs and so on (see Socher et al. (2013) and examples therein). In this work we assume a completely data-driven approach wherein such structures are derived based on previously observed sentiment-interactions in sentiment-augmented parses.

Lexicalized Grammar. Our base grammar suffers from a clear pitfall: the structure lacks sensitivity to lexical information, and vice versa. This base grammar essentially generates lexical items as an afterthought, conditioned only on the local part-of-speech label and sentiment value. Our first modification of the base grammar is *lexicalization* in the spirit of Collins (2003).

In this representation each non-terminal node is decorated with a phrase-structure category C and a sentiment label s, and it is augmented with a lexical *head* l_h. The lexical head is common to the parent and the left (or right) daughter. A new lexical item, termed *modifier* l_m, is introduced in the right (left) daughter. The resulting depth-1 subtree for a parent p with daughters d_1, d_2 and a lexical head on the left (without loss of generality) is:

$$C_p[s_p, l_h] \rightarrow C_{d1}[s_{d1}, l_h] \;\; C_{d2}[s_{d2}, l_m]$$

Lexicalization makes the grammar more useful for generation as lexical choices can be made at any stage of the derivation conditioned on part of the structure. But it has one drawback – it assumes very strong dependence between lexical items that happen to appear as sisters.

To overcome this, we define a *head-driven* generative story that follows the model of Collins (2003), where the mother non-terminal generates first the head node, and then, conditioned on the head it generates a modifying constituent to the left (right) of the head and its corresponding modifying lexical dependent. An example subtree and its associated head-driven generative story is illustrated in Figure 2(Lex).

Relational-Realizational Grammar. Generating phrase-structures along with lexical realization can manage *form* — control how sentences are built. For coherent generation we would like to also control for the *function* of nodes in the derivation. To this end, we define a grammar and a generative story in the spirit of the *Relational-Realizational* (RR) grammar of Tsarfaty (2010).

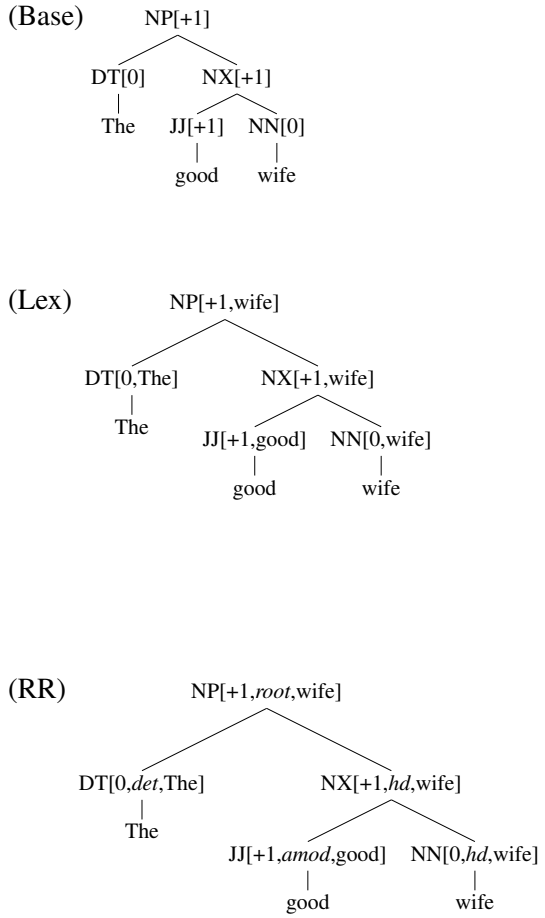
In our RR-augmented trees, each non-terminal node includes, on top of the phrase-structure category C, the lexical head l and the sentiment s, a relation label dep_i which determines its functional role in relation to its parent. The functional component will affect the selection of daughters so that the derived subtree fulfils its function. A depth-1 subtree will thus appear as follows:

$$C_i[s_i, dep_i, l_i] \rightarrow C_j[s_j, dep_j, l_i] \;\; C_k[s_k, dep_k, l_k]$$

The generative story of our RR representation follows the three-phase process defined by Tsarfaty and Sima'an (2008) and Tsarfaty (2010):

(i) *projection*: given a constituent and a sentiment value, generate a set of grammatical relations which define the functions of the daughters to be generated.

(a)



(b)

Type	LHS		RHS	
SYN	NP[+1]	→	DT[0]	NX[+1]
SYN	NX[+1]	→	JJ[+1]	NN[0]
LEX	DT[0]	→	The	
LEX	JJ[+1]	→	good	
LEX	NN[0]	→	wife	

Type	LHS		RHS
HEAD	NP[+1,wife]	$\rightarrow_r$	NX[+1]
MOD	NP[+1,wife], NX[+1]	$\rightarrow_l$	DT[0]
LEX-H	NP[+1,wife],NX[+1]	→	wife
LEX	NP[+1,wife], NX[+1,wife], DT[0]	→	the
HEAD	NX[+1,wife]	$\rightarrow_r$	NN[0]
MOD	NX[+1,wife], NN[0]	$\rightarrow_l$	JJ[+1]
LEX-H	NX[+1,wife], NN[0]	→	wife
LEX	NX[+1,wife], NN[0,wife],JJ[+1]	→	good

Type	LHS		RHS
PROJ	NP[+1]	→	{*amod,det,hd*}@NP[+1]
CONF	{*amod,det,hd*}@NP[+1]	→	<*det*>@NP[+1], <{*amod,hd*}>@NP[+1]
REAL-C	<*det*>@NP[+1]	→	DT[0]
REAL-C	<{*amod,hd*}>@NP[+1]	→	NX[+1]
REAL-L	DT[0,*det*]@NP[+1,*hd*,wife]	→	The
REAL-L	NX[+1,*hd*]@NP[+1,*hd*,wife]	→	wife
PROJ	NX[+1]	→	{*amod,hd*} @NX[+1]
CONF	{amod, hd}@NX[+1]	→	<*amod*>@NX[+1] , <*hd*>@NX[+1]
REAL-C	<*amod*>@NX[+1]	→	JJ[+1]
REAL-C	<*hd*>@NX[+1]	→	NN[0]
REAL-L	JJ[+1,*amod*]@NX[+1,*hd*,wife]	→	good
REAL-L	NN[+1,*hd*]@NX[+1,*hd*,wife]	→	wife

Figure 2: Our grammatical representations, with (a) a sample tree and (b) its generation sequence. A rule of type SYN marks syntactic rules, LEX indicates lexical realization, HEAD, MOD indicate head selection and modifier selection, PROJ,CONF,REAL indicate projection, configuration and realization, respectively. The @ sign indicates aspects in the generation history that the production is conditioned on (Φ in eq. 4).

(ii) *configuration*: given a constituent, sentiment and an unordered set of relations, an ordering for the relations is generated. Unlike the original RR derivations which fully order the set, here we partition the set into two disjoint sets (one of which is a singleton) and order them. This modification ensures that we adhere to binary trees.

(iii) *realization*: For each function-labels' set we select the daughter's constituent realizing it. We first generate the constituent and sentiment realizing this function, and then, conditioned on the constituent, sentiment, head and function, we select the lexical dependent.

An example tree along with its RR derivation is given in Figure 2(RR).

5 Grammar-Based Generation

Our grammar-based generator is a top-down algorithm which starts with a frontier that includes a selected root, and expands the tree continually by substituting non-terminals at the left-hand-side of rules with their daughters on the right hand side, until no more non-terminals exist. This generation procedure yields one sentence for any given root. Due to independence assumptions inherent in the generative processes we defined, there is no guarantee that generated sentences will be completely grammatical, relevant and human-like. To circumvent this, we develop an over-generation algorithm that modifies the basic algorithm to select multiple rules at each generation point, and apply them to uncover several derivation trees, or a *forest*.

We then use a variation on the beam search algorithm (Reddy, 1977) and devise a methodology to select the k-best scoring trees to be carried on to the next iteration. Specifically, we use a Breadth-First algorithm for expanding the tree and define a dynamic programming algorithm that takes the score of a derivation tree of $n-1$ *expanded* nodes, selects a new rule for the next non-expanded node, and from it, calculates the score of the expanded tree with now n nodes. For comparing the trees, we computed a score according to Eq. (4) for the tree generated so far, and used an average node score to neutralize size difference between trees.

To make sure our responses target a particular topic, we propose to condition the selection of lexical items at the root on the *topic* at the intersection of the document content and user agenda, essentially preferring derivations that yield words related to the input topic distribution. In practice we use topic model scores to estimate the root rule probability, selecting lexical item(s) for generation to start with:

$$\hat{P}(root(t)|a \otimes c(d)) = \hat{P}(\text{ROOT} \rightarrow l_1 l_2 | a \otimes c(d)) = \sum_{c=1}^{N} \sum_{i=1}^{2} tm\_weight(c) * word\_weight(c, l_i) \quad (5)$$

where $tm\_weight(c)$ is the weight of topic c in the topic distribution at the document-agenda intersection, and $word\_weight(c, l_i)$ is the weight of the lexical head word l_i within the word distribution of topic c in the given topic model.

The generation process ends when all derivations reach (at most) a pre-defined height (to avoid endless recursions). We then re-rank the generated candidates. The re-ranking is based on a 3-grams language model on the raw yield of the sentence, divided by the length of the sentence to obtain a per-word average and avoid length biases.[1]

6 Evaluation

Goal. We aim to evaluate the grammars' applicability to the ONLG task. Set in an open domain, it is not trivial to find a "gold-standard" for this task, or even a method to obtain one. Our evaluation thus follows two tracks: an automated assessment track, where we quantitatively assess the responses, and a Turing-like test similar to that of Cagan et al. (2014), where we aim to gauge human-likeness and response relevance.

[1]Here we use Microsoft's WebLM API which is part of the Microsoft Oxford Project (Microsoft, 2011).

Materials. We collected a new corpus of news articles and corresponding user comments from the NY-Times®web site, using their open Community API. We focus on sports news, which gave us 3,583 news articles and 13,100 user comments, or 55,700 sentences. The articles are then used for training a topic model using the Mallet library (McCallum, 2002). Next, we use the comments in the corpus to induce the grammars. To obtain our Base representation we parse the sentences using the Stanford CoreNLP suite (Manning et al., 2014) which can provide both phrase-structure and sentiment annotation. To obtain our Lexicalized representation we follow the same procedure, this time also using a head-finder which locates the head word for each non-terminal. To obtain the Relational-Realizational representation we followed the algorithm described in Tsarfaty et al. (2011), which, given both a constituency parse and a dependency parse of a sentence, unifies them into a lexicalized and functional phrase-structure. The merging is based on matching spans over words within the sentence.[2]

Setup. We simulated several scenarios. In each, the system generates sentences with one grammar ($G \in \{$*Base, Lex, RR*$\}$) and one scoring scheme (with/without topic model scores). The results of each simulation are 5,000 responses for each variant of the system, consisting of 1,000 sentences for each sentiment class, $s \in \{-2, -1, 0, 1, 2\}$. The same 5000 generated sentences were used in all experiments. We set the generator for trees of maximum depth of 13 which can yield up to 4096 words. In reality, the realization was of much shorter sentences. Examples for generated responses are given in Table 1.

6.1 Comparing Grammars

Goal and Metrics. In this experiment we compare and contrast the generation capacity of the grammars, using the following metrics:

(i) *Fluency* measures how grammatical or natural the generated sentences are. We base this measure on a probabilistic language model which gives an indication of how common word-sequences within the sentence are. We express fluency as a Language Model (LM) score which is calculated using the Microsoft Web ML API to get aggregated minus-log probabilities of all 3-grams

[2]The collected corpus and supplementary annotations are available at www.tomercagan.com/onlg.

in the sentence. The aggregated score is then normalized to give a per-word average in order to cancel any effects of sentence length.

(ii) *Sentiment Agreement* measures whether the inferred sentiment of the response matches the input sentiment parameter used for generation. Specifically, we take the raw yield of the generated tree (a sentence) and run it through the sentiment classifier implemented in Socher et al. (2013), to assign the full sentence one of 5 sentiment classes between -2 and $+2$. During evaluation, we compare the classified sentiment of the generated sentence is with the sentiment entered as input for the derivation of the sentence, and report the rate of agreement on (a) level ($-2.. + 2$) and (b) polarity ($-/+$), which is a more relaxed measure.

(iii) The *Consiceness/tightness* metric aims to evaluate which grammar derives a simpler structure across generations of similar content. Our tightness evaluation is based on the percentage of sentences that were fully realization as terminals within the specific height limit;[3] we simply observe how many trees have all leaves as terminal symbols. Intuitively, tighter grammars lead to improved performance and better control over the generated content. It is possible to think of what it captures in terms Occams Razor, preferring the simpler structure to derive comparable outcome.

Empirical Results The results of our evaluation are presented in Table 2. With respect to the above metrics, the RR grammar was more compact and natural compared to the lexicalized (LEX) grammar: the per-word LM Score for the RR is -5.6 as compared to -6.5 for LEX. Also, RR has 95.7% complete sentences as compared to only 67.3% for LEX. The LEX grammar was more sensitive to the sentiment input but only slightly, having a 44.6% sentiment agreement and 63.9% sentiment polarity agreement compared to 43.8% and 61.0% for RR grammar. The BASE grammar gave the worst performance for all measures. This provides preliminary evidence in support of incorporating surface realization (lexicalization) into the syntactic generation, rather than filling slots in retrospect.

6.2 Testing Relevance

Goal and Metrics Next we aim to evaluate the relevance of the responses to the input document triggering the response. We do so by calculating

[3] A height of 13 makes a maximum sentence length of $2^{13-1} = 2^{12} = 4096$ words.

Grammar	Sentiment	Sentence
BASE	-2	(and badly should doesn't..
	-1	doesn't of the yankees..
	0	who is the the game,.
	1	is the the united states..
	2	is the best players..
LEX	-2	is a rhyme ... mahi mahi, and, I not quote Bunny.
	-1	Dumpster unpire are the villans.
	0	Derogatory big names symbols wider
	1	New england has been playful, and infrequent human.
	2	That's a huge award – having get fined!
RR	-2	he is very awkward, and to any ridiculous reason.
	-1	the malfeasance underscores the the widespread belief.
	0	the programs serve the purposes.
	1	McIlroy is a courageous competitor.
	2	The urgent service's a grand idea.

Table 1: Responses generated by the system with the different grammars and sentiment levels.

Grammar	Avg. LM Score		Avg. LM Score per word		Complete Sentences (%)	Sentiment Agreement / Polarity (%)	Avg. Length (words)
	Mean	CI	Mean	CI			
BASE	-79.7	±0.054	-8.9	±0.007	20.1	13.3 / 41.8	9.5
LEX	-73.7	±0.016	-6.5	±0.002	67.3	**44.6 / 63.9**	12.3
RR	**-51.8**	±0.011	**-5.6**	±0.001	**95.7**	43.8 / 61.0	9.6
HUMAN	-50.1	±0.000	-5.4	±0.000	N/A	N/A	10.3

Table 2: Mean and 95% Confidence Interval (CI) of language model scores, and measures of compactness and sentiment agreement. The last row, *HUMAN* refers to the collected human responses.

Topic Agreement, a measure that, given a trained topic model, determines how close the topic distribution of the input document and that of the generated response are. We use L2 to calculate the distance between the inferred topic distribution vectors. We focus here on relevance testing for the RR grammar, which gave superior LM scores. In this test we use two generators – RR generator as defined above, and RRTM generator that uses the scoring scheme of Equation (5) to select a start rule deriving the root lexical item. Example sentences of each generator are presented in Table 3.

Empirical Results The results of the two generators and their average distance from the topic distribution of the input document are presented in Table 4. Here we see that the generator using topic models for selecting start rules (RRTM) gets topic distribution that is closer to the input document's topic distribution. The last row, HUMAN, calculates the distance between the topic distributions in the documents and their human responses from the collected corpus. The fact that RRTM outperforms HUMAN is not necessarily surprising, as sentences in human responses are typically from longer paragraphs where some sentences are more generic, used as connectives, interjections, etc.

Grammar	Sentiment	Sentence
RR	-2	they deserve it, but I is fear.
	-1	the saga is correct.
	0	the indirect penalty?
	1	the job is correct.
	2	a salaries excels.
RRTM	-2	Unfortunately, they remind that to participate in baseball.
	-1	the franchise would he made?
	0	Probably the LONG time .
	1	In a good addition, he is a good baseball player.
	2	the baseball game sublime.

Table 3: Responses generated by the system using emission probabilities and topic models for the start rule selection.

Generator	Mean	CI
RR	0.473	$\pm$ 0.003
RRTM	0.424	$\pm$ 0.003
HUMAN	0.429	$\pm$ 0.000

Table 4: Mean and 95% Confidence Interval (CI) for generators with / without topic models scores (RRTM / RR respectively). The last row, *HUMAN* refers to the collected human responses.

6.3 Human Surveys

Goal and Procedure. We evaluate human-likeness of the generated responses by collecting data via an online survey on Amazon Mechanical Turk. In the survey, participants were asked to judge whether generated sentences were written by a human or a computer. The participants were screened to have a good level of English and reside in the US. Each survey comprised of 50 randomly ordered trials. In each trial the participant was shown a response. The task was to categorize each response on a 7-point scale with labels 'Certainly human/computer', 'Probably human/computer', 'Maybe human/computer' and 'Unsure'. In 50 trials the participant was exposed to 3-4 sentences for each grammar/sentiment combination.

Empirical Results. Average human-likeness ratings (scale 1–7) are presented in Table 5. Here, we see that sentences generated by the lexicalized grammar were perceived as most human-like. This result is in contrast with the automatic evaluation. Such a discrepancy need not be very surprising, as noted by others before (Belz and Reiter, 2006). Cagan et al. (2014) show that there are extra-grammatical factors affecting human-likeness, e.g. world knowledge. We hypothesise that the LEX grammar, which relies heavily on lexical co-occurrences frequencies, is better at replicating world knowledge and idiomatic phrases thus judged as more human.

Grammar	Mean	CI
BASE	2.4561	$\pm$ 0.004
LEX	4.1681	$\pm$ 0.004
RR	3.7278	$\pm$ 0.004

Table 5: Mean and 95% Confidence Interval (CI) for human-likeness ratings (scaling 1:low–7:high).

Factor	b	Std. Error	z-value	$P(>\|z\|)$
G-LEX	2.90	0.189	15.32	<.00001
G-RR	2.33	0.164	14.20	<.00001
SENT	0.17	0.074	2.32	.020
NWORD	-1.60	0.107	-14.95	<.00001
POS	0.21	0.036	5.97	<.00001
G-LEX $\times$ SENT	-0.18	0.095	-1.91	.056
G-RR $\times$ SENT	0.44	0.096	4.53	<.00001
G-LEX $\times$ NWORD	1.31	0.117	11.16	<.00001
G-RR $\times$ NWORD	1.35	0.138	9.80	<.00001
NWORD $\times$ POS	0.10	0.037	2.81	.005

Table 6: Regression analysis of the human survey.

In a qualitative inspection on a sample of the results we could verify that the LEX grammar tends to replicate idiomatic sequences while the RR grammar generates novel phrases in a more compositional fashion. Grammaticality is not hindered by it, but apparently human-likeness is.

We also run an ordinal mixed-effects regression, which is an appropriate way to analyse discrete rating data. Regression model predictors were Grammar (G), sentiment level (SENT), response length (NWORD), position of response in rating session (POS), and all two-way interactions between these. Quantitative predictors were standardized and non-significant ($p > .05$) interactions were dropped from the fitted model. By-participant random intercepts and slopes of G and SENT were included as random effects.

Table 6 displays the fitted model fixed effects, with BASE grammar as the reference level. Consistent with Table 5, we see that LEX and RR score significantly higher on human likeness than BASE. These effects are modulated by sentiment: more positive sentiment makes BASE and RR more human-like (respectively: $b = 0.17$ and $b = 0.44$) whereas the LEX grammar becomes less human like (although this effect is only marginally significant: $b = -.18$). In addition, these effects are also modulated by sentence length in #words – longer sentences make BASE less human-like ($b = -1.60$) but RR and LEX more human-like (respectively: $b = 1.31$ and $b = 1.35$)

Importantly, there is a weak but significant *positive* effect of position ($b = 0.21$), indicating that human-likeness ratings increase over the course of a rating session. This effect does not depend on the grammar, but is somewhat stronger for longer

sentences ($b = 0.10$). The position effect contrasts markedly with the decrease of human-likeness ratings that (Cagan et al., 2014) ascribed to a learning effect: there, raters noticed the repetitive structure and took this to be a sign that the utterances were machine generated. The fact that we find no such effect means that our grammars successfully avoided such repetitiveness.

7 Related and Future Work

NLG is often cast as a *concept-to-text* (C2T) challenge, where a structured record is transformed into an utterance expressing its content. C2T is usually addressed using template-based (Becker, 2002) or data-driven (Konstas and Lapata, 2013; Yuan et al., 2015) approaches. In particular, researchers explored data-driven grammar-based approaches (Cahill and van Genabith, 2006), often assuming a custom grammar (Konstas and Lapata, 2013) or a closed-domain approach (DeVault et al., 2008). ONLG in contrast is set in an open domain, and expresses multiple dimensions (grammaticality, sentiment, topic).

In the context of social media, generating responses to tweets has been cast as a sequence-to-sequence (seq2seq) transduction problem, and has been addressed using statistical machine translation (SMT) methods (Ritter et al., 2011; Hasegawa et al., 2013). In this seq2seq setup, moods and sentiments expressed in the past are replicated or reused, but these responses do not target particular topics and are not driven by a concrete user agenda. An exception is a recent work by Li et al. (2016), exploring a persona-based conversational model, and Xu et al. (2016) who encode loose structured knowledge to condition the generation on. These studies present a stepping stone towards full-fledge neural ONLG architectures with some control over the user characteristics.

The surge of interest in neural network generation architectures has spawned the development of seq2seq models based on encoder-decoder setup (Sordoni et al. (2015); Li et al. (2016, 2017) and references therein). These architectures require a very large dataset to train on. In the future we aim to extend our dataset and explore neural network architectures for ONLG that can encode a user-agenda, a document, and possibly stylistic choices (Biber and Conrad, 2009; Reiter and Williams, 2010) — in the hope of yielding more diverse, relevant and coherent responses to online content.

8 Conclusion

We approached ONLG from a data-driven perspective, aiming to overcome the shortcomings of previous template-based approaches. Our contribution is threefold: (i) we designed three types of broad-coverage grammars appropriate for the task, (ii) we developed a new enriched data-set for inducing the grammars, and (iii) we empirically demonstrated the strengths of the LEX and RR grammars for generation, as well as the overall usefulness of sentiment and topic models incorporated into the syntactic derivation. Our results show that the proposed grammar-based architecture indeed avoids the repetitiveness and learning effects observed in the template-based ONLG.

To the best of our knowledge, this is the first data-driven agenda-driven baseline for ONLG, and we believe it can be further improved. Some future avenues for investigation include improving the relevance and human-likeness results by improving the automatic parses quality, acquiring more complex templates via abstract grammars, and experimenting with more sophisticated scoring functions for reranking. With the emergence of deep learning, we further embrace the opportunity to combine the *sequence-to-sequence* modeling view explored so far with conditioning generation on speakers agendas and user profiles, pushing the envelope of opinionated generation further. Finally, we believe that future work should be evaluated *in situ*, to examine if, and to what extent, the generated responses participate in and affect the discourse (feed) in social media.

References

Tilman Becker. 2002. Practical, template-based natural language generation with TAG. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6)*.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of NLG systems. In *Proceeding of EACL'06*. pages 313–320.

D. Biber and S. Conrad. 2009. *Register, Genre, and Style*. Cambridge Textbooks in Linguistics. Cambridge University Press. https://books.google.de/books?id=0HUhombmOJUC.

Tomer Cagan, Stefan L. Frank, and Reut Tsarfaty. 2014. Generating subjective responses to opinionated articles in social media: An agenda-driven architecture and a Turing-like test. In *Proceedings of the Joint Workshop on Social Dynamics and*

Personal Attributes in Social Media. Association for Computational Linguistics, pages 58–67. http://www.aclweb.org/anthology/W/W14/W14-2708.

Aoife Cahill and Josef van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL-44, pages 1033–1040. https://doi.org/10.3115/1220175.1220305.

Eugene Charniak. 1995. Parsing with context-free grammars and word statistics. Technical report, Providence, RI, USA.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics* 29(4):589–637. https://doi.org/10.1162/089120103322753356.

David DeVault, David Traum, and Ron Artstein. 2008. Practical grammar-based NLG from examples. In *Proceedings of the Fifth International Natural Language Generation Conference*. Association for Computational Linguistics, Stroudsburg, PA, USA, INLG '08, pages 77–85. http://dl.acm.org/citation.cfm?id=1708322.1708338.

Donghui Feng, Erin Shaw, Jihie Kim, and Eduard Hovy. 2006. An intelligent discussion-bot for answering student queries in threaded discussions. In *Proceedings of Intelligent User Interface (IUI-2006)*. pages 171–177.

Michael Haenlein and Andreas M. Kaplan. 2009. Flagship brand stores within virtual worlds: The impact of virtual store exposure on real-life attitude toward the brand and purchase intent. *Recherche et Applications en Marketing (English Edition)* 24(3):57–79. https://doi.org/10.1177/205157070902400303.

Takayuki Hasegawa, Nobuhiro Kaji, Naoki Yoshinaga, and Masashi Toyoda. 2013. Predicting and eliciting addressee's emotion in online dialogue. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 964–972. http://www.aclweb.org/anthology/P13-1095.

Philip N. Howard, Aiden Duffy, Deen Freelon, Muzammil Hussain, Will Mari, and Marwa Mazaid. 2011. Opening closed regimes: What was the role of social media during the Arab spring? *Project on Information Technology and Political Islam.* http://pitpi.org/index.php/2011/09/11/opening-closed-regimes-what-was-the-role-of-social-media-during-the-arab-spring/.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research* 48:305–346.

Wiebke Lamer. 2012. Twitter and tyrants: New media and its effects on sovereignty in the Middle East. *Arab Media and Society* http://www.arabmediasociety.com/?article=798.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *CoRR* abs/1603.06155. http://arxiv.org/abs/1603.06155.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *CoRR* abs/1701.06547. http://arxiv.org/abs/1701.06547.

Paul Mah. 2012. Tools to automate your customer service response on social media. Visited August 2013. http://www.itbusinessedge.com/blogs/smb-tech/tools-to-automate-your-customer-service-response-on-social-media.html.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. `http://www.cs.umass.edu/~mccallum/mallet`.

Microsoft. 2011. Microsoft cognitive services. https://www.microsoft.com/cognitive-services/en-us/web-language-model-api.

Kyoshi Mori, Adam Jatowt, and Mitsuru Ishizuka. 2003. Enhancing conversational flexibility in multimodal interactions with embodied lifelike agent. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*. ACM, New York, NY, USA, IUI '03, pages 270–272. https://doi.org/10.1145/604045.604096.

Jeremiah Owyang. 2012. Brands Start Automating Social Media Responses on Facebook and Twitter. Visited August 2013. http://techcrunch.com/2012/06/07/brands-start-automating-social-media-responses-on-facebook-and-twitter/.

Erik Qualman. 2012. *Socialnomics: How social media transforms the way we live and do business*. John Wiley & Sons, Hoboken, NJ, USA, 2nd edition. https://books.google.co.il/books?id=yAqD19i2U0UC.

D. Raj Reddy. 1977. Speech understanding systems: summary of results of the five-year research effort at Carnegie-Mellon University. Technical report, Carnegie-Mellon University.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering* 3(1):57–87. https://doi.org/10.1017/S1351324997001502.

Ehud Reiter and Sandra Williams. 2010. Generating texts in different styles. In Shlomo Argamon, Kevin Burns, and Shlomo Dubnov, editors, *The Structure of Style - Algorithmic Approaches to Understanding Manner and Meaning.*, Springer, pages 59–75.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 583–593. http://dl.acm.org/citation.cfm?id=2145432.2145500.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, pages 1631–1642.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 196–205. http://www.aclweb.org/anthology/N15-1020.

Reut Tsarfaty. 2010. *Relational-Realizational Parsing*. Ph.D. thesis, Institute for Logic, Language and Computation, University of Amsterdam.

Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2011. Evaluating dependency parsing: Robust and Heuristics-Free Cross-Annotation evaluation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. pages 385–396. http://www.aclweb.org/anthology/D11-1036.

Reut Tsarfaty and Khalil Sima'an. 2008. Relational-realizational parsing. In *Proceedings of the 22Nd International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 889–896. http://dl.acm.org/citation.cfm?id=1599081.1599193.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2016. Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling. *CoRR* abs/1605.05110. http://arxiv.org/abs/1605.05110.

Caixia Yuan, Xiaojie Wang, and Qianhui He. 2015. *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, Association for Computational Linguistics, chapter Response Generation in Dialogue Using a Tailored PCFG Parser, pages 81–85. http://aclweb.org/anthology/W15-4713.

Learning to Ask: Neural Question Generation for Reading Comprehension

Xinya Du[1] **Junru Shao**[2] **Claire Cardie**[1]
[1]Department of Computer Science, Cornell University
[2]Zhiyuan College, Shanghai Jiao Tong University
{xdu, cardie}@cs.cornell.edu yz_sjr@sjtu.edu.cn

Abstract

We study automatic question generation for sentences from text passages in reading comprehension. We introduce an attention-based sequence learning model for the task and investigate the effect of encoding sentence- vs. paragraph-level information. In contrast to all previous work, our model does not rely on hand-crafted rules or a sophisticated NLP pipeline; it is instead trainable end-to-end via sequence-to-sequence learning. Automatic evaluation results show that our system significantly outperforms the state-of-the-art rule-based system. In human evaluations, questions generated by our system are also rated as being more natural (*i.e.*, grammaticality, fluency) and as more difficult to answer (in terms of syntactic and lexical divergence from the original text and reasoning needed to answer).

1 Introduction

Question generation (QG) aims to create natural questions from a given a sentence or paragraph. One key application of question generation is in the area of education — to generate questions for reading comprehension materials (Heilman and Smith, 2010). Figure 1, for example, shows three manually generated questions that test a user's understanding of the associated text passage. Question generation systems can also be deployed as chatbot components (*e.g.*, asking questions to start a conversation or to request feedback (Mostafazadeh et al., 2016)) or, arguably, as a clinical tool for evaluating or improving mental health (Weizenbaum, 1966; Colby et al., 1971).

In addition to the above applications, question generation systems can aid in the development of

Sentence:
Oxygen is used in cellular respiration and released by photosynthesis, which uses the energy of sunlight to produce oxygen from water.

Questions:
– What life process produces oxygen in the presence of light?
photosynthesis
– Photosynthesis uses which energy to form oxygen from water?
sunlight
– From what does photosynthesis get oxygen?
water

Figure 1: Sample sentence from the second paragraph of the article *Oxygen*, along with the natural questions and their answers.

annotated data sets for natural language processing (NLP) research in reading comprehension and question answering. Indeed the creation of such datasets, *e.g.*, SQuAD (Rajpurkar et al., 2016) and MS MARCO (Nguyen et al., 2016), has spurred research in these areas.

For the most part, question generation has been tackled in the past via rule-based approaches (*e.g.*, Mitkov and Ha (2003); Rus et al. (2010). The success of these approaches hinges critically on the existence of well-designed rules for declarative-to-interrogative sentence transformation, typically based on deep linguistic knowledge.

To improve over a purely rule-based system, Heilman and Smith (2010) introduced an overgenerate-and-rank approach that generates multiple questions from an input sentence using a rule-based approach and then ranks them using a supervised learning-based ranker. Although the ranking algorithm helps to produce more ac-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1342–1352
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1123

ceptable questions, it relies heavily on a manually crafted feature set, and the questions generated often overlap word for word with the tokens in the input sentence, making them very easy to answer.

Vanderwende (2008) point out that learning to ask good questions is an important task in NLP research in its own right, and should consist of more than the syntactic transformation of a declarative sentence. In particular, a natural sounding question often compresses the sentence on which it is based (*e.g.*, question 3 in Figure 1), uses synonyms for terms in the passage (*e.g.*, "form" for "produce" in question 2 and "get" for "produce" in question 3), or refers to entities from preceding sentences or clauses (*e.g.*, the use of "photosynthesis" in question 2). Othertimes, world knowledge is employed to produce a good question (*e.g.*, identifying "photosynthesis" as a "life process" in question 1). In short, constructing natural questions of reasonable difficulty would seem to require an *abstractive* approach that can produce fluent phrasings that do not exactly match the text from which they were drawn.

As a result, and in contrast to all previous work, we propose here to frame the task of question generation as a sequence-to-sequence learning problem that directly maps a sentence from a text passage to a question. Importantly, our approach is fully data-driven in that it requires no manually generated rules.

More specifically, inspired by the recent success in neural machine translation (Sutskever et al., 2014; Bahdanau et al., 2015), summarization (Rush et al., 2015; Iyer et al., 2016), and image caption generation (Xu et al., 2015), we tackle question generation using a conditional neural language model with a global attention mechanism (Luong et al., 2015a). We investigate several variations of this model, including one that takes into account paragraph- rather than sentence-level information from the reading passage as well as other variations that determine the importance of pre-trained vs. learned word embeddings.

In evaluations on the SQuAD dataset (Rajpurkar et al., 2016) using three automatic evaluation metrics, we find that our system significantly outperforms a collection of strong baselines, including an information retrieval-based system (Robertson and Walker, 1994), a statistical machine translation approach (Koehn et al., 2007), and the overgenerate-and-rank approach of Heilman and Smith (2010). Human evaluations also rated our generated questions as more grammatical, fluent, and challenging (in terms of syntactic divergence from the original reading passage and reasoning needed to answer) than the state-of-the-art Heilman and Smith (2010) system.

In the sections below we discuss related work (Section 2), specify the task definition (Section 3) and describe our neural sequence learning based models (Section 4). We explain the experimental setup in Section 5. Lastly, we present the evaluation results as well as a detailed analysis.

2 Related Work

Reading Comprehension is a challenging task for machines, requiring both understanding of natural language and knowledge of the world (Rajpurkar et al., 2016). Recently many new datasets have been released and in most of these datasets, the questions are generated in a synthetic way. For example, bAbI (Weston et al., 2016) is a fully synthetic dataset featuring 20 different tasks. Hermann et al. (2015) released a corpus of cloze style questions by replacing entities with placeholders in abstractive summaries of CNN/Daily Mail news articles. Chen et al. (2016) claim that the CNN/Daily Mail dataset is easier than previously thought, and their system almost reaches the ceiling performance. Richardson et al. (2013) curated MCTest, in which crowdworker questions are paired with four answer choices. Although MCTest contains challenging natural questions, it is too small for training data-demanding question answering models.

Recently, Rajpurkar et al. (2016) released the Stanford Question Answering Dataset[1] (SQuAD), which overcomes the aforementioned small size and (semi-)synthetic issues. The questions are posed by crowd workers and are of relatively high quality. We use SQuAD in our work, and similarly, we focus on the generation of natural questions for reading comprehension materials, albeit via automatic means.

Question Generation has attracted the attention of the natural language generation (NLG) community in recent years, since the work of Rus et al. (2010).

Most work tackles the task with a rule-based approach. Generally, they first transform the input sentence into its syntactic representation, which

[1] https://stanford-qa.com

they then use to generate an interrogative sentence. A lot of research has focused on first manually constructing question templates, and then applying them to generate questions (Mostow and Chen, 2009; Lindberg et al., 2013; Mazidi and Nielsen, 2014). Labutov et al. (2015) use crowdsourcing to collect a set of templates and then rank the relevant templates for the text of another domain. Generally, the rule-based approaches make use of the syntactic roles of words, but not their semantic roles.

Heilman and Smith (2010) introduce an overgenerate-and-rank approach: their system first overgenerates questions and then ranks them. Although they incorporate learning to rank, their system's performance still depends critically on the manually constructed generating rules. Mostafazadeh et al. (2016) introduce visual question generation task, to explore the deep connection between language and vision. Serban et al. (2016) propose generating simple factoid questions from logic triple (subject, relation, object). Their task tackles mapping from structured representation to natural language text, and their generated questions are consistent in terms of format and diverge much less than ours.

To our knowledge, none of the previous works has framed QG for reading comprehension in an end-to-end fashion, and nor have them used deep sequence-to-sequence learning approach to generate questions.

3 Task Definition

In this section, we define the question generation task. Given an input sentence $\mathbf{x}$, our goal is to generate a natural question $\mathbf{y}$ related to information in the sentence, $\mathbf{y}$ can be a sequence of an arbitrary length: $[y_1, ..., y_{|\mathbf{y}|}]$. Suppose the length of the input sentence is M, $\mathbf{x}$ could then be represented as a sequence of tokens $[x_1, ..., x_M]$. The QG task is defined as finding $\overline{\mathbf{y}}$, such that:

$$\overline{\mathbf{y}} = \arg\max_{\mathbf{y}} P\left(\mathbf{y}|\mathbf{x}\right) \tag{1}$$

where $P\left(\mathbf{y}|\mathbf{x}\right)$ is the conditional log-likelihood of the predicted question sequence $\mathbf{y}$, given the input $\mathbf{x}$. In section 4.1, we will elaborate on the global attention mechanism for modeling $P\left(\mathbf{y}|\mathbf{x}\right)$.

4 Model

Our model is partially inspired by the way in which a human would solve the task. To ask a natural question, people usually pay attention to certain parts of the input sentence, as well as associating context information from the paragraph. We model the conditional probability using RNN encoder-decoder architecture (Bahdanau et al., 2015; Cho et al., 2014), and adopt the global attention mechanism (Luong et al., 2015a) to make the model focus on certain elements of the input when generating each word during decoding.

Here, we investigate two variations of our models: one that only encodes the sentence and another that encodes both sentence and paragraph-level information.

4.1 Decoder

Similar to Sutskever et al. (2014) and Chopra et al. (2016), we factorize the the conditional in equation 1 into a product of word-level predictions:

$$P\left(\mathbf{y}|\mathbf{x}\right) = \prod_{t=1}^{|y|} P\left(y_t|\mathbf{x}, y_{<t}\right)$$

where probability of each y_t is predicted based on all the words that are generated previously (*i.e.*, $y_{<t}$), and input sentence $\mathbf{x}$.

More specifically,

$$P\left(y_t|\mathbf{x}, y_{<t}\right) = \text{softmax}\left(\mathbf{W}_s \text{tanh}\left(\mathbf{W}_t[\mathbf{h}_t; \mathbf{c}_t]\right)\right) \tag{2}$$

with $\mathbf{h}_t$ being the recurrent neural networks state variable at time step t, and $\mathbf{c}_t$ being the attention-based encoding of $\mathbf{x}$ at decoding time step t (Section 4.2). $\mathbf{W}_s$ and $\mathbf{W}_t$ are parameters to be learned.

$$\mathbf{h}_t = \text{LSTM}_1\left(y_{t-1}, \mathbf{h}_{t-1}\right) \tag{3}$$

here, LSTM is the Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997). It generates the new state $\mathbf{h}_t$, given the representation of previously generated word y_{t-1} (obtained from a word look-up table), and the previous state $\mathbf{h}_{t-1}$.

The initialization of the decoder's hidden state differentiates our basic model and the model that incorporates paragraph-level information.

For the basic model, it is initialized by the sentence's representation $\mathbf{s}$ obtained from the sentence encoder (Section 4.2). For our paragraph-level model, the *concatenation* of the sentence

encoder's output $\mathbf{s}$ and the paragraph encoder's output $\mathbf{s}'$ is used as the initialization of decoder hidden state. To be more specific, the architecture of our paragraph-level model is like a "Y"-shaped network which encodes both sentence- and paragraph-level information via two RNN branches and uses the concatenated representation for decoding the questions.

4.2 Encoder

The attention-based sentence encoder is used in both of our models, while the paragraph encoder is only used in the model that incorporates paragraph-level information.

Attention-based sentence encoder:

We use a bidirectional LSTM to encode the sentence,

$$\overrightarrow{\mathbf{b}_t} = \overrightarrow{\text{LSTM}_2}\left(x_t, \overrightarrow{\mathbf{b}_{t-1}}\right)$$
$$\overleftarrow{\mathbf{b}_t} = \overleftarrow{\text{LSTM}_2}\left(x_t, \overleftarrow{\mathbf{b}_{t+1}}\right)$$

where $\overrightarrow{\mathbf{b}_t}$ is the hidden state at time step t for the forward pass LSTM, $\overleftarrow{\mathbf{b}_t}$ for the backward pass.

To get *attention-based* encoding of $\mathbf{x}$ at decoding time step t, namely, $\mathbf{c}_t$, we first get the context dependent token representation by $\mathbf{b}_t = [\overrightarrow{\mathbf{b}_t}; \overleftarrow{\mathbf{b}_t}]$, then we take the weighted average over $\mathbf{b}_t$ ($t = 1, ..., |\mathbf{x}|$),

$$\mathbf{c}_t = \sum_{i=1,..,|\mathbf{x}|} a_{i,t}\mathbf{b}_i \tag{4}$$

The attention weight are calculated by the bilinear scoring function and softmax normalization,

$$a_{i,t} = \frac{\exp\left(\mathbf{h}_t^T \mathbf{W}_b \mathbf{b}_i\right)}{\sum_j \exp\left(\mathbf{h}_t^T \mathbf{W}_b \mathbf{b}_j\right)} \tag{5}$$

To get the sentence encoder's output for initialization of decoder hidden state, we concatenate last hidden state of the forward and backward pass, namely, $\mathbf{s} = [\overrightarrow{\mathbf{b}_{|\mathbf{x}|}}; \overleftarrow{\mathbf{b}_1}]$.

Paragraph encoder:

Given sentence $\mathbf{x}$, we want to encode the paragraph containing $\mathbf{x}$. Since in practice the paragraph is very long, we set a length threshold L, and truncate the paragraph at the L^{th} token. We call the truncated paragraph "paragraph" henceforth.

Denoting the paragraph as $\mathbf{z}$, we use another bidirectional LSTM to encode $\mathbf{z}$,

$$\overrightarrow{\mathbf{d}_t} = \overrightarrow{\text{LSTM}_3}\left(z_t, \overrightarrow{\mathbf{d}_{t-1}}\right)$$
$$\overleftarrow{\mathbf{d}_t} = \overleftarrow{\text{LSTM}_3}\left(z_t, \overleftarrow{\mathbf{d}_{t+1}}\right)$$

With the last hidden state of the forward and backward pass, we use the concatenation $[\overrightarrow{\mathbf{d}_{|\mathbf{z}|}}; \overleftarrow{\mathbf{d}_1}]$ as the paragraph encoder's output $\mathbf{s}'$.

4.3 Training and Inference

Giving a training corpus of sentence-question pairs: $\mathcal{S} = \left\{\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right)\right\}_{i=1}^{S}$, our models' training objective is to minimize the negative log-likelihood of the training data with respect to all the parameters, as denoted by θ,

$$\begin{aligned}\mathcal{L} &= -\sum_{i=1}^{S} \log P\left(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}; \theta\right) \\ &= -\sum_{i=1}^{S}\sum_{j=1}^{|\mathbf{y}^{(i)}|} \log P\left(y_j^{(i)}|\mathbf{x}^{(i)}, y_{<j}^{(i)}; \theta\right)\end{aligned}$$

Once the model is trained, we do inference using beam search. The beam search is parametrized by the possible paths number k.

As there could be many rare words in the input sentence that are not in the target side dictionary, during decoding many `UNK` tokens will be output. Thus, post-processing with the replacement of `UNK` is necessary. Unlike Luong et al. (2015b), we use a simpler replacing strategy for our task. For the decoded `UNK` token at time step t, we replace it with the token in the input sentence with the highest attention score, the index of which is $\arg\max_i a_{i,t}$.

5 Experimental Setup

We experiment with our neural question generation model on the processed SQuAD dataset. In this section, we firstly describe the corpus of the task. We then give implementation details of our neural generation model, the baselines to compare, and their experimental settings. Lastly, we introduce the evaluation methods by automatic metrics and human raters.

5.1 Dataset

With the SQuAD dataset (Rajpurkar et al., 2016), we extract sentences and pair them with the ques-

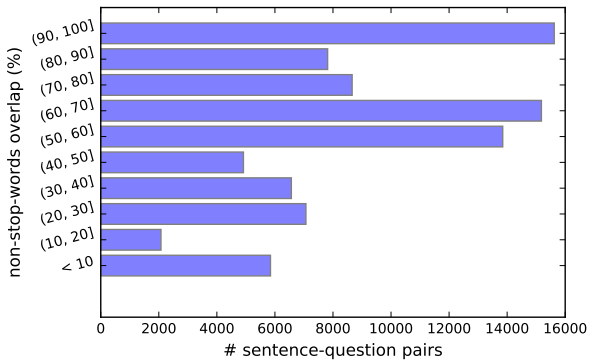


Figure 2: Overlap percentage of sentence-question pairs in training set. y-axis is # non-stop-words overlap with respect to the total # tokens in the question (a percentage); x-axis is # sentence-question pairs for a given overlap percentage range.

tions. We train our models with the sentence-question pairs. The dataset contains 536 articles with over 100k questions posed about the articles. The authors employ Amazon Mechanical Turks crowd-workers to create questions based on the Wikipedia articles. Workers are encouraged to use their own words without any copying phrases from the paragraph. Later, other crowd-workers are employed to provide answers to the questions. The answers are spans of tokens in the passage.

Since there is a hidden part of the original SQuAD that we do not have access to, we treat the accessible parts (∼90%) as the entire dataset henceforth.

We first run Stanford CoreNLP (Manning et al., 2014) for pre-processing: tokenization and sentence splitting. We then lower-case the entire dataset. With the offset of the answer to each question, we locate the sentence containing the answer and use it as the input sentence. In some cases (< 0.17% in training set), the answer spans two or more sentences, and we then use the concatenation of the sentences as the input "sentence".

Figure 2 shows the distribution of the token overlap percentage of the sentence-question pairs. Although most of the pairs have over 50% overlap rate, about 6.67% of the pairs have no non-stop-words in common, and this is mostly because of the answer offset error introduced during annotation. Therefore, we prune the training set based on the constraint: the sentence-question pair must have at least one non-stop-word in common. Lastly we add <SOS> to the beginning of the sentences, and <EOS> to the end of them.

# pairs (Train)	70484
# pairs (Dev)	10570
# pairs (Test)	11877
Sentence: avg. tokens	32.9
Question: avg. tokens	11.3
Avg. # questions per sentence	1.4

Table 1: Dataset (processed) statistics. Sentence average # tokens, question average # tokens, and average # questions per sentence statistics are from training set. These averages are close to the statistics on development set and test set.

We randomly divide the dataset at the article-level into a training set (80%), a development set (10%), and a test set (10%). We report results on the 10% test set.

Table 1 provides some statistics on the processed dataset: there are around 70k training samples, the sentences are around 30 tokens, and the questions are around 10 tokens on average. For each sentence, there might be multiple corresponding questions, and, on average, there are 1.4 questions for each sentence.

5.2 Implementation Details

We implement our models [2] in Torch7 [3] on top of the newly released OpenNMT system (Klein et al., 2017).

For the source side vocabulary $\mathcal{V}$, we only keep the 45k most frequent tokens (including <SOS>, <EOS> and placeholders). For the target side vocabulary $\mathcal{U}$, similarly, we keep the 28k most frequent tokens. All other tokens outside the vocabulary list are replaced by the UNK symbol. We choose word embedding of 300 dimensions and use the glove.840B.300d pre-trained embeddings (Pennington et al., 2014) for initialization. We fix the word representations during training.

We set the LSTM hidden unit size to 600 and set the number of layers of LSTMs to 2 in both the encoder and the decoder. Optimization is performed using stochastic gradient descent (SGD), with an initial learning rate of 1.0. We start halving the learning rate at epoch 8. The mini-batch size for the update is set at 64. Dropout with probability

[2] The code is available at https://github.com/xinyadu/nqg.

[3] http://torch.ch/

Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE_L
IR_{BM25}	5.18	0.91	0.28	0.12	4.57	9.16
$\text{IR}_{\text{Edit Distance}}$	18.28	5.48	2.26	1.06	7.73	20.77
MOSES+	15.61	3.64	1.00	0.30	10.47	17.82
DirectIn	31.71	21.18	15.11	11.20	14.95	22.47
H&S	38.50	22.80	15.52	11.18	15.95	30.98
Vanilla seq2seq	31.34	13.79	7.36	4.26	9.88	29.75
Our model (no pre-trained)	41.00	23.78	15.71	10.80	15.17	37.95
Our model (w/ pre-trained)	**43.09**	**25.96**	**17.50**	**12.28**	**16.62**	**39.75**
+ paragraph	42.54	25.33	16.98	11.86	16.28	39.37

Table 2: Automatic evaluation results of different systems by BLEU 1–4, METEOR and ROUGE_L. For a detailed explanation of the baseline systems, please refer to Section 5.3. The best performing system for each column is highlighted in boldface. Our system which encodes only sentence with pre-trained word embeddings achieves the best performance across all the metrics.

0.3 is applied between vertical LSTM stacks. We clip the gradient when the its norm exceeds 5.

All our models are trained on a single GPU. We run the training for up to 15 epochs, which takes approximately 2 hours. We select the model that achieves the lowest perplexity on the dev set.

During decoding, we do beam search with a beam size of 3. Decoding stops when every beam in the stack generates the `<EOS>` token.

All hyperparameters of our model are tuned using the development set. The results are reported on the test set.

5.3 Baselines

To prove the effectiveness of our system, we compare it to several competitive systems. Next, we briefly introduce their approaches and the experimental setting to run them for our problem. Their results are shown in Table 2.

IR stands for our information retrieval baselines. Similar to Rush et al. (2015), we implement the IR baselines to control memorizing questions from the training set. We use two metrics to calculate the distance between a question and the input sentence, *i.e.*, BM-25 (Robertson and Walker, 1994) and edit distance (Levenshtein, 1966). According to the metric, the system retrieves the training set to find the question with the highest score.

MOSES+ (Koehn et al., 2007) is a widely used phrase-based statistical machine translation system. Here, we treat sentences as source language text, we treat questions as target language text, and we perform the translation from sentences to questions. We train a tri-gram language model on target side texts with KenLM (Heafield et al., 2013), and tune the system with MERT on dev set. Performance results are reported on the test set.

DirectIn is an intuitive yet meaningful baseline in which the longest sub-sentence of the sentence is *directly* taken as the predicted question. [4] To split the sentence into sub-sentences, we use a set of splitters, *i.e.*, {"?", "!", ",", ".", ";"}.

H&S is the rule-based overgenerate-and-rank system that was mentioned in Section 2. When running the system, we set the parameter `just-wh` true (to restrict the output of the system to being only wh-questions) and set `max-length` equal to the longest sentence in the training set. We also set `downweight-pro` true, to down weight questions with unresolved pronouns so that they appear towards the end of the ranked list. For comparison with our systems, we take the top question in the ranked list.

Seq2seq (Sutskever et al., 2014) is a basic encoder-decoder sequence learning system for machine translation. We implement their model in Tensorflow. The input sequence is reversed before training or translating. Hyperparameters are tuned with dev set. We select the model with the lowest perplexity on the dev set.

[4] We also tried using the *entire* input sentence as the prediction output, but the performance is worse than taking sub-sentence as the prediction, across all the automatic metrics except for METEOR.

	Naturalness	Difficulty	Best %	Avg. rank
H&S	2.95	1.94	20.20	2.29
Ours	**3.36**	**3.03***	**38.38***	**1.94****
Human	3.91	2.63	66.42	1.46

Table 3: Human evaluation results for question generation. Naturalness and difficulty are rated on a 1–5 scale (5 for the best). Two-tailed t-test results are shown for our method compared to H&S (statistical significance is indicated with $^{*}(p < 0.005)$, $^{**}(p < 0.001)$).

5.4 Automatic Evaluation

We use the evaluation package released by Chen et al. (2015), which was originally used to score image captions. The package includes BLEU 1, BLEU 2, BLEU 3, BLEU 4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and ROUGE_L (Lin, 2004) evaluation scripts. BLEU measures the average n-gram precision on a set of reference sentences, with a penalty for overly short sentences. BLEU-n is BLEU score that uses up to n-grams for counting co-occurrences. METEOR is a recall-oriented metric, which calculates the similarity between generations and references by considering synonyms, stemming and paraphrases. ROUGE is commonly employed to evaluate n-grams recall of the summaries with gold-standard sentences as references. ROUGE_L (measured based on longest common subsequence) results are reported.

5.5 Human Evaluation

We also perform human evaluation studies to measure the quality of questions generated by our system and the H&S system. We consider two modalities: *naturalness*, which indicates the grammaticality and fluency; and *difficulty*, which measures the sentence-question syntactic divergence and the reasoning needed to answer the question. We randomly sampled 100 sentence-question pairs. We ask four professional English speakers to rate the pairs in terms of the modalities above on a 1–5 scale (5 for the best). We then ask the human raters to give a ranking of the questions according to the overall quality, with ties allowed.

6 Results and Analysis

Table 2 shows automatic metric evaluation results for our models and baselines. Our model which only encodes sentence-level information achieves

Sentence 1: the largest of these is the eldon square shopping centre , one of the largest city centre shopping complexes in the uk .
Human: what is one of the largest city center shopping complexes in the uk ?
H&S: what is the eldon square shopping centre one of ?
Ours: what is one of the largest city centers in the uk ?

Sentence 2: free oxygen first appeared in significant quantities during the paleoproterozoic eon -lrb- between 3.0 and 2.3 billion years ago -rrb- .
Human: during which eon did free oxygen begin appearing in quantity ?
H&S: what first appeared in significant quantities during the paleoproterozoic eon ?
Ours: how long ago did the paleoproterozoic exhibit ?

Sentence 3: inflammation is one of the first responses of the immune system to infection .
Human: what is one of the first responses the immune system has to infection ?
H&S: what is inflammation one of ?
Ours: what is one of the first objections of the immune system to infection ?

Sentence 4: tea , coffee , sisal , pyrethrum , corn , and wheat are grown in the fertile highlands , one of the most successful agricultural production regions in Africa.
Human: (1) where is the most successful agricultural prodcution regions ? (2) what is grown in the fertile highlands ?
H&S: what are grown in the fertile highlands in africa ?
Ours: what are the most successful agricultural production regions in africa ?

Sentence 5: as an example , income inequality did fall in the united states during its high school movement from 1910 to 1940 and thereafter .
Human: during what time period did income inequality decrease in the united states ?
H&S: where did income inequality do fall during its high school movement from 1910 to 1940 and thereafter as an example ?
Ours: when did income inequality fall in the us ?

Sentence 6: however , the rainforest still managed to thrive during these glacial periods , allowing for the survival and evolution of a broad diversity of species .
Human: did the rainforest managed to thrive during the glacial periods ?
H&S: what are treaties establishing european union ?
Ours: why do the birds still grow during glacial periods ?

Sentence 7: maududi founded the jamaat-e-islami party in 1941 and remained its leader until 1972.
Human: when did maududi found the jamaat-e-islami party ?
H&S: who did maududi remain until 1972 ?
Ours: when was the jamaat-e-islami party founded ?

Figure 3: Sample output questions generated by human (ground truth questions), our system and the H&S system.

Category	(%)	H&S			Ours			Ours + paragraph		
		BLEU-3	BLEU-4	METEOR	BLEU-3	BLEU-4	METEOR	BLEU-3	BLEU-4	METEOR
w/ sentence	70.23 (243)	20.64	15.81	16.76	**24.45**	**17.63**	17.82	24.01	16.39	**19.19**
w/ paragraph	19.65 (68)	6.34	< 0.01	10.74	3.76	< 0.01	11.59	**7.23**	**4.13**	**12.13**
All*	100 (346)	19.97	14.95	16.68	23.63	**16.85**	17.62	**24.68**	16.33	**19.61**

Table 4: An estimate of categories of questions of the processed dataset and per-category performance comparison of the systems. The estimate is based on our analysis of the 346 pairs from the dev set. Categories are decided by the information needed to generate the question. Bold numbers represent the best performing method for a given metric. *Here, we leave out performance results for "w/ article" category (2 samples, 0.58%) and "not askable" category (33 samples, 9.54%).

the best performance across all metrics. We note that IR performs poorly, indicating that memorizing the training set is not enough for the task. The baseline DirectIn performs pretty well on BLEU and METEOR, which is reasonable given the overlap statistics between the sentences and the questions (Figure 2). H&S system's performance is on a par with DirectIn's, as it basically performs syntactic change without paraphrasing, and the overlap rate is also high.

Looking at the performance of our three models, it's clear that adding the pre-trained embeddings generally helps. While encoding the paragraph causes the performance to drop a little, this makes sense because, apart from useful information, the paragraph also contains much noise.

Table 3 shows the results of the human evaluation. We see that our system outperforms H&S in all modalities. Our system is ranked best in 38.4% of the evaluations, with an average ranking of 1.94. An inter-rater agreement of Krippendorff's Alpha of 0.236 is achieved for the overall ranking. The results imply that our model can generate questions of better quality than the H&S system.

For our qualitative analysis, we examine the sample outputs and the visualization of the alignment between the input and the output. In Figure 3, we present sample questions generated by H&S and our best model. We see a large gap between our results and H&S's. For example, in the first sample, in which the focus should be put on "the largest." Our model successfully captures this information, while H&S only performs some syntactic transformation over the input without paraphrasing. However, outputs from our system are not always "perfect", for example, in pair 6, our system generates a question about the reason why birds still grow, but the *most related* question would be why many species still grow. But from a different perspective, our question is more challenging (readers need to understand that birds are one kind of species), which supports our system's performance listed in human evaluations (See Table 3). It would be interesting to further investigate how to interpret why certain irrelavant words are generated in the question. Figure 4 shows the attention weights ($\alpha_{i,t}$) for the input sentence when generating each token in the question. We see that the key words in the output ("introduced", "teletext", etc.) aligns well with those in the input sentence.

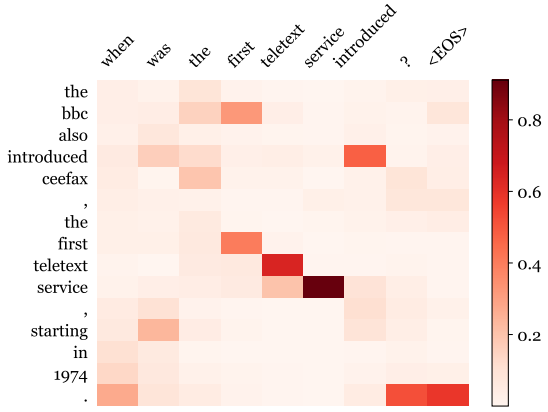


Figure 4: Heatmap of the attention weight matrix, which shows the soft alignment between the sentence (left) and the generated question (top).

Finally, we do a dataset analysis and fine-grained system performance analysis. We randomly sampled 346 sentence-question pairs from the dev set and label each pair with a category. [5] The four categories are determined by *how much* information is needed to ask the question. To be specific, "w/ sentence" means it only requires

[5]The IDs of the questions examined will be made available at https://github.com/xinyadu/nqg/blob/master/examined-question-ids.txt.

the sentence to ask the question; "w/ paragraph" means it takes other information in the paragraph to ask the question; "w/ article" is similar to "w/ paragraph"; and "not askable" means that world knowledge is needed to ask the question or there is mismatch of sentence and question caused by annotation error.

Table 4 shows the per-category performance of the systems. Our model which encodes paragraph information achieves the best performance on the questions of "w/ paragraph" category. This verifies the effectiveness of our paragraph-level model on the questions concerning information outside the sentence.

7 Conclusion and Future Work

We have presented a fully data-driven neural networks approach to automatic question generation for reading comprehension. We use an attention-based neural networks approach for the task and investigate the effect of encoding sentence- vs. paragraph-level information. Our best model achieves state-of-the-art performance in both automatic evaluations and human evaluations.

Here we point out several interesting future research directions. Currently, our paragraph-level model does not achieve best performance across all categories of questions. We would like to explore how to better use the paragraph-level information to improve the performance of QG system regarding questions of all categories. Besides this, it would also be interesting to consider to incorporate mechanisms for other language generation tasks (*e.g.*, copy mechanism for dialogue generation) in our model to further improve the quality of generated questions.

Acknowledgments

We thank the anonymous ACL reviewers, Kai Sun and Yao Cheng for their helpful suggestions. We thank Victoria Litvinova for her careful proofreading. We also thank Xanda Schofield, Wil Thomason, Hubert Lin and Junxian He for doing the human evaluations.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations Workshop (ICLR)*.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2358–2367. http://www.aclweb.org/anthology/P16-1223.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325* .

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 93–98. http://www.aclweb.org/anthology/N16-1012.

Kenneth Mark Colby, Sylvia Weber, and Franklin Dennis Hilf. 1971. Artificial paranoia. *Artificial Intelligence* 2(1):1–25. https://doi.org/10.1016/0004-3702(71)90002-6.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 376–380. http://www.aclweb.org/anthology/W14-3348.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified kneser-ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 690–696. http://www.aclweb.org/anthology/P13-2121.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 609–617. http://www.aclweb.org/anthology/N10-1086.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*. pages 1693–1701.

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

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2073–2083. http://www.aclweb.org/anthology/P16-1195.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *ArXiv e-prints* .

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 177–180. http://dl.acm.org/citation.cfm?id=1557769.1557821.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 889–898. http://www.aclweb.org/anthology/P15-1086.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*. volume 10, page 707.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Association for Computational Linguistics, Barcelona, Spain, pages 74–81. http://aclweb.org/anthology/W/W04/W04-1013.pdf.

David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. 2013. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*. Association for Computational Linguistics, Sofia, Bulgaria, pages 105–114. http://www.aclweb.org/anthology/W13-2114.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 11–19. http://www.aclweb.org/anthology/P15-1002.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny F., Steven B., and David M. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. http://www.aclweb.org/anthology/P14-5010.

Karen Mazidi and Rodney D. Nielsen. 2014. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 321–326. http://www.aclweb.org/anthology/P14-2053.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In Jill Burstein and Claudia Leacock, editors, *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*. pages 17–22. http://www.aclweb.org/anthology/W03-0203.pdf.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1802–1813. http://www.aclweb.org/anthology/P16-1170.

Jack Mostow and Wei Chen. 2009. Generating instruction automatically for the reading strategy of self-questioning. In *Proceedings of the 2nd Workshop on Question Generation (AIED 2009)*. pages 465–472.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine

reading comprehension dataset. *arXiv preprint arXiv:1611.09268* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Austin, Texas, pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 193–203. http://www.aclweb.org/anthology/D13-1020.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Springer-Verlag New York, Inc., New York, NY, USA, SIGIR '94, pages 232–241. http://dl.acm.org/citation.cfm?id=188490.188561.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 251–257. http://dl.acm.org/citation.cfm?id=1873738.1873777.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 379–389. http://aclweb.org/anthology/D15-1044.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 588–598. http://www.aclweb.org/anthology/P16-1056.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NIPS)*. pages 3104–3112.

Lucy Vanderwende. 2008. The importance of being important: Question generation. In *Proceedings of the 1st Workshop on the Question Generation Shared Task Evaluation Challenge, Arlington, VA*.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM* 9(1):36–45. https://doi.org/10.1145/365153.365168.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *International Conference on Learning Representations Workshop (ICLR)*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. volume 14, pages 77–81.

Joint Optimization of User-desired Content in Multi-document Summaries by Learning from User Feedback

Avinesh P.V.S and **Christian M. Meyer**
Research Training Group AIPHES and UKP Lab
Computer Science Department, Technische Universität Darmstadt
www.aiphes.tu-darmstadt.de, www.ukp.tu-darmstadt.de

Abstract

In this paper, we propose an extractive multi-document summarization (MDS) system using joint optimization and active learning for content selection grounded in user feedback. Our method interactively obtains user feedback to gradually improve the results of a state-of-the-art integer linear programming (ILP) framework for MDS. Our methods complement fully automatic methods in producing high-quality summaries with a minimum number of iterations and feedbacks. We conduct multiple simulation-based experiments and analyze the effect of feedback-based concept selection in the ILP setup in order to maximize the user-desired content in the summary.

1 Introduction

The task of producing summaries from a cluster of multiple topic-related documents has gained much attention during the Document Understanding Conference[1] (DUC) and the Text Analysis Conference[2] (TAC) series. Despite a lot of research in this area, it is still a major challenge to automatically produce summaries that are on par with human-written ones. To a large extent, this is due to the complexity of the task: a good summary must include the most relevant information, omit redundancy and irrelevant information, satisfy a length constraint, and be cohesive and grammatical. But an even bigger challenge is the high degree of subjectivity in content selection, as it can be seen in the small overlap of what is considered important by different users. Optimizing a system towards one single best summary that fits all users, as it is assumed by current state-of-the-art systems, is highly impractical and diminishes the usefulness of a system for real-world use cases.

In this paper, we propose an interactive concept-based model to assist users in creating a personalized summary based on their feedback. Our model employs integer linear programming (ILP) to maximize user-desired content selection while using a minimum amount of user feedback and iterations. In addition to the joint optimization framework using ILP, we explore pool-based active learning to further reduce the required feedback. Although there have been previous attempts to assist users in single-document summarization, no existing work tackles the problem of multi-document summaries using optimization techniques for user feedback. Additionally, most existing systems produce only a single, globally optimal solution. Instead, we put the human in the loop and create a personalized summary that better captures the users' needs and their different notions of importance.

Need for personalization. Table 1 shows the ROUGE scores (Lin, 2004) of multiple existing summarization systems, namely TF*IDF (Luhn, 1958), LexRank (Erkan and Radev, 2004), TextRank (Mihalcea and Tarau, 2004), LSA (Gong and Liu, 2001), KL-Greedy (Haghighi and Vanderwende, 2009), provided by the sumy package[3] and ICSI[4] (Gillick and Favre, 2009; Boudin et al., 2015), a strong state-of-the-art approach (Hong et al., 2014) in comparison to the extractive upper bound on DUC'04 and DBS. DUC'04 is an English dataset of abstractive summaries from ho-

[1] http://duc.nist.gov/
[2] http://www.nist.gov/tac/
[3] https://github.com/miso-belica/sumy
[4] https://github.com/boudinfl/sume

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1353–1363
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1124

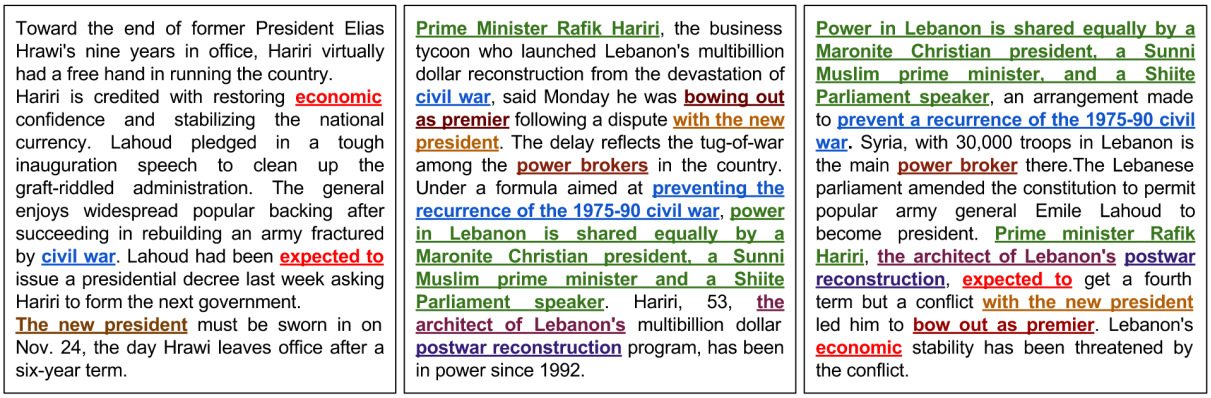


Figure 1: Lexical overlap of a reference summary (cluster D31043t in DUC 2004) with the summary produced by ICSI's state-of-the-art system (Boudin et al., 2015) and the extractive upper bound

	DUC'04			DBS		
Systems	R1	R2	SU4	R1	R2	SU4
TF*IDF	.292	.055	.086	.377	.144	.144
LexRank	.345	.070	.108	.434	.161	.180
TextRank	.306	.057	.096	.400	.167	.167
LSA	.294	.045	.081	.394	.154	.147
KL-Greedy	.336	.072	.104	.369	.133	.134
ICSI	.374	.090	.118	.452	.183	.190
UB	**.472**	**.210**	**.182**	**.848**	**.750**	**.532**

Table 1: ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-SU4 (SU4) scores of multiple systems compared to the extractive upper bound (UB)

mogenous news texts, whereas DBS (Benikova et al., 2016) is a German dataset of cohesive extracts from heterogeneous sources from the educational domain (see details in section 4.1). For each dataset, we compute an extractive upper bound (UB) by optimizing the sentence selection which maximizes ROUGE-2, i.e., the occurrence of bigrams as in the reference summary (Cao et al., 2016). Although some systems achieve state-of-the-art performance, their scores are still far from the extractive upper bound of individual reference summaries as shown in Figure 1. This is due to low inter-annotator agreement for concept selection: Zechner (2002) reports, for example, $\kappa = .13$ and Benikova et al. (2016) $\kappa = .23$. Most systems try to optimize for *all* reference summaries instead of personalizing, which we consider essential to capture user-desired content.

Need for user feedback. The goal of concept selection is finding the important information within a given set of source documents. Although existing summarization algorithms come up with a generic notion of importance, it is still far from the user-specific importance as shown in Figure 1. In contrast, humans can easily assess importance given a topic or a query. One way to achieve personalized summarization is thus by combining the advantages of both human feedback and the generic notion of importance built in a system. This allows users to interactively steer the summarization process and integrate their user-specific notion of importance.

Contributions. In this work, (1) we propose a novel ILP-based model using an interactive loop to create multi-document user-desired summaries, and (2) we develop models using pool-based active learning and joint optimization techniques to collect user feedback on identifying important concepts of a topic. In order to encourage the community to advance research and replicate our results, we provide our interactive summarizer implementation as open-source software.[5].

Our proposed method and our new interactive summarization framework can be used in multiple application scenarios: as an interactive annotation tool, which highlights important sentences for the annotators, as a journalistic writing aid that suggests important, user-adapted content from multiple source feeds (e.g., live blogs), and as a medical data analysis tool that suggests key information assisting a patient's personalized medical diagnosis.

The rest of the paper is structured as follows: In section 2, we discuss related work. Section 3

[5] https://github.com/UKPLab/acl2017-interactive_summarizer

introduces our computer-assisted summarization framework using the concept-based optimization. Section 4 describes our experiment data and setup. In section 5, we then discuss our results and analyze the performance of our models across different datasets. Finally, we conclude the paper in section 6 and discuss future work.

2 Related Work

Previous works related to our research address extractive summarization as a budgeted subset selection problem, computer-assisted approaches, and personalized summarization models.

Bugeted subset selection. Extractive summarization systems that compose a summary from a number of important sentences from the source documents are by far the most popular solution for MDS. This task can be modeled as a budgeted maximum coverage problem. Given a set of sentences in the document collection, the task is to maximize the coverage of the subset of sentences under a length constraint. The scoring function estimates the importance of the content units for a summary. Most previous works consider sentences as content units and try different scoring functions to optimize the summary.

One of the earliest systems by McDonald (2007) models a scoring function by simultaneously maximizing the relevance scores of the selected content units and minimizing their pairwise redundancy scores. They solve the global optimization problem using an ILP framework. Later, several state-of-the-art results employed an ILP to maximize the number of relevant concepts in the created summary: Gillick and Favre (2009) use an ILP with bigrams as concepts and hand-coded deletion rules for compression. Berg-Kirkpatrick et al. (2011) combine grammatical features relating to the parse tree and use a maximum-margin SVM trained on annotated gold-standard compressions. Woodsend and Lapata (2012) jointly optimize content selection and surface realization, Li et al. (2013) estimate the weights of the concepts using supervised methods, and Boudin et al. (2015) propose an approximation algorithm to achieve the optimal solution. Although these approaches achieve state-of-the-art performance, they produce only one globally optimal summary which is impractical for various users due to the subjectivity of the task. Therefore, we research interactive computer-assisted approaches in order to produce personalized summaries.

Computer-assisted summarization. The majority of the existing computer-assisted summarization tools (Craven, 2000; Narita et al., 2002; Orăsan et al., 2003; Orăsan and Hasler, 2006) present important elements of a document to the user. Creating a summary then requires the human to cut, paste, and reorganize the important elements in order to formulate a final text. The work by Orăsan and Hasler (2006) is closely related to ours, since they assist users in creating summaries for a source document based on the output of a given automatic summarization system. However, their system is neither interactive nor does it consider the user's feedback in any way. Instead, they suggest the output of the state-of-the-art (single-document) summarization method as a summary draft and ask the user to construct the summary without further interaction.

Personalized summarization. While most previous work focuses on generic summaries, there have been a few attempts to take a user's preferences into account. The study by Berkovsky et al. (2008) shows that users prefer personalized summaries that precisely reflect their interests. These interests are typically modeled with the help of a query (Park and An, 2010) or keyword annotations reflecting the user's opinions (Zhang et al., 2003).

In another strand of research, Díaz and Gervás (2007) create user models based on social tagging and Hu et al. (2012) rank sentences by combining informativeness scores with a user's interests based on fuzzy clustering of social tags. Extending the use of social content, another recent work showed how personalized review summaries (Poussevin et al., 2015) can be useful in recommender systems beyond rating predictions. Although these approaches show that personalized summaries are more useful than generic summaries, they do not attempt to iteratively refine a summary in an interactive user–system dialog.

3 Approach

The goal of our work is maximizing the user-desired content in a summary within a minimum number of iterations. To this end, we propose an interactive loop that alternates the automatic creation of a summary and the acquisition of user feedback to refine the next iteration's summary.

3.1 Summary Creation

Our starting point is the concept-based ILP summarization framework by Boudin et al. (2015). Let C be the set of concepts in a given set of source documents D, c_i the presence of the concept i in the resulting summary, w_i a concept's weight, ℓ_j the length of sentence j, s_j the presence of sentence j in the summary, and Occ_{ij} the occurrence of concept i in sentence j. Based on these definitions, we formulate the following ILP:

$$\max \quad \sum_i w_i c_i \tag{1}$$

$$\forall j. \quad \sum_j \ell_j s_j \leq L \tag{2}$$

$$\forall i, j. \quad \sum_j s_j Occ_{ij} \geq c_i \tag{3}$$

$$\forall i, j. \quad s_j Occ_{ij} \leq c_i \tag{4}$$

$$\forall i. \quad c_i \in \{0, 1\} \tag{5}$$

$$\forall j. \quad s_j \in \{0, 1\} \tag{6}$$

The objective function (1) maximizes the occurrence of concepts c_i in the summary based on their weights w_i. The constraint formalized in (2) ensures that the summary length is restricted to a maximum length L, (3) ensures the selection of all concepts in a sentence s_j if s_j has been selected for the summary. Constraint (4) ensures that a concept is only selected if it is present in at least one of the selected sentences.

The two key factors for the performance of this ILP are defining the concept set C and a method to estimate the weights $w_i \in W$. Previous works have used word bigrams as concepts (Gillick and Favre, 2009; Li et al., 2013; Boudin et al., 2015) and either use document frequency (i.e. the number of source documents containing the concept) as weights (Woodsend and Lapata, 2012; Gillick and Favre, 2009) or estimate them using a supervised regression model (Li et al., 2013). For our implementation, we likewise use bigrams as concepts and document frequency as weights, as Boudin et al. (2015) report good results with this simple strategy. Our approach is, however, not limited to this setup, as our interactive approach allows for any definition of C and W, including potentially more sophisticated weight estimation methods, e.g., based on deep neural networks. In section 5.2, we additionally analyze how other notions of concepts can be integrated into our approach.

3.2 Interactive Summarization Loop

Algorithm 1 provides an overview of our interactive summarization approach. The system takes the set of source documents D as input, derives the set of concepts C, and initializes their weights W. In line 5, we start the interactive feedback loop iterating over $t = 0, \ldots, T$. We first create a summary S_t (line 6) by solving the ILP and then extract a set of concepts Q_t (line 7), for which we query the user in line 11 As the user feedback in the current time step, we use the concepts $I_t \subseteq Q_t$ that have been considered important by the user. For updating the weights W in line 12, we may use all feedback collected until the current time step t, i.e., $I_0^t = \bigcup_{j=0}^{t} I_j$ and the set of concepts $Q_0^t = \bigcup_{j=0}^{t} Q_j$ seen by the user (with $Q_0^{-1} = \emptyset$). If there are no more concepts to query (i.e., $Q_t = \emptyset$), we stop the iteration and return the personalized summary S_t.

Algorithm 1 Interactive summarizer

```
1:  procedure INTERACTIVESUMMARIZER()
2:      input: Documents D
3:      C ← extractConcepts(D)
4:      W ← conceptWeights(C)
5:      for t = 0...T do
6:          S_t ← getSummary(C, W)
7:          Q_t ← extractConcepts(S_t) − Q_0^{t−1}
8:          if Q_t = ∅ then
9:              return S_t
10:         else
11:             I_t ← obtainFeedback(S_t, Q_t)
12:             W ← updateWeights(W, I_0^t, Q_0^t)
13:         end if
14:     end for
15: end procedure
```

3.3 User Feedback Optimization

To optimize the summary creation based on user feedback, we iteratively change the concept weights in the objective function of the ILP setup. We define the following models:

Accept model (ACCEPT). This model presents the current summary S_t with highlighted concepts Q_t to a user and asks him/her to select all important concepts I_t. We assign the maximum weight MAX to all concepts in I_t and consider the remaining $Q_t - I_t$ as unimportant by setting their weight to 0 (see equation 7 and 8). The intuition

behind this baseline is that the modified scores cause the ILP to prefer the user-desired concepts while avoiding unimportant ones.

$$\forall i \in I_0^t. \quad w_i = MAX \tag{7}$$

$$\forall i \in Q_0^t - I_0^t. \quad w_i = 0 \tag{8}$$

Joint ILP with User Feedback (JOINT). The ACCEPT model fails in cases where the user could not accept concepts that never appear in one of the S_t summaries. To tackle this, in our JOINT model, we change the objective function of the ILP in order to create S_t by jointly optimizing importance and user feedback. We thus replace the equation (1) with:

$$\max \begin{cases} \sum_{i \notin Q_0^t} w_i c_i - \sum_{i \in Q_0^t} w_i c_i & \text{if } t \leq \tau \\ \sum_i w_i c_i & \text{if } t > \tau \end{cases} \tag{9}$$

Equation (9) maximizes the use of concepts for which we yet lack feedback ($i \notin Q_0^t$) and minimizes the use of concepts for which we already have feedback ($i \in Q_0^t$). In this JOINT model, we use an exploration phase $t = 0 \ldots \tau$ to collect the feedback, which terminates when the user does not return any important concepts (i.e., $I_t = \emptyset$). In the exploratory phase, the minus term in the equation 9 helps to reduce the score of the sentences whose concepts have received feedback already. In other words, it causes higher scores for sentences consisting of concepts which yet lack feedback. After the exploration step, we fall back to the original importance-based optimization function from equation (1).

Active learning with uncertainty sampling (AL). Our JOINT model explores well in terms of prioritizing the concepts which yet lack user feedback. However, it gives equal probabilities to all the unseen concepts. The AL model employs pool-based active learning (Kremer et al., 2014) during the exploration phase in order to prioritize concepts for which the model is most uncertain. We distinguish the unlabeled concept pool $C_u = \{\Phi(\tilde{x}_1), \Phi(\tilde{x}_2), ..., \Phi(\tilde{x}_N)\}$ and the labeled concept pool $C_\ell = \{(\Phi(x_1), y_1), (\Phi(x_2), y_2), \ldots, (\Phi(x_N), y_N)\}$, where each concept x_i is represented as a d-dimensional feature vector $\Phi(x_i) \in \mathbb{R}^d$. The labels $y_i \in \{-1, 1\}$ are 1 for all important concepts in I_0^t and -1 for all unimportant concepts in $Q_0^t - I_0^t$. Initially, the labeled concept pool C_ℓ is small or empty, whereas the unlabeled concept pool C_u is relatively large.

The learning algorithm is presented with a $C = C_\ell \cup C_u$ and is first called to learn a decision function $f^{(0)}\colon \mathbb{R}^d \rightarrow \mathbb{R}$, where the function $f^{(0)}(\Phi(\tilde{x}))$ is taken to predict the label of the input vector $\Phi(\tilde{x})$. Then, in each t^{th} iteration, where $t = 1, 2, \ldots, \tau$, the querying algorithm selects an instance of $\tilde{x}_t \in C_u$ for which the learning algorithm is least certain. Thus, our learning goal of active learning is to minimize the expected loss $\mathcal{L}$ (i.e., hinge loss) with limited querying opportunities to obtain a decision function $f^{(1)}, f^{(2)}, \ldots, f^{(\tau)}$ that can achieve low error rates:

$$\min \mathbb{E}_{(\Phi(x),y) \in C_\ell} \left[\mathcal{L}(f^{(t)}(\Phi(x)), y) \right] \tag{10}$$

As the learning algorithm, we use a support vector machine (SVM) with a linear kernel. To obtain the probability distribution over classes we use Platt's calibration (Platt, 1999), an effective approach for transforming classification models into a probability distribution. Equation (11) shows the probability estimates for $f^{(t)}$, where $f^{(t)}$ is the uncalibrated output of the SVM in the t^{th} iteration and A, B are scalar parameters that are learned by the calibration algorithm. The uncertainty scores are calculated as described in the equation (12) for all the concepts which lack feedback (C_u).

$$p(y \mid f^{(t)}) = \frac{1}{1 + \exp(Af^{(t)} + B)} \tag{11}$$

$$u_i = 1 - \max_{y \in \{-1,1\}} p(y \mid f^{(t)}) \tag{12}$$

For our AL model, we now change the objective function in order to create S_t by multiplying uncertainty scores u_i to the weights w_i. We thus replace the objective function from (9) with

$$\max \begin{cases} \sum_{i \notin Q_0^t} u_i w_i c_i & \text{if } t \leq \tau \\ \sum_i w_i c_i & \text{if } t > \tau \end{cases} \tag{13}$$

Active learning with positive sampling (AL+). One way to sample the unseen concepts is using uncertainty as in AL, but another way is to actively choose samples for which the learning algorithm predicts as a possible important concept. In AL+, we introduce the notion of certainty $(1 - u_i)$ for the positively predicted samples ($f^{(t)}(\Phi(\tilde{x}_i)) = 1$) in

Dataset	Lang	Topics	Summary type	Length
DBS	de	10	Coherent extracts	≈500 words
DUC'01	en	30	Abstracts	100 words
DUC'02	en	59	Abstracts	100 words
DUC'04	en	50	Abstracts	100 words

Table 2: Statistics of the MDS datasets used

the objective function (1) for producing S_t

$$\max \begin{cases} \sum_{i \notin Q_0^t} (1 - u_i) \ell_i w_i c_i & \text{if } t \leq \tau \\ \sum_i w_i c_i & \text{if } t > \tau \end{cases} \quad (14)$$

$$\text{where } \ell_i = \begin{cases} 0 & \text{if } f^{(t)}(\Phi(\tilde{x}_i)) = -1 \\ 1 & \text{if } f^{(t)}(\Phi(\tilde{x}_i)) = 1 \end{cases} \quad (15)$$

4 Experimental Setup

4.1 Data

For our experiments, we mainly focus on the DBS corpus, which is an MDS dataset of coherent extracts created from heterogeneous sources about multiple educational topics (Benikova et al., 2016). This corpus is well-suited for our evaluation setup, since we are able to easily simulate a user's feedback based on the overlap between generated and reference summary.

Additionally, we carry out experiments on the most commonly used evaluation corpora published by DUC/NIST from the generic multi-document summarization task carried out in DUC'01, DUC'02 and DUC'04. The documents are all from the news domain and are grouped into various topic clusters. Table 2 shows the properties of these corpora.

For evaluating the summaries against the reference summary we use ROUGE (Lin, 2004) with the parameters suggested by (Owczarzak et al., 2012) yielding high correlation with human judgments (i.e., with stemming and without stopword removal).[6] Since DBS summaries do not have a fixed length, we use a variable length parameter L for evaluation, where L denotes the length of the reference summary. All results are averaged across all topics and reference summaries.

4.2 Data Pre-processing and Features

To pre-process the datasets, we perform tokenization and stemming with NLTK (Loper and Bird, 2002) and constituency parsing with the Stanford parser (Klein and Manning, 2003) for English and German. The parse trees will be used in section 5.2 below to experiment with a syntactically motivated concept notion.

As a concept's feature representation Φ for our active learning setups AL and AL+, we use pre-trained word embeddings. We use the Google News embeddings with 300 dimensions by Mikolov et al. (2013) for English and the 100-dimensional news- and Wikipedia-based embeddings by Reimers et al. (2014) for German. Additionally, we add TF*IDF, number of stop words, presence of named entities, and word capitalization as features. Discrete features, such as part-of-speech tags, are mapped into the word representation via lookup tables.

4.3 Oracle-Based Simulation of User Feedback

The presence of a human in the loop typically demands for a user study based evaluation, but to collect sufficient data for various settings of our models would be too expensive. Therefore, we resort to an oracle-based approach, where the oracle is a system simulating the user by generating the feedback based on reference outputs. This idea has been widely used in the development of interactive systems (González-Rubio et al., 2012; Knowles and Koehn, 2016) for studying the problem and exhibiting solutions in a theoretical and controlled environment.

To simulate user feedback in our setting, we consider all concepts $I_t \subseteq Q_t$ from the system-suggested summary S_t as important if they are present in the reference summary. Let Ref be the set of concepts in the reference summary. In the t^{th} iteration, we return $I_t = Q_t \cap Ref$ as the simulated user feedback. Thus, the goal of our system is to reach the upper bound for a user's reference summary within a minimal number of iterations.

We limit our experiments to ten iterations, since it appears unrealistic that users are willing to participate in more feedback cycles. Petrie and Bevan (2009) even report only three to five iterations.

5 Results and Analysis

5.1 Methods

Table 3 shows the evaluation results of our four models. When evaluating a summarization system, it is common to report the mean ROUGE scores across clusters using all the reference summaries. However, since we aim at personalizing

[6] -n 4 -m -a -x -c 95 -r 1000 -f A -p 0.5 -t 0 -2 -4 -u

Datasets	ICSI			UB			ACCEPT			JOINT			AL			AL+		
	R1	R2	SU4	R1	R2	SU4	R1	R2	SU4	R1	R2	SU4	R1	R2	SU4	R1	R2	SU4
Concept Notion: Bigrams																		
DBS	.451	.183	.190	.848	.750	.532	.778	.654	.453	.815	.707	.484	**.833**	**.729**	.498	.828	.721	**.500**
DUC'04	.374	.090	.118	.470	.212	.185	.442	.176	.165	**.444**	**.180**	**.166**	.440	.178	.160	.427	.166	.154
DUC'02	.350	.085	.110	.474	.216	.187	.439	.178	.161	.444	.182	.165	**.448**	**.188**	.165	**.448**	.184	**.170**
DUC'01	.333	.073	.105	.450	.213	.181	.414	.171	.156	.418	.167	.149	**.435**	**.186**	**.163**	.426	.181	.158
Concept Notion: Content Phrases																		
DBS	.403	.135	.154	.848	.750	.532	.691	.531	.430	.742	.597	.419	**.776**	**.652**	**.448**	.767	.629	.440
DUC'04	.374	.090	.118	.470	.212	.185	.441	.176	.160	.441	.179	.162	**.444**	**.180**	**.162**	.422	.164	.150
DUC'02	.350	.085	.110	.474	.216	.187	.436	.181	.162	.444	.183	.165	**.446**	**.185**	**.168**	.442	.182	.162
DUC'01	.333	.073	.105	.450	.213	.181	.410	.165	.153	.417	.170	.156	**.433**	**.182**	**.161**	.420	.179	.154

Table 3: ROUGE-1 (R1), ROUGE-2 (R2) and ROUGE SU-4 (SU4) achieved by our models after the tenth iteration of the interactive loop in comparison to the upper bound and the basic ILP setup

Datasets	ACCEPT #F	JOINT #F	AL #F	AL+ #F
Concept Notion: Bigrams				
DBS	313	296	348	342
DUC'04	15	14	16	14
DUC'02	14	13	15	15
DUC'01	13	11	13	13
Concept Notion: Content Phrases				
DBS	110	114	133	145
DUC'04	8	9	10	10
DUC'02	7	7	8	6
DUC'01	7	7	8	6

Table 4: Average amount of user feedback (#F) considered by our models at the end of the tenth iteration of the interactive summarization loop

the summary for an individual user, we evaluate our models based on the mean ROUGE scores across clusters per reference summary. In Table 4, we additionally evaluate the models based on the amount of feedback (#F $= |I_0^T|$) taken by the oracles to converge to the upper bound within ten iterations.

To examine the system performance based on user feedback, we analyze our models' performance on multiple datasets. The results in Table 3 show that our idea of interactive multi-document summarization allows users to steer a general summary towards a personalized summary consistently across all datasets. From the results, we can see that the AL model starts from the concept-based ILP summarization and nearly reaches the upper bound for all the datasets within ten iterations. AL+ performs similar to AL in terms of ROUGE, but requires less feedback (compare Table 4). Furthermore, the ACCEPT and JOINT models get stuck in a local optimum due to the less exploratory nature of the models.

5.2 Concept Notion

Our interactive summarization approach is based on the scalable global concept-based model which uses bigrams as concepts. Thus, it is intuitive to use bigrams for collecting user feedback as well.[7] Although our models reach the upper bound when using bigram-based feedback, they require a significantly large number of iterations and much feedback to converge, as shown in Table 4.

To reduce the amount of feedback, we also consider content phrases to collect feedback. That is, syntactic chunks from the constituency parse trees consisting of non-function words (i.e., nouns, verbs, adjectives, and adverbs). For DBS being extractive dataset, we use bigrams and content phrases as concepts, both for the objective function in equation (1) and as feedback items, whereas for the DUC datasets, the concepts are always bigrams for both the feedback types (bigrams/content phrases). For DUC being abstractive, in the case of feedback given on content phrases, they are projected back to the bigrams to change the concept weights in order to have more overlap of simulated feedback. Table 4 shows feedbacks based on the content phrases reduces the number of feedbacks by a factor of 2. Furthermore, when content phrases are used as concepts for DBS, the performance of the models is lower compared to bigrams, as seen in Table 3.

5.3 Datasets

Figure 2 compares the ROUGE-2 scores and the amount of feedback used over time when applied to the DBS and the DUC'04 corpus. We can see from the figure that all models show an improvement of +.45 ROUGE-2 after merely 4 iterations

[7] We prune bigrams consisting of only functional words.

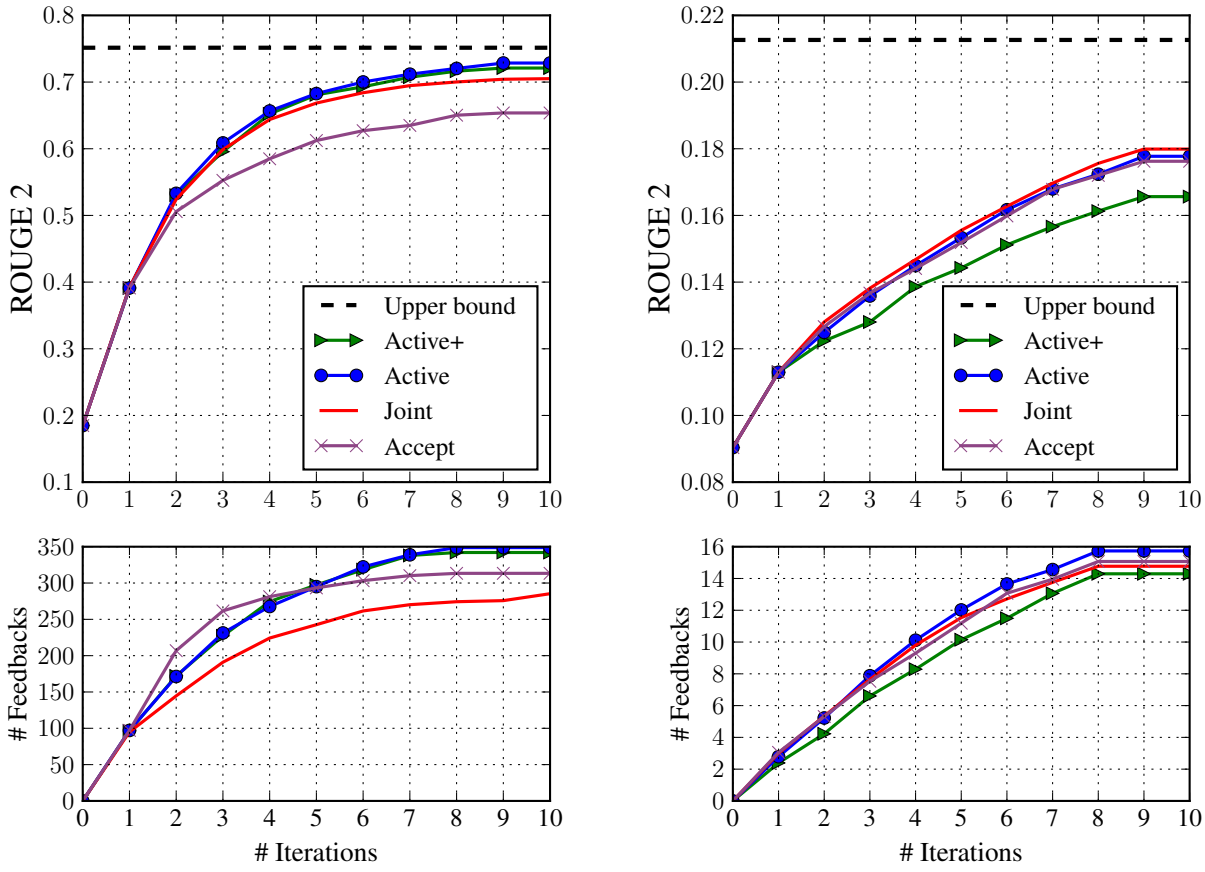


Figure 2: Analysis for the models over the DBS (left) and DUC'04 (right) datasets

on DBS. For DUC'04, the improvements are +.1 ROUGE-2 after ten iterations, which is relatively notable considering the lower upper bound of .21 ROUGE-2. This is primarily because DBS is a corpus of cohesive extracts, whereas DUC'04 consists of abstractive summaries. As a result, the oracles created using abstractive reference summaries have lower overlap of concepts as compared to that of the oracles created using extractive summaries.

For DBS, it becomes clear that the JOINT model converges faster with an optimum amount of feedback as compared to other models. ACCEPT takes relatively more feedbacks than JOINT, but performs low in terms of ROUGE scores. The best performing models are AL and AL+, which reach closest to the upper bound. This is clearly due to the exploratory nature of the models which use semantic representations of the concepts to predict uncertainty and importance of possible concepts for user feedback.

For DUC'04, the JOINT model reaches the closest to the upper bound, closely followed by AL. The JOINT model consistently stays above all other models and it gathers more important concepts due to optimizing feedbacks for concepts which lack feedback. Interestingly, AL+ performs rather worse in terms of both ROUGE scores and gathering important concepts. The primary reason for this is the fewer feedback collected from the simulation due to the abstractive property of reference summaries, which makes the AL+ model's prediction inconsistent.

5.4 Personalization

Figure 3 shows the performance of different models in comparison to two different oracles for the same document cluster. For DBS, the JOINT, AL, and AL+ models consistently converge to the upper bound in 4 iterations for different oracles, whereas ACCEPT takes longer for one oracle and does not reach the upper bound for the other.

For DUC'04, JOINT and AL show consistent performance across the oracles, whereas AL+ performs worse than the state-of-the-art system (iteration 0) for oracle created using abstractive summaries as shown in Figure 3 (right) for User:1.

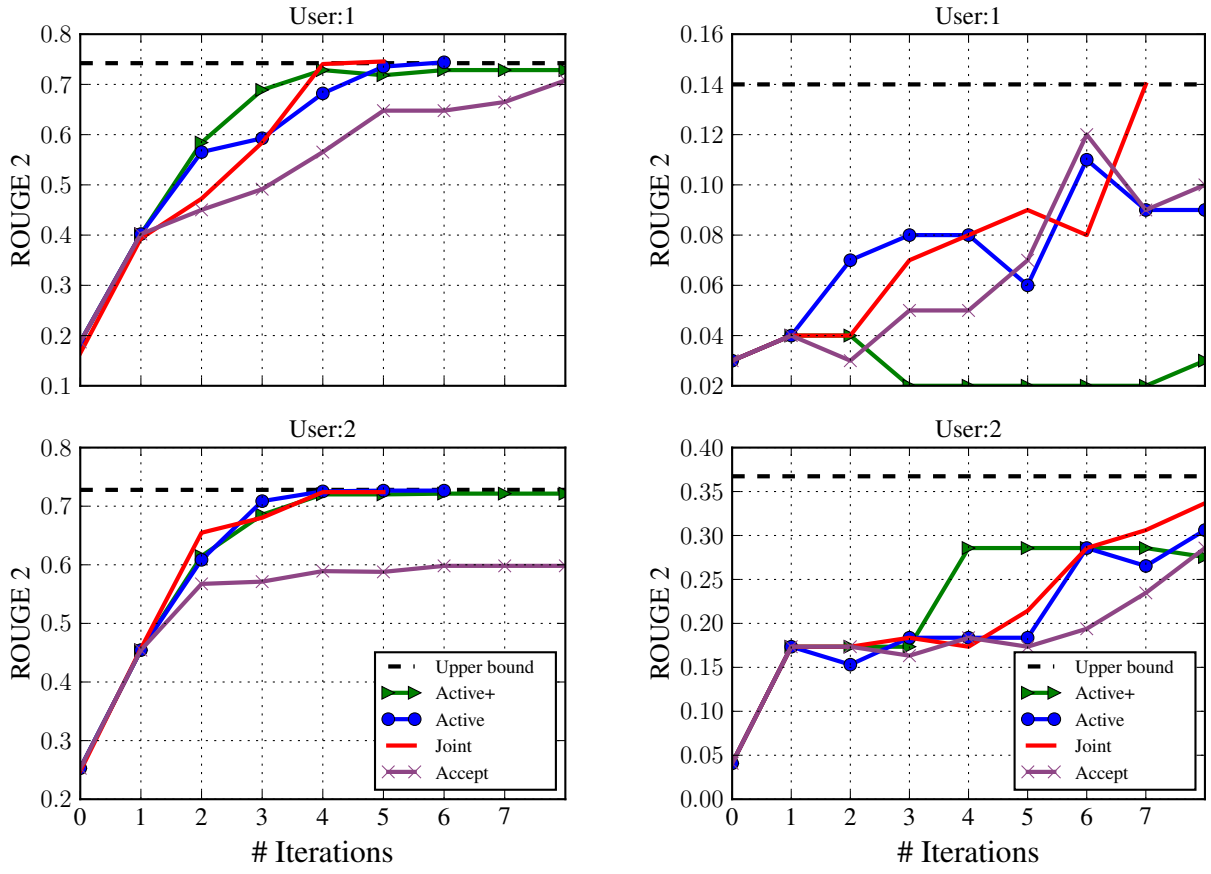


Figure 3: Analysis of models over cluster 7 from DBS (left) and cluster d30051t from DUC'04 (right) respectively for different oracles

However, for User:2, we observe a ROUGE-2 improvement of +.1 indicating that the predictions of the active learning system are better if there is more feedback. Nevertheless, we expect that in practical use, the human summarizers may give more feedback similar to DBS in comparison to DUC'04 simulation setting.

6 Conclusion and Future Work

We propose a novel ILP-based approach using interactive user feedback to create multi-document user-desired summaries. In this paper, we investigate pool-based active learning and joint optimization techniques to collect user feedback for identifying important concepts for a summary. Our models show that interactively collecting feedback consistently steers a general summary towards a user-desired personalized summary. We empirically checked the validity of our approach on standard datasets using simulated user feedback and observed that our framework shows promising results in terms of producing personalized multi-document summaries.

As future work, we plan to investigate more sophisticated sampling strategies based on active learning and concept graphs to incorporate lexical-semantic information for concept selection. We also plan to look into ways to propagate feedback to similar and related concepts with partial feedback, to reduce the total amount of feedback. This is a promising direction as we have shown that interactive methods help to create user-desired personalized summaries, and with minimum amount of feedbacks, it has propitious use in scenarios where user-adapted content is a requirement.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1. We also acknowledge the useful comments and suggestions of the anonymous reviewers.

References

Darina Benikova, Margot Mieskes, Christian M. Meyer, and Iryna Gurevych. 2016. Bridging the gap between extractive and abstractive summaries: Creation and evaluation of coherent extracts from heterogeneous sources. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*. Osaka, Japan, pages 1039–1050. http://aclweb.org/anthology/C16-1099.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL/HLT)*. Portland, OR, USA, pages 481–490. http://aclweb.org/anthology/P11-1049.

Shlomo Berkovsky, Timothy Baldwin, and Ingrid Zukerman. 2008. Aspect-based personalized text summarization. In *Adaptive Hypermedia and Adaptive Web-Based Systems. Proceedings of the 5th International Conference*, Springer, Berlin/Heidelberg, volume 5149 of *Lecture Notes in Computer Science*, pages 267–270. https://doi.org/10.1007/978-3-540-70987-9_31.

Florian Boudin, Hugo Mougard, and Benoit Favre. 2015. Concept-based summarization using integer linear programming: From concept pruning to multiple optimal solutions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMLNP)*. Lisbon, Portugal, pages 1914–1918. http://aclweb.org/anthology/D15-1220.

Ziqiang Cao, Chengyao Chen, Wenjie Li, Sujian Li, Furu Wei, and Ming Zhou. 2016. TGSum: Build Tweet Guided Multi-Document Summarization Dataset. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*. Phoenix, AZ, USA, pages 2906–2912. http://www.aaai.org/ocs/index.php/AAAI/AAAI16.

T. C. Craven. 2000. Abstracts produced using computer assistance. *Journal of the American Society for Information Science* 51(8):745–756. https://doi.org/10.1002/(SICI)1097-4571(2000)51:8<745::AID-ASI70>3.0.CO;2-Z.

Alberto Díaz and Pablo Gervás. 2007. User-model based personalized summarization. *Information Process Management* 43(6):1715–1734. https://doi.org/10.1016/j.ipm.2007.01.009.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *Journal of Artificial Intelligence Research* 22(1):457–479. https://www.jair.org/papers/paper1523.html.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*. Boulder, CO, USA, pages 10–18. http://aclweb.org/anthology/W09-1802.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. New Orleans, LA, USA, pages 19–25. https://doi.org/10.1145/383952.383955.

Jesús González-Rubio, Daniel Ortiz-Martínez, and Francisco Casacuberta. 2012. Active learning for interactive machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Avignon, France, pages 245–254. http://aclweb.org/anthology/E12-1025.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. Boulder, CO, USA, pages 362–370. http://aclweb.org/anthology/N09-1041.

Kai Hong, John M. Conroy, Benoît Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*. Reykjavik, Iceland, pages 1608–1616. http://www.lrec-conf.org/proceedings/lrec2014/summaries/1093.html.

Po Hu, Donghong Ji, Chong Teng, and Yujing Guo. 2012. Context-enhanced personalized social summarization. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*. Mumbai, India, pages 1223–1238. http://www.aclweb.org/anthology/C12-1075.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*. Sapporo, Japan, pages 423–430. https://doi.org/10.3115/1075096.1075150.

Rebecca Knowles and Philipp Koehn. 2016. Neural interactive translation prediction. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*.

Jan Kremer, Kim Steenstrup Pedersen, and Christian Igel. 2014. Active learning with support vector machines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4(4):313–326. https://doi.org/10.1002/widm.1132.

Chen Li, Xian Qian, and Yang Liu. 2013. Using supervised bigram-based ILP for extractive summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria, pages 1004–1013. http://aclweb.org/anthology/P13-1099.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Barcelona, Spain, pages 74–81. http://aclweb.org/anthology/W04-1013.

Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*. pages 63–70. https://doi.org/10.3115/1118108.1118117.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development* 2(2):159–165. https://doi.org/10.1147/rd.22.0159.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Advances in Information Retrieval. Proceedings of the 29th European Conference on IR Research (ECIR)*, Springer, Berlin/Heidelberg, volume 4425 of *Lecture Notes in Computer Science*, pages 557–564. https://doi.org/10.1007/978-3-540-71496-5_51.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Barcelona, Spain, pages 404–411. http://aclweb.org/anthology/W04-3252.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

Masumi Narita, Kazuya Kurokawa, and Takehito Utsuro. 2002. A Web-based English Abstract Writing Tool Using a Tagged E–J Parallel Corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*. Las Palmas, Spain. http://www.lrec-conf.org/proceedings/lrec2002/sumarios/137.htm.

Constantin Orăsan and Laura Hasler. 2006. Computer-aided Summarisation: What the User Really Wants. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*. Genoa, Italy, pages 1548–1551. http://www.lrec-conf.org/proceedings/lrec2006/summaries/52.html.

Constantin Orăsan, Ruslan Mitkov, and Laura Hasler. 2003. CAST: a computer-aided summarisation tool. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics (EACL)*. Budapest, Hungary, pages 135–138. http://aclweb.org/anthology/E03-1066.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*. Montréal, Canada, pages 1–9. http://aclweb.org/anthology/W12-2601.

Sun Park and Dong Un An. 2010. Automatic Query-based Personalized Summarization That Uses Pseudo Relevance Feedback with NMF. In *Proceedings of the 4th International Conference on Ubiquitous Information Management and Communication (ICUIMC)*. pages 61:1–61:7. https://doi.org/10.1145/2108616.2108690.

Helen Petrie and Nigel Bevan. 2009. The evaluation of accessibility, usability, and user experience. In Constantine Stephanidis, editor, *The Universal Access Handbook*, Boca Raton: CRC Press, Human Factors and Ergonomics, chapter 20, pages 1–16. https://doi.org/10.1201/9781420064995-c20.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances In Large Margin Classifiers*. MIT Press, pages 61–74.

Mickaël Poussevin, Vincent Guigue, and Patrick Gallinari. 2015. Extended recommendation framework: Generating the text of a user review as a personalized summary. In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015*. pages 34–41. http://ceur-ws.org/Vol-1448/paper7.pdf.

Nils Reimers, Judith Eckle-Kohler, Carsten Schnober, Jungi Kim, and Iryna Gurevych. 2014. GermEval-2014: Nested Named Entity Recognition with Neural Networks. In *Workshop Proceedings of the 12th Edition of the KONVENS Conference*. Hildesheim, Germany, pages 117–120.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*. Jeju Island, Korea, pages 233–243. http://aclweb.org/anthology/D12-1022.

Klaus Zechner. 2002. Automatic summarization of open-domain multiparty dialogues in diverse genres. *Journal of Computational Linguistics* 28(4):447–485. https://doi.org/10.1162/089120102762671945.

Haiqin Zhang, Zheng Chen Wei-ying Ma, and Qingsheng Cai. 2003. A study for documents summarization based on personal annotation. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop*. pages 41–48. https://doi.org/10.3115/1119467.1119473.

Flexible and Creative Chinese Poetry Generation Using Neural Memory

Jiyuan Zhang[1,2], **Yang Feng**[3,1,4], **Dong Wang**[1,*]
Yang Wang[1], **Andrew Abel**[6], **Shiyue Zhang**[1,5], **Andi Zhang**[1,5]
[1]Center for Speech and Language Technologies(CSLT), RIIT, Tsinghua University, China
[2]Shool of Software & Microelectronics, Peking University, China
[3]Key Laboratory of Intelligent Information Processing,Institute of Computing Technology,CAS
[4]Huilan Limited, Beijing, China
[5]Beijing University of Posts and Telecommunications, China
[6]Xi'an Jiaotong-Liverpool University, China
zhangjy_ml@pku.edu.cn, wangdong99@mails.tsinghua.edu.cn

Abstract

It has been shown that Chinese poems can be successfully generated by sequence-to-sequence neural models, particularly with the attention mechanism. A potential problem of this approach, however, is that neural models can only learn abstract rules, while poem generation is a highly creative process that involves not only rules but also innovations for which pure statistical models are not appropriate in principle. This work proposes a memory-augmented neural model for Chinese poem generation, where the neural model and the augmented memory work together to balance the requirements of linguistic accordance and aesthetic innovation, leading to innovative generations that are still rule-compliant. In addition, it is found that the memory mechanism provides interesting flexibility that can be used to generate poems with different styles.

1 Introduction

Classical Chinese poetry is a special cultural heritage with over 2,000 years of history and is still fascinating us today. Among the various genres, perhaps the most popular one is the quatrain, a special style with a strict structure (four lines with five or seven characters per line), a regulated rhythmical form (the last characters in the second and fourth lines must follow the same rhythm), and a required tonal pattern (tones of characters in some positions should satisfy a predefined regulation) (Wang, 2002). This genre flourished mostly in the Tang Dynasty, and so are often called 'Tang poems'. An example of a quatrain written by Wei Wang, a famous poet in the Tang Dynasty, is shown in Table 1.

Due to the stringent restrictions in both rhythm and tone, it is not trivial to create a fully rule-compliant quatrain. More importantly, besides such strict regulations, a good quatrain should also read fluently, hold a consistent theme, and express a unique affection. Therefore, poem generation is widely recognized as a very intelligent activity and can be performed only by knowledgeable people with a lot of training.

乐游原
Climbing the Paradise Mound
向晚意不适， (* Z Z P Z)
As I was not in a good mood this evening round,
驱车登古原。(P P P Z P)
I went by cart to climb the Ancient Paradise Mound.
夕阳无限好， (* P P Z Z)
It is now nearing dusk,
只是近黄昏。(* Z Z P P)
When the setting sun is infinitely fine, which is a must.

Table 1: An example of a 5-char quatrain. The tonal pattern is shown at the end of each line, where 'P' indicates a level tone, 'Z' indicates a downward tone, and '*' indicates the tone can be either. The translation is from (Tang, 2005).

In this paper we are interested in machine poetry generation. Several approaches have been studied by researchers. For example, rule-based methods (Zhou et al., 2010), statistical machine translation (SMT) models (Jiang and Zhou, 2008; He et al., 2012) and neural models (Zhang and Lapata, 2014; Wang et al., 2016a,c). Compared to previ-

[1]Corresponding author: Dong Wang; RM 1-303, FIT BLDG, Tsinghua University, Beijing (100084), P.R. China.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1364–1373
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1125

ous approaches (e.g., rule-based or SMT), the neural model approach tends to generate more fluent poems and some generations are so natural that even professional poets can not tell they are the work of machines (Wang et al., 2016a).

In spite of these promising results, neural models suffer from a particular problem in poem generation, a lack of innovation. Due to the statistical nature of neural models, they pay much more attention to high-frequency patterns, whereas they ignore low-frequency ones. In other words, the more regular and common the patterns, the better the neural model is good at learning them and tends to use them more frequently at run-time. This property certainly helps to generate fluent sentences, but it is not always useful: the major value of poetry is not fluency, but the aesthetic innovation that can stimulate some unique feelings. This is particularly true for Chinese quatrains that are highly compact and expressive: it is nearly impossible to find two similar works in the thousands of years of history in this genre, demonstrating the importance of uniqueness or innovation. Ironically, the most important thing, innovation, is largely treated as trivial, if not noise, by present neural models.

Actually this problem is shared by all generation models based on statistics (although it is more serious for neural models) and has aroused a long-standing criticism for machine poem generation: it can generate, and sometimes generate well, but the generation tends to be unsurprising and not particularly interesting. More seriously, this problem exists not only in poem generation, but also in all generation tasks that require innovation.

This paper tries to solve this extremely challenging problem. We argue that the essential problem is that statistical models are good at learning general rules (usage of regular words and their combinations) but are less capable of remembering special instances that are difficult to cover with general rules. In other words, there is only rule-based reasoning, no instance-based memory. We therefore present a memory-augmented neural model which involves a neural memory so that special instances can be saved and referred to at run-time. This is like a human poet who creates poems by not only referring to common rules and patterns, but also recalls poems that he has read before. It is hard to say whether this combination of rules and instances produces true innovation (which often requires real-life motivation rather than simple word reordering), but it indeed offers interesting flexibility to generate new outputs that look creative and are still rule-compliant. Moreover, this flexibility can be used in other ways, e.g., generating poems with different styles.

In this paper, we use the memory-augmented neural model to generate flexible and creative Chinese poems. We investigate three scenarios where adding a memory may contribute: the first scenario involves a well trained neural model where we aim to promote innovation by adding a memory, the second scenario involves an over-fitted neural model where we hope the memory can regularize the innovation, and in the third scenario, the memory is used to encourage generation of poems of different styles.

2 Related Work

A multitude of methods have been proposed for automatic poem generation. The first approach is based on rules and/or templates. For example, phrase search (Tosa et al., 2009; Wu et al., 2009), word association norm (Netzer et al., 2009), template search (Oliveira, 2012), genetic search (Zhou et al., 2010), text summarization (Yan et al., 2013). Another approach involves various SMT methods, e.g., (Jiang and Zhou, 2008; He et al., 2012). A disadvantage shared by the above methods is that they are based on the surface forms of words or characters, having no deep understanding of the meaning of a poem.

More recently, neural models have been the subject of much attention. A clear advantage of the neural-based methods is that they can 'discover' the meaning of words or characters, and can therefore more deeply understand the meaning of a poem. Here we only review studies on Chinese poetry generation that are mostly related to our research. The first study we have found in this direction is the work by Zhang and Lapata (2014), which proposed an RNN-based approach that produces each new line character-by-character using a recurrent neural network (RNN), with all the lines generated already (in the form of a vector) as a contextual input. This model can generate quatrains of reasonable quality. Wang et al. (2016b) proposed a much simpler neural model that treats a poem as an entire character sequence, and poem generation is conducted character-by-character. This approach can be

easily extended to various genres such as Song Iambics. To avoid theme drift caused by this long-sequence generation, Wang et al. (2016b) utilized the neural attention mechanism (Bahdanau et al., 2014) by which human intention is encoded by an RNN to guide the generation. The same model was used by Wang et al. (2016a) for Chinese quatrain generation. Yan (2016) proposed a hierarchical RNN model that conducts iterative generation. Recently, Wang et al. (2016c) proposed a similar sequence generation model, but with the difference that attention is placed not only on the human input, but also on all the characters that have been generated so far. They also proposed a topic planning scheme to encourage a smooth and consistent theme.

All the neural models mentioned above try to generate fluent and meaningful poems, but none of them consider innovation. The memory-augmented neural model proposed in this study intends to address this issue. Our system was built following the model structure and training strategy proposed by Wang et al. (2016a) due to its simplicity and demonstrated quality, but the memory mechanism is general and can be applied to any of the models presented above.

The idea of memory argumentation was inspired by the recent advance in neural Turing machine (Graves et al., 2014, 2016) and memory network (Weston et al., 2014). These new models equip neural networks with an external memory that can be accessed and manipulated via some trainable operations. In comparison, the memory in our work plays a simple role of knowledge storage, and the only operation is simple pre-defined READ. In this sense, our model can be regarded as a simplified neural Turing machine that omits training.

3 Memory-augmented neural model

In this section, we first present the idea of memory augmentation, and then describe the model structure and training method.

3.1 Memory augmentation

The idea of memory augmentation is illustrated in Fig. 1. It contains two components, the neural model component on the left, and the memory component on the right. In this work, the attention-based RNN generation model presented by (Wang et al., 2016a) is used as the neural model component, although any neural model is suitable. The memory component involves a set of 'direct' mappings from input to output, and therefore can be used to memorize some special cases of the generation that can not be represented by the neural model. For poem generation, the memory stores the information regarding which character should be generated in a particular context. The output from the two components are then integrated, leading to a consolidated output.

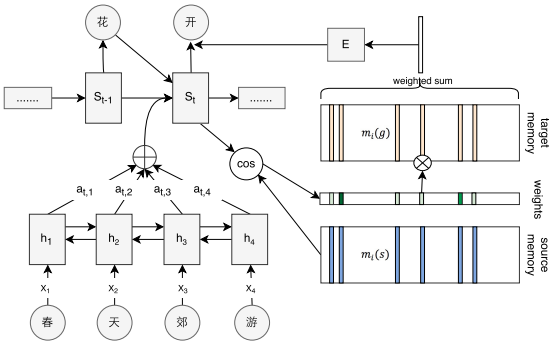


Figure 1: The memory-augmented neural model used for Chinese poetry generation.

There are several ways to understand the memory-augmented neural model. Firstly, it can be regarded as a way of combining reasoning (neural model) and knowledge (memory). Secondly, it can be regarded as a way of combining rule-based inference (neural model) and instance-based retrieval (memory). Thirdly, it can be regarded as a way of combining predictions from complementary systems, where the neural model is continuous and parameter-shared, while the memory is discrete and contains no parameter sharing. Finally, the memory can be regarded as an effective regularization that constrains and modifies the behavior of the neural model, resulting in generations with desired properties. Note that this memory-augmented neural model is inspired by and related to the memory network proposed by Weston et al.(2014) and Graves et al.(2016), but we more focus on an accompanying memory that plays the role of assistance and regularization.

3.2 Model structure

Using the Chinese poetry generation model shown in Fig. 1 as an example, this section discusses the creation of a memory-augmented neural model. Firstly, the neural model part is an attention-based sequence-to-sequence model (Bahdanau et al., 2014). The encoder is a bi-

directional RNN (with GRU units) that converts the input topic words, denoted by the embeddings of the compositional characters $(x_1, x_2, ..., x_N)$, into a sequence of hidden states $(h_1, h_2, ..., h_N)$. The decoder then generates the whole quatrain character-by-character, denoted by the corresponding embeddings $(y_1, y_2, ...)$. At each step t, the prediction for the state s_t is based on the last generation y_{t-1}, the previous status s_{t-1} of the decoder, as well as all the hidden states $(h_1, h_2, ...)$ of the encoder. Each hidden state h_i contributes to the generation according to a relevance factor α_t that measures the similarity between s_{t-1} and h_i. This is written as:

$$s_t = f_d(y_{t-1}, s_{t-1}, \sum_{i=1}^{N} \alpha_{t,i} h_i)$$

where $\alpha_{t,i}$ represents the contribution of h_i to the present generation, and can be implemented as any function. The output of the model is a posterior probability over the whole set of characters, written by

$$z_t = \sigma(s_t W)$$

where W is the projection parameter.

The memory consists of a set of elements $\{m_i\}_{i=1}^{K}$, where K is the size of the memory. Each element m_i involves two parts, the source part $m_i(s)$, that encodes the context, i.e. when this element should be selected, and the target part $m_i(g)$, that encodes what should be output if this element is selected. In our study, the neural model is firstly trained, and then the memory is created by running f_d (the decoder of the neural model). Specifically, for the k-th poem selected to be in the memory, the character sequence is input to the decoder one by one, with the contribution from the encoder set to zero. Denoting the starting position of this poem in the memory is p_k, the status of the decoder at the j-th step is used as the source part of the $(p_k + j)$-th element of the memory, and the embedding of the corresponding character, x_j, is set to be the target part. this is formally written as:

$$m_i(s) = f_d(x_{j-1}, s_{j-1}, 0) \quad (1)$$

and

$$m_i(g) = x_j$$

where

$$i = p_k + j.$$

At run-time, the memory elements are selected according to their fit to the present decoder status s_t, and then the outputs of the selected elements are averaged as the output of the memory component. We choose cosine distance to measure the fitting degree, and have[1]:

$$v_t = \sum_{i=1}^{K} cos(s_t, m_i(s)) m_i(g). \quad (2)$$

The output of the neural model and the memory can be combined in various ways. Here, a simple linear combination before the softmax is used, i.e.,

$$z_t = \sigma(s_t W + \beta v_t E) \quad (3)$$

where β is a pre-defined weighting factor, and E contains word embeddings of all the characters. Although it is possible to train β from the data, we found that the learned β is not better than the manually-selected one. This is probably because β is a factor to trade-off the contribution from the model and the memory, and how to make the trade-off should be a 'prior knowledge' rather than a tunable parameter. In fact, if it is trained, than it will be immediately adapted to match the training data, which will nullify our effort to encourage innovative generation.

3.3 Model Training

In our implementation, only the neural model component is required to be trained. The training algorithm follows the scheme defined in (Wang et al., 2016a), where the cross entropy between the distributions over Chinese characters given by the decoder and the ground truth is used as the objective function. The optimization uses the SGD algorithm together with AdaDelta to adjust the learning rate (Zeiler, 2012).

4 Memory augmentation for Chinese poetry generation

This section describes how the memory mechanism can be used to trade-off between the requirements for rule-compliant generation and aesthetic innovation, and how it can also be used to do more interesting things, for example style transfer.

4.1 Memory for innovative generation

In this section, we describe how the memory mechanism promotes innovation. Monitoring the

[1] In fact, we run a parallel decoder to provide s_t in Eq.(2). This decoder does not accept input from the encoder and so is consistent with the memory construction process as Eq.(1).

training process for the attention-based model, we found that the cost on the training set will keep decreasing until approaching zero, but on the validation set, the degradation stops after only one iteration. This can be explained by the fact that Chinese quatrains are highly unique, so the common patterns can be fully learned in one iteration, resulting in overfitting with additional iterations. Due to the overfitting, we observe that with the one-iteration model, reasonable poems can be generated, and with the over-fitted model, the generated poems are meaningless, in that they do not resemble feasible character sequences.

The energy model perspective helps to explain this difference. For the one-iteration model, the energy surface is smooth and the energy of the training data is not very low, as illustrated in plot (a) in Fig. 2, where the x-axis represents the input and y-axis represents the output, and the z-axis represents the energy. With this model, inputs with small variance will be attracted to the same low-energy area, leading to similar generations. These generations are trivial, but at least reasonable. If the model is overfitted, however, the energy at the locations of the training data becomes much lower than their surrounding areas, leading to a bumpy energy surface as shown in plot (b) in Fig. 2. With this model, inputs with a small variation may be attracted to very different low-energy areas, leading to significantly different generations. Since many of the low-energy areas are nothing to do with good generations but are simply caused by the complex energy function, the generations can be highly surprising for human readers, and the quality is not guaranteed. In some sense, these generations can be regarded as 'innovative' , but based on observations made in our experiments, most of them are meaningless.

The augmented memory introduces a new energy function, which is combined with the energy function of the neural model to change the energy surface of the generation system. This can be seen in Eq. (3), where $s_t W$ and $\beta v_t E$ can be regarded as the energy function of the neural model component and the memory component, respectively, and the energy function of the memory-augmented system is the sum of the energy functions of these two components. For this reason, the effect of the memory mechanism can be regarded as a regularization of the neural model that will adjust its generation behavior.

This regularization effect is illustrated in Fig. 2, where the energy function of the memory shown in plot (c) is added to the energy function of the one-iteration model and the overfitted model, as shown in plot (e) and plot (f) respectively. It can be seen that with the memory involved, the energy surface becomes more bumpy with the one-iteration model, and more smooth with the overfitted model. In the former case, the effect of the memory is to encourage innovation, while still focusing on rule-compliance, and in the latter case, the effect is to encourage rule compliance, while keeping the capability for innovation.

It is important to notice that the energy function of the memory component is a linear combination of the energy functions of the compositional elements (see Eq.(2)), each of which is convex and is minimized at the location represented by the element. This means that the energy surface of the memory is rather 'healthy', in the sense that low-energy locations mostly correspond to good generations. For this reason, the regularization provided by the memory is safe and helpful.

4.2 Memory for style transfer

The effect of the memory is easy to control. For example, the complexity of the behavior can be controlled by the memory size, the featured bias can be controlled by memory selection, and the strength of the impact can be controlled by the weighting parameter β. This means that the memory mechanism is very flexible and can be used to produce poems with desired properties.

In this work, we use these capabilities to generate poems with different styles. This has been illustrated in Fig. 2, where the energy function of the style memory shown in plot (d) is biased towards a particular style, and once it is added to energy function of the one-iteration model, the resulting energy function shown in plot (g) obtains lower values at locations corresponding to the locations of the memory, which encourages generation of poems with similar styles as those poems in the memory.

5 Experiments

This section describes the experiments and results carried out in this paper. Here, The baseline system was a reproduction of the Attention-based system presented in (Wang et al., 2016a). the model in This system has been shown to be rather flexi-

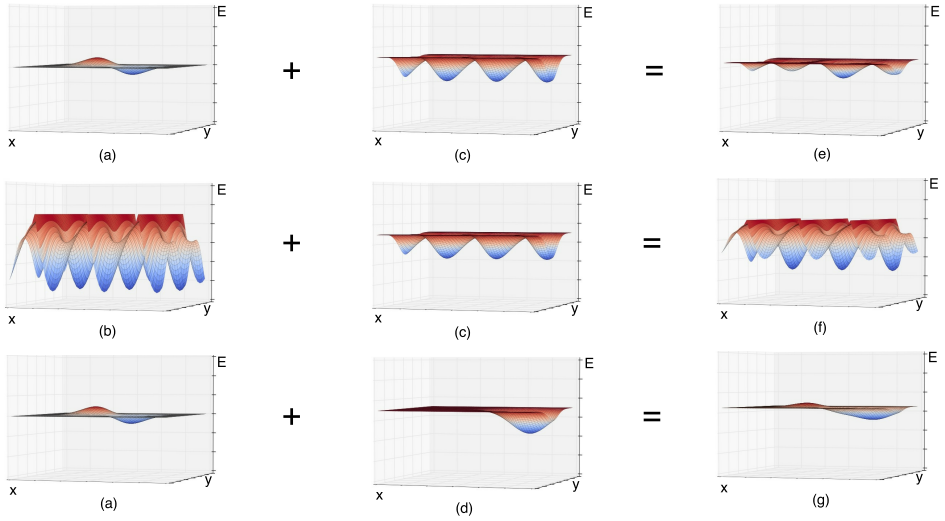


Figure 2: The energy surface for (a) one-iteration model (b) overfitted model (c) memory (d) style memory (e) one-iteration model augmented with memory (f) overfitted model augmented with memory (g) one-iteration model augmented with style memory.

ble and powerful: it can generate different genres of Chinese poems, and when generating quatrains it has been shown to be able to fool human experts in many cases (Wang et al., 2016a) and the authors had did a thorough comparison with competitive methods mentioned in the related work of this paper. We obtained the database and the source code (in theano), and reproduced their system using Tensorflow from Google[2]. We didn't make comparisons with some previous methods such as NNLM, SMT, RNNPG as they had been fully compared in (Wang et al., 2016a) and all of them were much worse than the attention-based system. Another reason was that the experts were not happy to evaluate poems with clearly bad quality. We also reproduced the model in (Wang et al., 2016c) with the help of the first author. However, since their implementation did not involve any restrictions on rhythm and tone, the experts were reluctant to recognize them as good poems. With a larger dataset (e.g., 1 Million poems), it is assumed that the rhythm and tone can be learned and their system would be good in both fluency and rule compliance. It should be also emphasized that the memory approach proposed in this paper is a general technique and is complementary to other efforts such as the planning approach (Wang et al., 2016c) and the recursive approach (Yan, 2016).

Based on the baseline system, we built the memory-augmented model, and conducted two experiments to demonstrate its power. The first is an innovation experiment which employs memory to promote or regularize the generation of innovative poems, and the second is a style-transfer experiment which employs memory to generate flexible poems in different styles.

We invited 34 experts to participate in the experiments, and all of them have rich experience not only evaluating poems, but also in writing them. Most of the experts are from prestigious institutes, including Peking university and the Chinese Academy of Social Science (CASS). Following the suggestions of the experts, we use five metrics to evaluate the generation, as listed below:

- Compliance: if regulations on tones and rhymes are satisfied;
- Fluency: if the sentences read fluently and convey reasonable meaning;
- Theme consistency: if the entire poem adheres to a single theme;
- Aesthetic innovation: if the quatrain stimulates any aesthetic feeling with elaborate innovation;
- Scenario consistency: if the scenario remains consistent.

5.1 Datasets

The baseline system was built with two customized datasets. The first dataset is a Chinese po-

[2]https://www.tensorflow.org/

em corpus (CPC), which we used in this work to train the embeddings of Chinese characters. Our CPC dataset contains 284,899 traditional Chinese poems in various genres, including Tang quatrains, Song Iambics, Yuan Songs, and Ming and Qing poems. This large quantity of data ensures reliable learning for the semantic content of most Chinese characters.

Our second dataset is a Chinese quatrain corpus (CQC) that we have collected from the internet, which consists of $13,299$ 5-char quatrains and $65,560$ 7-char quatrains. This corpus was used to train the attention-based RNN baseline. We filtered out the poems whose characters are all low-frequency (less than 100 counts in the database). After the filtering, the remaining corpus contains 9,195 5-char quatrains and 49,162 7-char quatrains. We used 9,000 5-char and 49,000 7-char quatrains to train the attention model, and the rest for validation.

Another two datasets were created for use in the memory-augmented system. Our first dataset, MEM-I, contains 500 quatrains randomly selected from our CQC corpus. This dataset was used to produce the memory in the innovation experiment; the second dataset, MEM-S, contains 300 quatrains with clear styles, including 100 pastoral, 100 battlefield and 100 romantic quatrains. It was used to generate memory with different styles in the style-transfer experiment. All the datasets will be released online[3].

5.2 Evaluation Process

We invited 34 experts to evaluate the quality of the poem generation. In the innovation experiment, the evaluation consisted of a comparison between different systems and configurations in terms of the five metrics. The innovation questions presented the expert with two poems, and asked them to judge which of the poems was better in terms of the five metrics; in the style-transfer experiment, the evaluation was performed by identifying the style of a generated poem. The evaluation was conducted online, with each questionnaire containing 11 questions focusing on innovation and 4 questions concerned with style-transfer. Each of the style-transfer questions presented the expert with a single poem and asked them to score it between 1 to 5, with a larger score being better, in terms of compliance, aesthetic innovation, scenario consistency, and fluency. They were also asked to specify the style of the poem.

Using the poems generated by our systems, we generated many different questions of both types, and then created a number of online questionnaires that randomly selected from these questions. This meant that as discussed above, each questionnaire had 11 randomly selected innovation questions, and 4 randomly selected style transfer questions. Each question was only used once, meaning that it was not duplicated on multiple questionnaires, and so each questionnaire was different.

Experts could choose to answer multiple questionnaires if they wished, as each one was different. From the 34 experts, we collected 69 completed questionnaires, which equals to 759 innovation questions and 276 style-transfer questions.

5.3 Innovation experiment

This experiment focuses on the contribution of memory for innovative poem generation. We experimented with two configurations: one is with a one-iteration model (C_1) and the other is with an overfitted model (C_∞). The memory was generated from the 500 quatrains in MEM-I, and the weighting factor was defined empirically as 16 for C_1 and 49 for C_∞.

The topics of the generation were 160 keywords randomly selected from Shixuhanyinge (Liu, 1735). Given a pair of poems generated by two different configurations using the same topic, the experts were asked to choose which one they preferred. The evaluation is therefore pair-wised, and each pair of configurations contains at least 180 evaluations. The results are shown in Table 2, where the preference ratio for each pair of configurations was tested in terms of the 5 metrics.

From the first row of Table 2, we observe that the experts have a clear preference for the poems generated by the C_1 model, the one that can produce fluent yet uninteresting poems. In particular, the 'aesthetic innovation' score for C_∞ is not better than C_1, which was different from what we expected. Informal offline discussions with the poetry experts found that the experts identified some innovative expression in the C_∞ condition, but most of the them was regarded as being nonsense in the opinion of many of the experts. In comparison to sparking innovation, fluency and being meaningful is more important not only for non-expert readers, but also for professional poet-

[3] http://vivi.cslt.org

	Preference Ratio				
	Compliance	Fluency	Theme Consistency	Aesthetic Innovation	Scenario Consistency
C_1 vs C_∞	0.59:0.41	0.68:0.32	0.70:0.30	0.68:0.32	0.69:0.31
C_1 vs C_1+Mem	0.41:0.59	0.36:0.64	0.37:0.63	0.33:0.67	0.43:0.57
C_∞ vs C_∞+Mem	0.40:0.60	0.26:0.74	0.32:0.68	0.30:0.70	0.36:0.64
C_1 vs C_∞+Mem	0.43:0.57	0.58:0.42	0.59:0.41	0.50:0.50	0.59:0.41

Table 2: Preference ratios for systems with or without overfitting and with or without memory augmentation.

s. In other words, only meaningful innovation is regarded as innovation, and irrational innovation is simply treated as junk.

From the second and third rows of Table 2, it can be seen that involving memory significantly improves both C_1 and C_∞, particularly for C_∞. For C_1, the most substantial improvement is observed in terms of 'Aesthetic innovation', which is consistent with our argument that memory can help encourage innovation for this model. For C_∞, 'Fluency' seems to be the most improved metric. This is also consistent with our argument that involving memory constrains over-innovation for over-fitted models.

The last row of Table 2 is an extra experiment that investigates if C_∞ is regularized well enough after introducing the memory. It seems that with the regularization, the overfitting problem is largely solved, and the generation is nearly as fluent and consistent as the C_1 condition. Interestingly, the score for aesthetic innovation is also significantly improved. Since the regularization is not supposed to boost innovation, this seems confusing at first glance (in comparison to the result on the same metric in the first row), but this is probably because the increased fluency and consistency makes the innovation more appreciated, therefore doubly confirming our argument that true innovation should be reasonable and meaningful.

5.4 Style-transfer experiment

In the second experiment, the memory mechanism is used to generate poems in different styles. We chose three styles: pastoral, battlefield, and romantic. A style-specific memory, which we call style memory, was constructed for each style by the corresponding quatrains in the MEM-S dataset. The system with one-iteration model C_1 was used as the baseline. Two sets of topics were used in the experiment, one is general and the other is style-biased. The experiments then investigate if the memory mechanism can produce a clear style if the topic is general, and can transfer to a different style if the topic is style-biased already. The experts were asked to specify the style from four options including the three defined above and a 'unclear style' option. In addition, the experts were asked to score the poems in terms of compliance, fluency, aesthetic innovation, and scenario consistency, which we can use to check if the style transfer impacts the quality of the poem generation. Note that we did not ask for the theme consistency to be scored in this experiment because the topic words were not presented to the experts, in order to prevent the topic affecting their judgment regarding the style. The score ranges from 1 to 5, with a larger score being better.

Table 3 presents the results with the general topics. The numbers show the probabilities that the poems generated by a particular system were labeled as having various styles. Since the topics are unbiased in types, the generation of the baseline system is assumed to be with unclear styles. For other systems, the style of the generation is assumed to be the same as the style of their memories. The results in Table 3 clearly demonstrates these assumptions. The tendency that romantic poems are recognized as pastoral poems is a little surprising. Further analysis shows that experts tend to recognize romantic poems as pastoral poems only if there are any related symbols such as trees, mountain, river. These words are very general in Chinese quatrains. The indicator words of romantic poems such as skirt, rouge, and singing are not as popular and their indication power is not as strong, leading to less labeling of romantic poems, as shown in the results.

	Probability			
Model	Pastoral	Battlefield	Romantic	Unclear
C_1 (Baseline)	0.09	0.04	0.18	0.69
C_1 + Pastoral Mem	0.94	0.00	0.06	0.00
C_1 + Battlefield Mem	0.05	0.93	0.00	0.02
C_1 + Romantic Mem	0.17	0.00	0.61	0.22

Table 3: Probability that poems generated by each configuration with general topics are labeled as various styles.

We also tested transferring from one style to

another. This was achieved by generating poems with some style-biased topics, and then using a style memory to force the generation to change the style. Our experiments show that in 73% cases the style can be successfully transferred.

Finally, the scores of the poems generated with and without the style memories are shown in Table 4, where the poems generated with both general and style-biased topics are accounted for. It can be seen that overall, the style transfer may degrade fluency a little. This is understandable, as enforcing a particular style has to break the optimal generation with the baseline, which is assumed to be good at generating fluent poems. Nevertheless the sacrifice is not significant.

Method	Compliance	Fluency	Aesthetic Innovation	Scenario Consistence
C_1 (baseline)	4.10	3.01	2.53	2.94
C_1 + Pastoral Mem	4.07	3.00	3.07	3.17
C_1 + Battlefield Mem	3.82	2.63	2.60	2.95
C_1 + Romantic Mem	4.00	2.78	2.59	3.00
C_1 + All Mem	3.95	2.80	2.74	3.05

Table 4: Averaged scores for systems with or without style memory.

5.5 Examples

Table 5 to Table 7 shows example poems generated by the system C_1, C_1+Mem and C_1+Style Mem where the style in this case is set to be romantic. The three poems were generated with the same, very general, topic ('自(oneself)'). More examples are given in the supporting material.

自从此意无心物，
Nothing in my heart,
一日东风不可怜。
Spring wind is not a pity.
莫道人间何所在，
Don't ask where it is,
我今已有亦相传。
I've noticed that and tell others.

Table 5: Example poems generated by the C_1 system.

6 Conclusions

In this paper, we proposed a memory mechanism to support innovative Chinese poem generation by neural models augmented with a memory. Experimental results demonstrated that memory can boost innovation from two opposite directions: either by encouraging creative generation for regularly-trained models, or by encouraging rule-compliance for overfitted models. Both strategies work well, although the former generated poetry that was preferred by experts in our experiments. Furthermore, we found that the memory can be used to modify the style of the generated poems in a flexible way. The experts we collaborated with feel that the present generation is comparable to today's experienced amateur poets. Future work involves investigating a better memory selection scheme. Other regularization methods (e.g., norm or drop out) are also interesting and may alleviate the over-fitting problem.

一山自有无人语，
Nobody speaking in the mountain,
不是青云入水边。
Also no green cloud stepping into the river.
莫把春风吹落叶，
Spring wind does not stir leaves,
花开绿树满江船。
But flowers blooming in trees and flying to boats.

Table 6: Example poems generated by the C_1+Mem system.

花香粉脸胭脂染，
Beautiful face addressed by rouge,
帘影鸳鸯绿嫩妆。
Mandarin duck outside the curtain.
翠袖红蕖春色冷，
Green sleeves and red flowers in cold spring,
柳梢褪叶暗烟芳。
Willow leaves gone in fragrant mist.

Table 7: Example poems generated by the C_1+Style Mem system where the style is romantic.

Acknowledgments

This paper was supported by the National Natural Science Foundation of China (NSFC) under the project NO.61371136, NO.61633013, NO.61472428.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 .

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. arXiv preprint arXiv:1410.5401 .

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. Nature 538(7626):471–476.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating Chinese classical poems with statistical machine translation models. In Twenty-Sixth AAAI Conference on Artificial Intelligence.

Long Jiang and Ming Zhou. 2008. Generating Chinese couplets using a statistical mt approach. In Proceedings of the 22nd International Conference on Computational Linguistics. Association for Computational Linguistics, volume 1, pages 377–384.

Wenwei Liu. 1735. ShiXueHanYing.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku: Generating haiku with word associations norms. In Proceedings of the Workshop on Computational Approaches to Linguistic Creativity. Association for Computational Linguistics, pages 32–39.

H Oliveira. 2012. Poetryme: a versatile platform for poetry generation. In Proceedings of the ECAI 2012 Workshop on Computational Creativity, Concept Invention, and General Intelligence.

Yihe Tang. 2005. English Translation for Tang Poems (Ying Yi Tang Shi San Bai Shou). Tianjin People Publisher.

Naoko Tosa, Hideto Obara, and Michihiko Minoh. 2009. Hitch haiku: An interactive supporting system for composing haiku poem. Entertainment Computing-ICEC 2008 pages 209–216. Springer.

Li Wang. 2002. A Summary of Rhyming Constraints of Chinese Poems (Shi Ci Ge Lv Gai Yao), volume 1. Beijin Press.

Qixin Wang, Tianyi Luo, and Dong Wang. 2016a. Can machine generate traditional Chinese poetry? a feigenbaum test. In BICS 2016.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016b. Chinese song iambics generation with neural attention-based model. In IJCAI 16.

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016c. Chinese poetry generation with planning based neural network. In COLING 2016.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. arXiv preprint arXiv:1410.3916 .

Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. 2009. New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. Entertainment Computing-ICEC 2009 pages 191–196. Springer.

Rui Yan. 2016. i, Poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In IJCAI2016.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 2013. i, Poet: automatic Chinese poetry composition through a generative summarization framework under constrained optimization. In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. AAAI Press, pages 2197–2203.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 .

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pages 670–680.

Cheng-Le Zhou, Wei You, and Xiaojun Ding. 2010. Genetic algorithm and its implementation of automatic generation of Chinese Songci. Journal of Software 21(3):427–437.

Learning to Generate Market Comments from Stock Prices

Soichiro Murakami[†,*] **Akihiko Watanabe** [†,*] **Akira Miyazawa** [‡,¶,*] **Keiichi Goshima** [†,*]
Toshihiko Yanase [§] **Hiroya Takamura** [†,*] **Yusuke Miyao** [‡,¶,*]
[†] Tokyo Institute of Technology [‡] The Graduate University for Advanced Studies
[§] Hitachi, Ltd. [¶] National Institute of Informatics
[*] National Institute of Advanced Industrial Science and Technology
{murakami,watanabe}@lr.pi.titech.ac.jp,
goshima.k.aa@trn.dis.titech.ac.jp, takamura@pi.titech.ac.jp
toshihiko.yanase.gm@hitachi.com, {miyazawa-a,yusuke}@nii.ac.jp

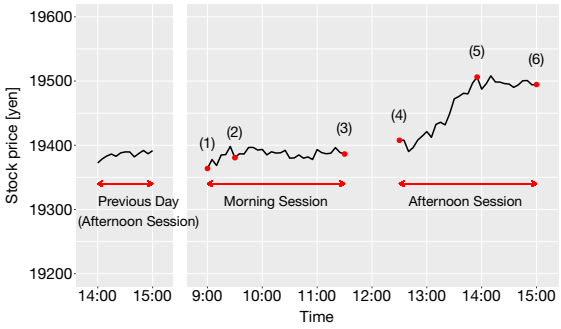
Abstract

This paper presents a novel encoder-decoder model for automatically generating market comments from stock prices. The model first encodes both short- and long-term series of stock prices so that it can mention short- and long-term changes in stock prices. In the decoding phase, our model can also generate a numerical value by selecting an appropriate arithmetic operation such as subtraction or rounding, and applying it to the input stock prices. Empirical experiments show that our best model generates market comments at the fluency and the informativeness approaching human-generated reference texts.

1 Introduction

Various industries such as finance, pharmaceuticals, and telecommunications have been increasingly providing opportunities to treat various types of large-scale numerical time-series data. Such data are hard for non-specialists to interpret in detail and time-consuming even for specialists to construe. As a result, there has been a growing interest in automatically generating concise descriptions of such data, i.e., data summarization. This interest in data summarization is encouraged by the recent development of neural network-based text generation methods. Given an appropriate architecture, a neural network can generate a sentence that is mostly grammatical and semantically reasonable.

In this study, we focus on the task of generating market comments from a time-series of stock prices. We adopt an encoder-decoder model (Sutskever et al., 2014) and exploit its capability to learn to capture the behavior of the input and generate a description of it. Although encoder-decoder models can learn to do this, they need to be provided with an appropriate network-architecture and necessary information. We use Figure 1 to illustrate the characteristic problems of comment generation for time-series of stock prices. The figure shows the Nikkei Stock Average (*Nikkei 225*, or simply *Nikkei*), which is a stock market index calculated from 225 selected issues, on some consecutive trading days accompanied by the market comments made at some specific time points in the span. The first problem is that market comments do not merely describe the increase and decrease of the price. They also often describe how the price changes compared with the previous period, such as "*continues to fall*" in (3) of Figure 1, "*turns to rise*" in (2), and "*rebound*" in (6). Market comments sometimes describe the change in price compared with the prices in the previous week. The second problem is that market comments also



	Time	Comment
(1)	09:00	Nikkei opens with a continual fall.
(2)	09:29	Nikkei turns to rise.
(3)	11:30	Nikkei continues to fall. The closing price of the morning session decreases by 5 yen to 19,386 yen.
(4)	12:30	Nikkei rises at the beginning of the afternoon session.
(5)	13:54	Nikkei gains more than 100 yen.
(6)	15:00	Nikkei rebounds and closes up 102 yen to 19,494 yen.

Figure 1: Nikkei 225 and market comments.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1374–1384
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1126

contain expressions that depend on their delivery time: e.g., "*opens with*" in (1), "*closing price of the morning session*" in (3), and "*beginning of the afternoon session*" in (4). The third problem is that market comments typically contain numerical values, which often cannot be copied from the input prices. Such numerical values probably cannot be generated as other words are generated by the standard decoder. This difficulty can be easily understood as analogous with the difficulty of generating named entities by encoder-decoder models. To derive such values, the model needs arithmetic operations such as subtraction as in examples (3) and (6) mentioning the difference in price and rounding as in example (5).

To address these problems, we present a novel encoder-decoder model to automatically generate market comments from stock prices. To address the first problem of capturing various types of change in different time scales, the model first encodes data consisting of both short- and long-term time-series, where a multi-layer perceptron, a recurrent neural network, or a convolutional network is adopted as a basic encoder. In the decoding phase, we feed our model with the delivery time of the market comment to generate the expressions depending on time of day to address the second problem. To address the third problem regarding with numerical values mentioned in the generated text, we allow our model to choose an arithmetic operation such as subtraction or rounding instead of generating a word.

The proposed methods are evaluated on the task of generating Japanese market comments on the Nikkei Stock Average. Automatic evaluation with BLEU score (Papineni et al., 2002) and F-score of time-dependent expressions reveals that our model outperforms a baseline encoder-decoder model significantly. Furthermore, human assessment and error analysis prove that our best model generates characteristic expressions discussed above almost perfectly, approaching the fluency and the informativeness of human-generated market comments.

2 Related Work

The task of generating descriptions from time-series or structured data has been tackled in various domains such as weather forecasts (Belz, 2007; Angeli et al., 2010), healthcare (Portet et al., 2009; Banaee et al., 2013b), and sports (Liang et al., 2009). Traditionally, many studies used hand-crafted rules (Goldberg et al., 1994; Dale et al., 2003; Reiter et al., 2005). On the other hand, interest has recently been growing in automatically learning a correspondence relationship from data to text and generating a description of this relationship since large-scale data in diversified formats have become easy to acquire. In fact, a data-driven approach has been extensively studied nowadays for various tasks such as image caption generation (Vinyals et al., 2015) and weather forecast generation (Mei et al., 2016b).

The task, called data-to-text or concept-to-text, is generally divided into two subtasks: content selection and surface realization. Whereas previous studies tackled the subtasks separately (Barzilay and Lapata, 2005; Wong and Mooney, 2007; Lu et al., 2009), recent work has focused on solving them jointly using a single framework (Chen and Mooney, 2008; Kim and Mooney, 2010; Angeli et al., 2010; Konstas and Lapata, 2012, 2013).

More recently, there has been some work on an encoder-decoder model (Sutskever et al., 2014) for generating a description from time-series or structured data to solve the subtasks jointly in a single framework, and this model has been proven to be useful (Mei et al., 2016b; Lebret et al., 2016). However, the task of generating a description from numerical time-series data presents difficulties such as the second and third problems mentioned in Section 1. For the second problem, the model needs to be fed with information on delivery time. Also, the model needs arithmetic operations such as subtraction for the third problem because even if we simply apply a copy mechanism (Gu et al., 2016; Gulcehre et al., 2016) to the model, it cannot derive a calculated value such as (3), (5), or (6) in Figure 1 from input. Thus, in this work, we tackle these problems and develop a model on the basis of the encoder-decoder model that can mention a specific numerical value by referring to the input data or producing a processed value with mathematical calculation and mention time-dependent expressions by incorporating the information on delivery time into its decoder.

There has also been some work on generating market comments. Kukich (1983) developed a system consisting of rule-based components for generating stock reports from a database of daily stock quotes. Although she used several components individually and had to define a number of rules for the generation, our encoder-decoder model can

perform it with fewer and simpler rules for the calculation. Aoki and Kobayashi (2016) developed a method on the basis of a weighted bi-gram language model for automatically describing trends of time-series data such as the Nikkei Stock Average. However, they did not attempt to refer to specific numerical values such as closing prices and amounts of rises in price although such descriptions are often used in market comments as shown in Figure 1 (3), (5), and (6). In contrast, we present a novel approach to generate natural language descriptions of time-series data that can not only able to describe trends of the data but also mention specific numerical values by referring to the time-series data.

3 Generating Market Comments

To generate market comments on stock prices, we introduce an encoder-decoder model. Encoder-decoder models have been widely used and proven useful in various tasks of natural language generation such as machine translation (Cho et al., 2014) and text summarization (Rush et al., 2015). Our task is similar to these tasks in that the system takes sequential data and generates text. Therefore, it is natural to use an encoder-decoder model in modeling stock prices.

Figure 2 illustrates our model. In describing time-series data, the model is expected to capture various types of change and important values in the given sequence, such as absolute or relative changes and maximum or minimum value, in different time-scales. Moreover, it is necessary to generate time-dependent comments and numerical values that require arithmetic operations for derivation, such as "The closing price of *the morning session* decreases by *5 yen*...". To achieve these, we present three strategies that alter the standard encoder-decoder model.

First (Section 3.1), we use several encoding methods for time-series data, as in (1) of Figure 2, to capture the changes and important values. Second (Section 3.2), we incorporate delivery-time information into the decoder, as in (2) of Figure 2, to generate time-dependent comments. For the decoder, we use a recurrent neural network language model (RNNLM) (Mikolov et al., 2010), which is widely used in language generation tasks. Finally (Section 3.3), we extend the decoder to estimate arithmetic operations, as in (3) of Figure 2, to generate numerical values in market comments.

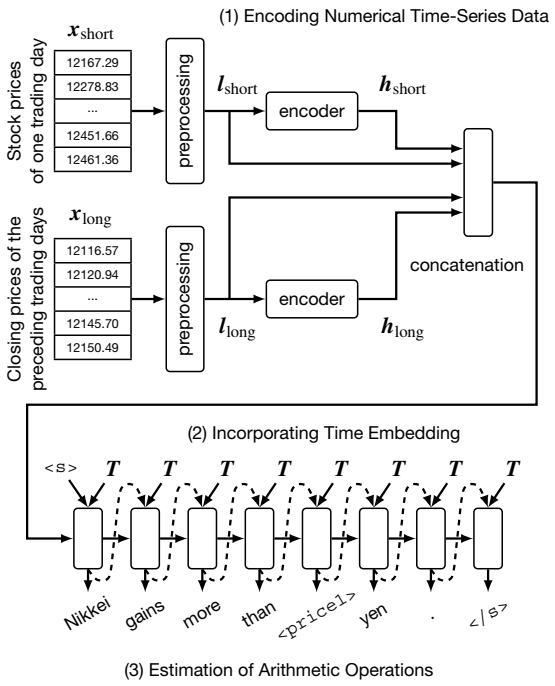


Figure 2: Overview of our model. Here $\boldsymbol{l}_{\text{short}}$ and $\boldsymbol{l}_{\text{long}}$ represent two vectors of preprocessed values, and $\boldsymbol{h}_{\text{short}}$ and $\boldsymbol{h}_{\text{long}}$ indicate hidden states of the encoder. $\boldsymbol{T}$ represents a time embedding vector.

3.1 Encoding Numerical Time-Series Data

We prepare short- and long-term data, using the five-minute chart of Nikkei 225. A vector for short-term data consists of the prices of one trading day and has N elements. We denote it as $\boldsymbol{x}_{\text{short}} = \left(x_{\text{short},\,i}\right)_{i=0}^{N-1}$. On the other hand, a vector for long-term data consists of the closing prices of the M preceding trading days. It is denoted as $\boldsymbol{x}_{\text{long}} = \left(x_{\text{long},\,i}\right)_{i=0}^{M-1}$.

Data are commonly preprocessed to remove noise and enhance generalizability of a model (Zhang and Qi, 2005; Banaee et al., 2013a). We use two preprocessing methods: *standardization* and *moving reference*. Standardization substitutes each element x_i of input $\boldsymbol{x}$ by

$$x_i^{\text{std}} = \frac{x_i - \mu}{\sigma}, \tag{1}$$

where μ and σ are the mean and standard deviation of the values in the training data, respectively. Standardized values are less affected by scale. The second method, moving reference (Freitas et al., 2009), substitutes each element x_i of input $\boldsymbol{x}$ by

$$x_i^{\text{move}} = x_i - r_i, \tag{2}$$

where r_i is the closing price of the previous trading day of $\boldsymbol{x}$. This is introduced to capture price fluctuations from the previous day.

By applying one of the preprocessing methods to $\boldsymbol{x}_{\text{short}}$ and $\boldsymbol{x}_{\text{long}}$, we obtain two vectors of preprocessed values $\boldsymbol{l}_{\text{short}}$ and $\boldsymbol{l}_{\text{long}}$. Given these, each encoder emits the corresponding hidden states $\boldsymbol{h}_{\text{short}}$ and $\boldsymbol{h}_{\text{long}}$. After obtaining the hidden states, we concatenate the two vectors of the preprocessed values and the outputs of the encoders as a *multi-level representation* of the input time-series data. The multi-level representation is an approach developed by Mei et al. (2016a) that enable the decoder to take into account both the high-level representation, e.g., $\boldsymbol{h}_{\text{short}}$, $\boldsymbol{h}_{\text{long}}$, and the low-level representation, e.g., $\boldsymbol{l}_{\text{short}}$, $\boldsymbol{l}_{\text{long}}$, at the same time. They have shown that it improves performance in terms of selecting salient objects in input data. We thus set the initial hidden state $\boldsymbol{s}_0$ of the decoder as

$$\boldsymbol{s}_0 = \boldsymbol{l}_{\text{short}} \oplus \boldsymbol{l}_{\text{long}} \oplus \boldsymbol{h}_{\text{short}} \oplus \boldsymbol{h}_{\text{long}}, \quad (3)$$

where $\oplus$ is the concatenation operator.

When we use both preprocessing methods, we have four preprocessed input vectors: $\boldsymbol{l}^{\text{move}}_{\text{short}}$, $\boldsymbol{l}^{\text{std}}_{\text{short}}$, $\boldsymbol{l}^{\text{move}}_{\text{long}}$, and $\boldsymbol{l}^{\text{std}}_{\text{long}}$. In this case, we introduce four encoders, and set the initial hidden state $\boldsymbol{s}_0$ of the decoder as

$$\begin{aligned}\boldsymbol{s}_0 = {} & \boldsymbol{l}^{\text{move}}_{\text{short}} \oplus \boldsymbol{l}^{\text{std}}_{\text{short}} \oplus \boldsymbol{l}^{\text{move}}_{\text{long}} \oplus \boldsymbol{l}^{\text{std}}_{\text{long}} \\ & \oplus \boldsymbol{h}^{\text{move}}_{\text{short}} \oplus \boldsymbol{h}^{\text{std}}_{\text{short}} \oplus \boldsymbol{h}^{\text{move}}_{\text{long}} \oplus \boldsymbol{h}^{\text{std}}_{\text{long}}. \quad (4)\end{aligned}$$

Since several encoding methods can be used for the time-series data, we use any one of the three conventional neural networks: Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), or Recurrent Neural Network (RNN) with Long Short-Term Memory cells (Hochreiter and Schmidhuber, 1997). In the experiments, we empirically evaluate and compare the encoding methods.

3.2 Incorporating Time Embedding

Even if identical sequences of values are observed, comments usually vary in accordance with price history or the time they are observed. For instance, when the market opens, comments usually mention how much the stock price has increased or decreased compared with the closing price of the previous trading day, as in (1) and (3) in Figure 1.

Our model creates vectors called *time embedding* vectors $\boldsymbol{T}$ on the basis of the time when the comment is delivered (e.g., 9:00 a.m. or 3:00 p.m.). Then a time embedding vector is added to each hidden state $\boldsymbol{s}_j$ in decoding so that words are generated depending on time. This mechanism is inspired by *speaker embedding* introduced by Li et al. (2016). They use an encoder-decoder model for a conversational agent that inherits the characteristics of a speaker, such as his/her manner of speaking. They encode speaker-specific information (e.g., dialect, age, and gender) into speaker embedding vectors and used them in decoding.

3.3 Estimation of Arithmetic Operations

Text generation systems based on language models such as RNNLM often generate erroneous words for named entities; that is, they often mention a similar but incorrect entity, e.g., Nissan for Toyota. To overcome this problem, Gulcehre et al. (2016) developed a text generation method called *copy mechanism*. The method copies rare words missing from the vocabulary from a given sequence of words using an attention mechanism, and emits the copied words.

Market comments often mention numerical values that appear in the input data, but they also mention values obtained through arithmetic operations, such as differences in prices as in (3) and (6) in Figure 1, or rounded values as in (5). Thus, another problem arises: what *type of operation* is suitable for text to be generated? In this work, we solve this problem by extending the idea of copy mechanism.

To enable our model to generate text with values calculated from input values, we add *generalization tags* to the vocabulary used in the model. Each generalization tag represents a type of arithmetic operation. When a generalization tag is emitted, the model performs the operations on the designated values in accordance with the tag, replaces the tag with the calculated value, and finally outputs text containing numerical values. For preprocessing, we replace each numerical value appearing in the market comments in the training data with generalization tags such as `<price1>`. The tag for a numerical value depends on what the value stands for in the text. Table 1 displays all the tags and the corresponding types of calculation. To illustrate, suppose a market comment says

(a) *Nikkei rebounds. The closing price of the morning session is 16,610 yen, which is 227 yen higher.*

Since this comment omits the phrase "than the

Tag	Arithmetic operation
<price1>	Return Δ
<price2>	Round down Δ to the nearest 10
<price3>	Round down Δ to the nearest 100
<price4>	Round up Δ to the nearest 10
<price5>	Round up Δ to the nearest 100
<price6>	Return z as it is
<price7>	Round down z to the nearest 100
<price8>	Round down z to the nearest 1,000
<price9>	Round down z to the nearest 10,000
<price10>	Round up z to the nearest 100
<price11>	Round up z to the nearest 1,000
<price12>	Round up z to the nearest 10,000

Table 1: Generalization tags and corresponding arithmetic operations. Here z and Δ stand for latest price and difference between z and closing price of previous trading day.

closing price of the previous day", 227 in this example indicates the difference between the closing price of the previous trading day $x_{\text{long},\,M-1}$ and the latest price $x_{\text{short},\,N-1}$ denoted by z in Table 1. Therefore, we replace 227 with the tag <price1>. Likewise, we replace 16,610 with <price6> because it represents the latest price z. To find the optimal tag for each value, we try all the types of operations listed in Table 1 using the values appearing in the text, i.e., 227 and 16,610 in this case. Then, we select the tag that has the operation that yields the value closest to the original one.

In prediction, the model first generates a tentative comment, which includes tags as well as words. Suppose that the input vectors are $\boldsymbol{x}_{\text{short}}$ and $\boldsymbol{x}_{\text{long}}$, with $x_{\text{short},\,N-1} = 14508$ and $x_{\text{long},\,M-1} = 14612$, and that the model generates the comment below:

(b) *Nikkei opens turning down. The loss exceeds* <price2> *yen, and it falls to the* <price7> *yen level.*

Since the tag <price2> represents "the difference between $x_{\text{short},\,N-1}$ and $x_{\text{long},\,M-1}$ rounded down to the nearest 10", we replace the tag with 100. Similarly, we replace <price7>, which is "the last price $x_{\text{short},\,N-1}$ rounded down to the nearest 100", with 14,500. Finally, we have a market comment containing the numbers as below:

(c) *Nikkei opens turning down. The loss exceeds 100 yen, and it falls to the 14,500 yen level.*

4 Experiments

4.1 Experimental Settings

We used the five-minute chart of Nikkei 225 from March 2013 to October 2016 as numerical time-series data, which were collected from IBI-Square Stocks[1], and 7,351 descriptions as market comments, which are written in Japanese and provided by Nikkei QUICK News. We divided the dataset into three parts: 5,880 for training, 730 for validation, and 741 for testing. For a human evaluation, we randomly selected 100 comments and their time-series data included in the test set.

We set $N = 62$, which is the number of time steps for stock prices for one trading day, and $M = 7$, which is the number of the time steps for closing prices of the preceding trading days. We used Adam (Kingma and Ba, 2015) for optimization with a learning rate of 0.001 and a mini-batch size of 100. The dimensions of word embeddings, time embeddings, and hidden states for both the encoder and decoder are set to 128, 64, and 256, respectively. For CNN, we used a single convolutional layer and set the filter size to 3.

In the experiments, we conducted three types of evaluation: two for automatic evaluation, and one for human evaluation. For one automatic evaluation, we used BLEU (Papineni et al., 2002) to measure the matching degree between the market comments written by humans as references and output comments generated by our model. We applied paired bootstrap resampling (Koehn, 2004) for a significance test. For the other automatic evaluation metric, we calculate F-measures for time-dependent expressions, using market comments written by humans as references, to investigate whether our model can correctly output time-dependent expressions such as "*open with*" and describe how the price changes compared with the previous period referring to the series of preceding prices such as "*continual fall*". Specifically, we calculate F-measures for 13 expressions shown in Figure 3.

For the human evaluation, we recruited a specialist in financial engineering as a judge to evaluate the quality of generated market comments. To evaluate the difference in the quality of generated comments between our models and human, we showed both system-generated and human-generated market comments together with their

[1] http://www.ibi-square.jp/index.htm

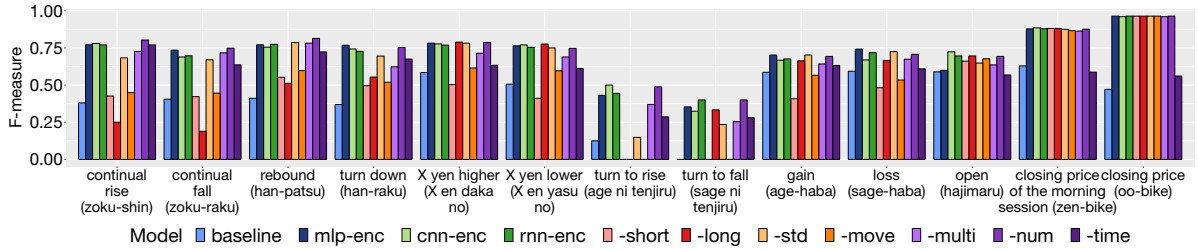


Figure 3: F-measure values for the expressions on the test set. Each expression is accompanied by its original Japanese expression transliterated into English alphabet in parenthesis. Out of the 13 expressions, 10 on the left are expressions that describe how the price changes compared with the previous period, and 3 on the right are time-dependent expressions.

Model		baseline	mlp-enc	cnn-enc	rnn-enc	-short	-long	-std	-move	-multi	-num	-time
Encoder		MLP	MLP	CNN	RNN	MLP	MLP	MLP	MLP	MLP	MLP	MLP
Input data	$\boldsymbol{x}_{\text{short}}$	✓	✓	✓	✓	–	✓	✓	✓	✓	✓	✓
	$\boldsymbol{x}_{\text{long}}$	–	✓	✓	✓	✓	–	✓	✓	✓	✓	✓
Preprocessing	Standardization	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	✓
	Moving reference	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓
	Multi-level	–	✓	✓	✓	✓	✓	✓	✓	–	✓	✓
	Arithmetic operation	–	✓	✓	✓	✓	✓	✓	✓	✓	–	✓
	Time-embedding	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	–

Table 2: Overview of the models we used in the experiments.

time-series data consisting of $\boldsymbol{x}_{\text{short}}$ and $\boldsymbol{x}_{\text{long}}$, without letting the judge know which comment is generated by which method. We asked the judge to give each market comment two scores: one for informativeness and one for fluency. Both scores have two levels, 0 or 1, where 1 indicates high informativeness or fluency. For informativeness, the judge used both generated comments and their input stock prices to rate the comments. Specifically, if the judge deem that a generated comment describes an important price movement or an outline of the movement properly, such comments are considered to be informative. For fluency, the judge read only the generated comments and rate them in terms of readability, regardless of their content of the comment.

In addition, since some of the market comments written by humans sometimes include external information such as "*Nikkei opens with a continual fall as yen pressures exporters*", we also asked the judge to ignore the correctness of external information mentioned in comments, for the sake of fairness in comparison, because external information cannot be retrieved from the time-series data.

To assess the effectiveness of the techniques we introduced, we conducted experiments with 11 models. Table 2 shows an overview of the models we compared. We compared three types of models: a baseline, full models (e.g., *mlp-enc*), and ablated models (e.g., *-short*). For example, *-short* is a model that does not use the short-term time series.

Model	baseline	mlp-enc	cnn-enc	rnn-enc	-short	-long
BLEU	0.243	**0.464**	0.449	0.454	0.380	0.433

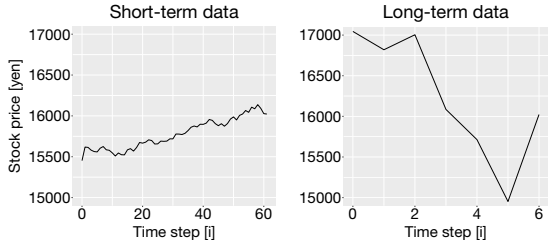
Model	-std	-move	-multi	-num	-time
BLEU	0.455	0.393	0.435	0.318	0.395

Table 3: BLEU scores on the test set. Differences between the best model, *mlp-enc*, and other models are statistically significant at $p < 0.05$.

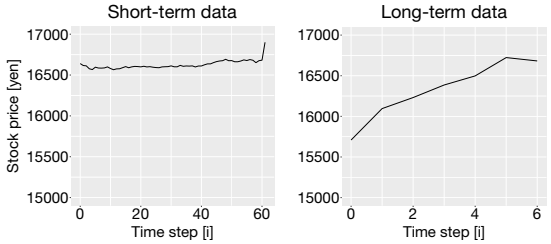
4.2 Results

Table 3 shows the BLEU scores on the test set. Figure 3 presents the F-measure of the models for each phrase. We also present output examples with human-generated market comments (Human) for reference in Figure 4.

In the results for the automatic evaluation in BLEU, the model using both MLP as encoders and all the techniques we developed, *mlp-enc*, outperformed *baseline* and the other models. The BLEU scores and F-measure values revealed differences among the models using MLP, CNN, or RNN (*mlp-*



(a) Price movements of Nikkei on Feb. 15, 2016.



(b) Price movements of Nikkei on July 21, 2016.

Model	F	I	Generated comment
baseline	1	0	Nikkei heikin, han-patsu zen-bike wa 81 en daka no <unk> en Nikkei average, rebound-pop first_half-closing TOP 81 yen higher GEN <unk> yen *Nikkei rebounds. The closing price of the morning session is <unk> yen, which is 81 yen higher.*
mlp-enc	1	1	Nikkei heikin, oo-haba han-patsu oo-bike wa 1,069 en daka no 16,022 en Nikkei average, big-range rebound-pop big-closing TOP 1,069 yen higher GEN 16,022 yen *Nikkei significantly rebounds. The closing price is 16,022 yen, which is 1,069 yen higher.*
human	1	1	Nikkei heikin, oo-haba han-patsu oo-bike wa 1,069 en daka no 16,022 en Nikkei average, big-range rebound-pop big-closing TOP 1,069 yen higher GEN 16,022 yen *Nikkei significantly rebounds. The closing price is 16,022 yen, which is 1,069 yen higher.*

(c) Comments on price at 3:00 p.m. on February 15, 2016.

Model	F	I	Generated comment
baseline	1	0	Nikkei heikin, zoku-shin de hajimaru age-haba 100 en koeru Nikkei average, continual-advance INSTR open-IMPERF raise-range 100 yen exceed-IMPERF *Nikkei opens with a continual rise. The gain exceeds 100 yen.*
mlp-enc	1	1	Nikkei heikin, age-haba 200 en koeru Nikkei average, raise-range 200 yen exceed-IMPERF *Nikkei gains more than 200 yen.*
human	1	1	Nikkei heikin, age-haba 200 en kosu Nikkei average, raise-range 200 yen exceed-IMPERF *Nikkei gains more than 200 yen.*

(d) Comments on price at 9:00 a.m. on July 21, 2016.

Figure 4: Examples of short- and long-term movements of Nikkei, and comments models made on them, where <unk> represents an unknown word. Columns F and I show scores on fluency and informativeness in human evaluation. Each example is accompanied by original Japanese comment transliterated into English alphabet, its literal translation, and the corresponding English sentence. Abbreviations used here are as follows. TOP: topic case, GEN: genitive case, INSTR: instrumental case, and IMPERF: imperfect form of a verb.

enc, *cnn-enc*, *rnn-enc*). In the comparison between the models that took two types of the time-series data $\boldsymbol{x}_{\text{short}}$, $\boldsymbol{x}_{\text{long}}$ as input (e.g., *mlp-enc* or *rnn-enc*) and the models that only used one of them (*-short*, *-long*), the models using both types of data such as *mlp-enc* and *rnn-enc* gained higher BLEU scores than *-short* and *-long*. Also, the models that encoded the two types of time-series data to capture their short- and long-term changes correctly output more expressions that described the changes such as "*turn to rise*", "*continue to fall*", and "*rebound*" than *-short* and *-long* as shown in Figure 3.

According to the comparison between preprocessing methods, *mlp-enc*, which used both standardization and moving reference as preprocessing methods, obtained a higher BLEU score than the models that used neither (*-std*, *-move*). In terms of the F-measure values, *mlp-enc* output phrases mentioning changes more appropriately and therefore achieved the higher values than the other two models as in "turn to rise" or "turn to fall" in Figure 3. Furthermore, we found that the BLEU score of *-multi*, which did not use the multi-level representation of the data, was inferior. In other words, incorporating the multi-level representation along with an output of an encoder into a decoder seems

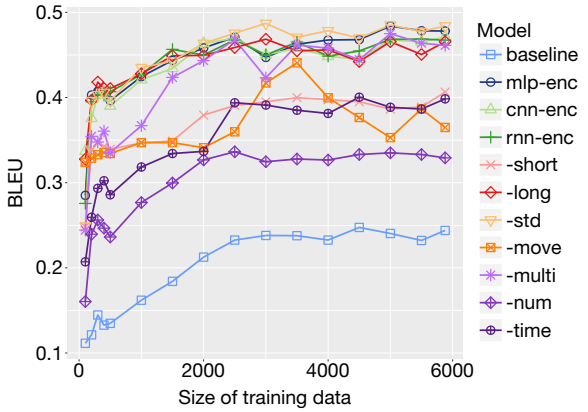


Figure 5: BLEU scores of market comments generated by models for each size of training data on the validation set.

to contribute to improving the automatic evaluation and producing a better representation of the input data.

baseline and *-num* output numerical values as "words" from the vocabulary for RNNLM because these models do not use any arithmetic operation. Therefore, there were many cases including <unk> that should be output as a numerical value as shown in Figure 4 (a). We found that *-num* had a lower BLEU score than the models such as *mlp-enc* and *-std* that used arithmetic operations. Furthermore, we observed that the models with arithmetic operations correctly generated stock prices in most cases.

By comparing *-time*, which did not incorporate time-embeddings into a decoder, and other models such as *mlp-enc* with respect to the F-measure of expressions depending on delivery time (e.g., "*open with*" or "*closing session*"), we found that the models that took time information into account, such as *mlp-enc*, generated those phrases more accurately than *-time*.

Moreover, we analyzed the effect of different sizes of training data. Figure 5 shows BLEU scores of market comments generated by our models for each size of training data on the validation set. According to the results, we found that the BLEU scores for the models saturated when we used 3000 training data. In addition, there was not much difference in convergence speed among the models.

The human evaluation results in Table 4 indicate that market comments generated by our model (*mlp-enc*) achieved a quality comparable even to that of market comments written by humans. Moreover, we found that *mlp-enc* significantly outperformed *baseline* in terms of informativeness but was outperformed by *baseline* in terms of fluency. The reason was that *mlp-enc* occasionally generated a market comment such as "*Nikkei gains more than 0 yen*" because of an error in the prediction of the operation, and such comments were not considered not to be fluent or informative by the judge, although most of comments generated by *mlp-enc* were as fluent as those of *baseline*. Note that *baseline* does not generate expressions like "*0 yen*" because they are not normally used in market comments and so not included in the vocabulary. Therefore, the judge considered all the comments generated by *baseline* to be fluent.

Model	Informativeness	Fluency	External
Human	**95**	95	25
mlp-enc	85	93	1
baseline	28	**100**	6

Table 4: Results of human evaluation. Each score indicates number of market comments judged to be level-1. External shows number of market comments including external information.

For another possibility to enhance our model, we have to consider that the model should mention a difference or gain for a duration from when to when. For example, our current model sometimes generated a market comment such as "*Nikkei gains more than 200 yen*", although Nikkei actually gained more than 300 yen. Such a comment is not incorrect but is imprecise. Therefore, we consider that a mechanism is needed to select the period to be mentioned when the model generates a comment to this problem and increase the generalizability of our model for generating a description from various time-series data.

5 Conclusion and Future Work

In this study, we presented a novel encoder-decoder model to automatically generate market comments from numerical time-series data of stock prices, using the Nikkei Stock Average as an example. Descriptions of numerical time-series data written by humans such as market comments have several writing style characteristics. For example, (1) content to be mentioned in the market comments varies depending on short- or long-term changes of the time-series data, (2) expressions depending on delivery time at which text is written are used, and (3) numerical values obtained through arith-

metic operations applied to the input data are often described. We developed approaches for generating comments that have these characteristics and showed the effectiveness of the proposed model.

In future work, we plan to apply our model to descriptions of time-series data in various domains such as weather forecasts and sports, which share the above writing-style characteristics. We also plan to use multiple time-series as input such as multiple brands of stock.

Acknowledgements

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 502–512. http://aclweb.org/anthology/D10-1049.

Kasumi Aoki and Ichiro Kobayashi. 2016. Linguistic summarization using a weighted n-gram language model based on the similarity of time-series data. In *Proceedings of IEEE International Conference on Fuzzy Systems*. pages 595–601. https://doi.org/10.1109/FUZZ-IEEE.2016.7737741.

Hadi Banaee, Mobyen Uddin Ahmed, and Amy Loutfi. 2013a. A framework for automatic text generation of trends in physiological time series data. In *Processing of IEEE International Conference on Systems, Man, and Cybernetics*. pages 3876–3881. https://doi.org/10.1109/SMC.2013.661.

Hadi Banaee, Mobyen Uddin Ahmed, and Amy Loutfi. 2013b. Towards NLG for physiological data monitoring with body area networks. In *Proceedings of the 14th European Workshop on Natural Language Generation*. Association for Computational Linguistics, pages 193–197. http://aclweb.org/anthology/W13-2127.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. pages 331–338. http://aclweb.org/anthology/H05-1042.

Anja Belz. 2007. Probabilistic generation of weather forecast texts. In *Proceedings of the 2007 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 164–171. http://aclweb.org/anthology/N07-1021.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*. pages 128–135. https://doi.org/10.1145/1390156.1390173.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1724–1734. https://doi.org/10.3115/v1/D14-1179.

Robert Dale, Sabine Geldof, and Jean-Philippe Prost. 2003. CORAL: Using natural language generation for navigational assistance. In *Proceedings of the 26th Australasian Computer Science Conference*. pages 35–44. http://dl.acm.org/citation.cfm?id=783106.783111.

Fabio D. Freitas, Alberto F. De Souza, and Ailson R. de Almeida. 2009. Prediction-based portfolio optimization model using neural networks. *Neurocomputing* 72(10):2155–2170. https://doi.org/10.1016/j.neucom.2008.08.019.

Eli Goldberg, Norbert Driedger, and Richard I. Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert* 9(2):45–53. https://doi.org/10.1109/64.294135.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1631–1640. https://doi.org/10.18653/v1/P16-1154.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 140–149. https://doi.org/10.18653/v1/P16-1014.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Joohyun Kim and Raymond J. Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics*. pages 543–551. http://aclweb.org/anthology/C10-2062.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*. https://arxiv.org/abs/1412.6980.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 388–395. http://aclweb.org/anthology/W04-3250.

Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 752–761. http://aclweb.org/anthology/N12-1093.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1503–1514. http://aclweb.org/anthology/D13-1157.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 145–150. http://aclweb.org/anthology/P83-1022.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1203–1213. https://doi.org/10.18653/v1/D16-1128.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 994–1003. https://doi.org/10.18653/v1/P16-1094.

Percy Liang, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of Association for Computational Linguistics and International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, pages 91–99. http://aclweb.org/anthology/P09-1011.

Wei Lu, Hwee Tou Ng, and Wee Sun Lee. 2009. Natural language generation with tree conditional random fields. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 400–409. http://aclweb.org/anthology/D09-1042.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016a. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of Association for the Advancement of Artificial Intelligence*. https://arxiv.org/abs/1506.04089.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016b. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 720–730. https://doi.org/10.18653/v1/N16-1086.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association*. International Speech Communication Association, 9, pages 1045–1048. http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 311–318. http://aclweb.org/anthology/P02-1040.

François Portet, Ehud Reiter, Albert Gatt, Jim Hunter, Somayajulu Sripada, Yvonne Freer, and Cindy Sykes. 2009. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence* 173(7-8):789–816. https://doi.org/10.1016/j.artint.2008.12.002.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence* 167(1-2):137–169. https://doi.org/10.1016/j.artint.2005.06.006.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 379–389. https://doi.org/10.18653/v1/D15-1044.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. pages 3104–3112. https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural

image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3156–3164. https://arxiv.org/abs/1411.4555.

Yuk Wah Wong and Raymond Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of the 2007 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 172–179. http://aclweb.org/anthology/N07-1022.

G. Peter Zhang and Min Qi. 2005. Neural network forecasting for seasonal and trend time series. *European journal of operational research* 160(2):501–514. https://doi.org/10.1016/j.ejor.2003.08.037.

Can Syntax Help? Improving an LSTM-based Sentence Compression Model for New Domains

Liangguo Wang[†*], **Jing Jiang**[*], **Hai Leong Chieu**[⋆], **Chen Hui Ong**[⋆], **Dandan Song**[†], **Lejian Liao**[†]
chenwangliangguo@bit.edu.cn, jingjiang@smu.edu.sg
{chaileon, ochenhui}@dso.org.sg, {sdd, liaolj}@bit.edu.cn
[†] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[*] School of Information Systems, Singapore Management University, Singapore
[⋆] DSO National Laboratories, Singapore

Abstract

In this paper, we study how to improve the domain adaptability of a deletion-based Long Short-Term Memory (LSTM) neural network model for sentence compression. We hypothesize that syntactic information helps in making such models more robust across domains. We propose two major changes to the model: using explicit syntactic features and introducing syntactic constraints through Integer Linear Programming (ILP). Our evaluation shows that the proposed model works better than the original model as well as a traditional non-neural-network-based model in a cross-domain setting.

1 Introduction

Sentence compression is the task of compressing long, verbose sentences into short, concise ones. It can be used as a component of a text summarization system. Figure 1 shows two example input sentences and the compressed sentences written by human. The task has been studied for almost two decades. Early work on this task mostly relies on syntactic information such as constituency-based parse trees to help decide what to prune from a sentence or how to re-write a sentence (Jing, 2000; Knight and Marcu, 2000). Recently, there has been much interest in applying neural network models to solve the problem, where little or no linguistic analysis is performed except for tokenization (Filippova et al., 2015; Rush et al., 2015; Chopra et al., 2016).

Although neural network-based models have achieved good performance on this task recently, they tend to suffer from two problems: (1) They require a large amount of data for training. For example, Filippova et al. (2015) used close to two million sentence pairs to train an LSTM-based sentence compression model. Rush et al. (2015) used about four million title-article pairs from the Gigaword corpus (Napoles et al., 2012) as training data. Although it may be easy to automatically obtain such training data in some domains (e.g., the news domain), for many other domains, it is not possible to obtain such a large amount of training data. (2) These neural network models trained on data from one domain may not work well on out-of-domain data. For example, when we trained a standard neural sequence-to-sequence model[1] on 3.8 million title-article pairs from the Gigaword corpus and applied it to both in-domain data and out-of-domain data, we found that the performance on in-domain data was good but the performance on out-of-domain data could be very

In-domain

Input: *The southern Chinese city of Guangzhou has set up a special zone allowing foreign consulates to build permanent offices and residences and avoid prohibitive local rents, the china daily reported Tuesday.*
Compressed (by human): *Guangzhou opens new consulate area.*
Compressed (by machine): *Guangzhou sets up special zone for foreign consulates.*

Out-of-domain

Input: *Wherever she was, she helped other loyal and flexible wives cope.*
Compressed (by human): *she helped other wives cope.*
Compressed (by machine): *wives and flexible wives*

Figure 1: Examples of in-domain and out-of-domain results by a standard abstractive sequence-to-sequence model trained on the Gigaword corpus. The first input sentence comes from the Gigaword corpus while the second input sentence comes from the written news corpus used by Clarke and Lapata (2008).

[1] http://opennmt.net/

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1385–1393
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1127

poor. Two example compressed sentences by this trained model are shown in Figure 1 to illustrate the comparison between in-domain and out-of-domain performance.

The two limitations above imply that these neural network-based models may not be good at learning generalizable patterns, or in other words, they tend to overfit the training data. This is not surprising because these models do not explicitly use much syntactic information, which is more general than lexical information.

In this paper, we aim to study how syntactic information can be incorporated into neural network models for sentence compression to improve their domain adaptability. We hope to train a model that performs well on both in-domain and out-of-domain data. To this end, we extend the deletion-based LSTM model for sentence compression by Filippova et al. (2015). Although deletion-based sentence compression is not as flexible as abstractive sentence compression, we chose to work on deletion-based sentence compression for the following reason. Abstractive sentence compression allows new words to be used in a compressed sentence, i.e., words that do not occur in the input sentence. Oftentimes these new words serve as paraphrases of some words or phrases in the source sentence. But to generate such paraphrases, the model needs to have seen them in the training data. Because we are interested in a cross-domain setting, the paraphrases learned in one domain may not work well in another domain if the two domains have very different vocabularies. On the other hand, a deletion-based method does not face such a problem in a cross-domain setting.

Specifically, we propose two major changes to the model by Filippova et al. (2015): (1) We explicitly introduce POS embeddings and dependency relation embeddings into the neural network model. (2) Inspired by a previous method (Clarke and Lapata, 2008), we formulate the final predictions as an Integer Linear Programming problem to incorporate constraints based on syntactic relations between words and expected lengths of the compressed sentences. In addition to the two major changes above, we also use bi-directional LSTM to include contextual information from both directions into the model.

We evaluate our method using around 10,000 sentence pairs released by Filippova et al. (2015) and two other data sets representing out-of-domain data. We test both in-domain and out-of-domain performance. The experimental results showed that our proposed method can achieve competitive performance compared with the original method in the single-domain setting but with much less training data (around 8,000 sentence pairs for training instead of close to two million sentence pairs). In the cross-domain setting, our proposed method can clearly outperform the original method. We also compare our method with a traditional ILP-based method using syntactic structures of sentences but not based on neural networks (Clarke and Lapata, 2008). We find that our method can outperform this baseline for both in-domain and out-of-domain data.

2 Method

In this section, we present our sentence compression method that is aimed at working in a cross-domain setting.

2.1 Problem Definition

Recall that we focus on deletion-based sentence compression. Our problem setup is the same as that by Filippova et al. (2015). Let us use $\boldsymbol{s} = (w_1, w_2, \ldots, w_n)$ to denote an input sentence, which consists of a sequence of words. Here $w_i \in \mathcal{V}$, where $\mathcal{V}$ is the vocabulary. We would like to delete some of the words in $\boldsymbol{s}$ to obtain a compressed sentence that still contains the most important information in $\boldsymbol{s}$. To represent such a compressed sentence, we can use a sequence of binary labels $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$, where $y_i \in \{0, 1\}$. Here $y_i = 0$ indicates that w_i is deleted, and $y_i = 1$ indicates that w_i is retained.

We assume that we have a set of training sentences and their corresponding deletion/retention labels, denoted as $\mathcal{D} = \{(\boldsymbol{s}_j, \boldsymbol{y}_j)\}_{j=1}^{N}$. Our goal is to learn a sequence labeling model from $\mathcal{D}$ so that for any unseen sentence $\boldsymbol{s}$ we can predict its label sequence $\boldsymbol{y}$ and thus compress the sentence.

2.2 Our Base Model

We first introduce our base model, which uses LSTM to perform sequence labeling. This base model is largely based on the model by Filippova et al. (2015) with some differences, which will be explained below.

We assume that each word in the vocabulary has a d-dimensional embedding vector. For input sentence $\boldsymbol{s}$, let us use $(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$ to denote the

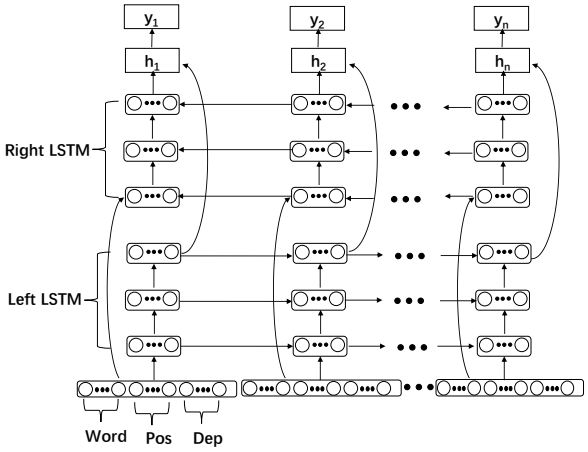


Figure 2: Our three-layered bi-LSTM model. Word embeddings, POS tag embeddings and dependency type embeddings are concatenated in the input layer.

sequence of the word embedding vectors, where $\mathbf{w}_i \in \mathbb{R}^d$. We use a standard bi-directional LSTM model to process these embedding vectors sequentially from both directions to obtain a sequence of hidden vectors $(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$, where $\mathbf{h}_i \in \mathbb{R}^h$. We omit the details of the bi-LSTM and refer the interested readers to the work by Graves et al. (2013) for further explanation. Following Filippova et al. (2015), our bi-LSTM has three layers, as shown in Figure 2.

We then use the hidden vectors to predict the label sequence. Specifically, label y_i is predicted from $\mathbf{h}_i$ as follows:

$$p(y_i \mid \mathbf{h}_i) = \text{softmax}(\mathbf{W}\mathbf{h}_i + \mathbf{b}), \quad (1)$$

where $\mathbf{W} \in \mathbb{R}^{2\times h}$ and $\mathbf{b} \in \mathbb{R}^2$ are a weight matrix and a weight vector to be learned.

There are some differences between our base model and the LSTM model by Filippova et al. (2015). (1) Filippova et al. (2015) first encoded the input sentence in its reverse order using the same LSTM before processing the sentence for sequence labeling. (2) Filippova et al. (2015) used only a single-directional LSTM while we use bi-LSTM to capture contextual information from both directions. (3) Although Filippova et al. (2015) did not use any syntactic information in their basic model, they introduced some features based on dependency parse trees in their advanced models. Here we follow their basic model because later we will introduce more explicit syntax-based features. (4) Filippova et al. (2015) combined the predicted y_{i-1} with $\mathbf{w}_i$ to help predict y_i. This adds some dependency between consecutive labels. We do not do this because later we will introduce an ILP layer to introduce dependencies among labels.

2.3 Incorporation of Syntactic Features

Note that in the base model that we presented above, there is no explicit use of any syntactic information such as the POS tags of the words or the parse tree structures of the sentences. Because we believe that syntactic information is important for learning a generalizable model for sentence compression, we would like to introduce syntactic features into our model.

First, we perform part-of-speech tagging on the input sentences. For sentence $\boldsymbol{s}$, let us use $(t_1, t_2, \ldots, t_n)$ to denote the POS tags of the words inside, where $t_i \in \mathcal{T}$ and $\mathcal{T}$ is a POS tag set. We further assume that each $t \in \mathcal{T}$ has an embedding vector (to be learned). Let us use $(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n)$ ($\mathbf{t}_i \in \mathbb{R}^p$, $p < |\mathcal{T}|$) to denote the sequence of POS embedding vectors of this sentence. We can then simply concatenate $\mathbf{w}_i$ with $\mathbf{t}_i$ as a new vector to be processed by the bi-LSTM model.

Next, we perform dependency parsing on the input sentences. For each word w_i in sentence $\boldsymbol{s}$, let $r_i \in \mathcal{R}$ denote the dependency relation between w_i and its parent word in the sentence, where $\mathcal{R}$ is the set of all dependency relation types. We then assume that each $r \in \mathcal{R}$ has an embedding vector (to be learned). Let $(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n)$ ($\mathbf{r} \in \mathbb{R}^q$, $q < |\mathcal{R}|$) denote corresponding dependency embedding vectors of this sentence. We can also concatenate $\mathbf{w}_i$ with $\mathbf{r}_i$ and feed the new vector to the bi-LSTM model.

In our model, we combine the word embedding, POS embedding and dependency embedding into a single vector to be processed by the bi-LSTM model:

$$\begin{aligned}
\mathbf{x}_i &= \mathbf{w}_i \oplus \mathbf{t}_i \oplus \mathbf{r}_i, \\
\overrightarrow{\mathbf{h}}_i &= \text{LSTM}_{\overrightarrow{\Theta}}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{x}_i), \\
\overleftarrow{\mathbf{h}}_i &= \text{LSTM}_{\overleftarrow{\Theta}}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{x}_i), \\
\mathbf{h}_i &= \overrightarrow{\mathbf{h}}_i \oplus \overleftarrow{\mathbf{h}}_i,
\end{aligned}$$

where $\oplus$ represents concatenation of vectors, and $\overrightarrow{\Theta}$ and $\overleftarrow{\Theta}$ are parameters of the bi-LSTM model. The complete model is shown in Figure 2.

2.4 Global Inference through ILP

Although the method above has explicitly incorporated some syntactic information into the bi-LSTM model, the syntactic information is used in a soft manner through the learned model weights. We hypothesize that there are also hard constraints we can impose on the compressed sentences. For example, the method above as well as the original method by Filippova et al. (2015) cannot impose any length constraint on the compressed sentences. This is because the labels $y_1, y_2, \ldots, y_n$ are not jointly predicted.

We propose to use Integer Linear Programming (ILP) to find an optimal combination of the labels $y_1, y_2, \ldots, y_n$ for a sentence, subject to some constraints. Specifically, the ILP problem consists of two parts: the objective function, and the constraints.

The Objective Function

Recall that the trained bi-LSTM model above produces a probability distribution for each label y_i, as defined in Eqn. (1). Let us use α_i to denote the probability of $y_i = 1$ as estimated by the bi-LSTM model. Intuitively, we would like to set y_i to 1 if α_i is large.

Besides the probability estimated by the bi-LSTM model, here we also consider the depth of the word w_i in the dependency parse tree of the sentence. Intuitively, a word closer to the root of the tree is more likely to be retained. In order to incorporate this observation, we define $dep(w_i)$ to be the depth of the word w_i in the dependency parse tree of the sentence. The root node of the tree has a depth of 0, an immediate child of the root has a depth of 1, and so on. For example, the dependency parse tree of an example sentence together with the depth of each word is shown in Figure 3. We can see that some of the words that are deleted according to the ground truth have a relatively larger depth, such as the first "she" (with a depth of 4) and the word "flexible" (with a depth of 5).

Based on these considerations, we define the objective function to be the following:

$$\max \quad \sum_{i=1}^{n} y_i(\alpha_i - \lambda \cdot dep(w_i)), \qquad (2)$$

where λ is a positive parameter to be manually set, and y_i is the same as defined before, which is either 0 or 1 to indicate whether w_i is deleted or not.

Constraints

We further introduce some constraints to capture tow considerations. The first consideration is related to the syntactic structure of a sentence, and the second consideration is related to the length of the compressed sentence. Some of the constraints are inspired by Clarke and Lapata (2008).

Our constraints are listed below: (1) No missing parent: Generally, we believe that if a word is retained in the compressed sentence, its parent in the dependency parse tree should also be retained. (2) No missing child: For some dependency relations such as *nsubj*, if the parent word is retained, it makes sense to also keep the child word; otherwise the sentence may become ungrammatical. (3) Max length: Since we are trying to compress a sentence, we may need to impose a minimum compression rate. This could be achieved by setting a maximum value of the sum of y_i. (4) Min length: We observe that the original model sometimes produces very short compressed sentences. We therefore believe that it is also important to maintain a mininum length of the compressed sentence. This can be achieved by setting a minimum value of the sum of y_i.

Formally, the constraints are listed as follows:

$$\begin{aligned} \sum_{i=1}^{n} y_i &<= \beta n, \\ \sum_{i=1}^{n} y_i &>= \gamma n, \\ \forall y_i: \quad y_i &\leq y_{p_i}, \\ \forall r_i \in \mathcal{T}': \quad y_i &\geq y_{p_i}, \end{aligned}$$

where w_{p_i} is the parent word of w_i in the dependency parse tree, r_i is the dependency relation type between w_i and w_{p_i}, and $\mathcal{T}'$ is a set of dependency relations for which the child word is often retained when the parent word is retained in the compressed sentence.

The set $\mathcal{T}'$ is derived as follows. For each dependency relation type, based on the training data, we compute the conditional probability of the child word being retained given that the parent word is retained. If this probability is higher than 90%, we include this dependency relation type in $\mathcal{T}'$.

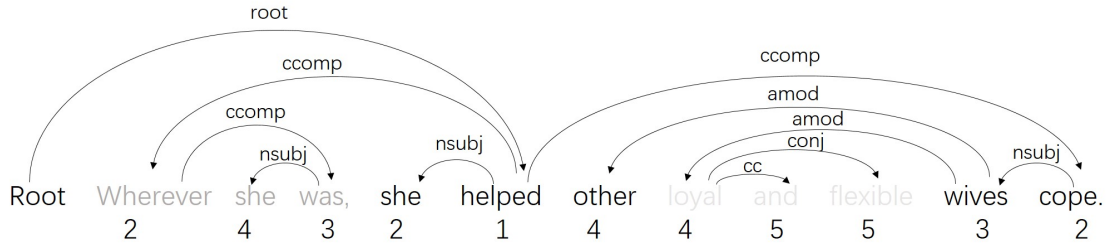


Figure 3: Dependency parse tree of an example sentence. The numbers below the words indicate the depths of the words in the tree. Words in gray are supposed to be deleted based on the ground truth.

3 Experiments

3.1 Datasets and Experiment Settings

Because we are mostly interested in a cross-domain setting where the model is trained on one domain and test on a different domain, we need data from different domains for our evaluation. Here we use three datasets.

***Google News*:** The first dataset contains 10,000 sentence pairs collected and released by Filippova et al. (2015)[2]. The sentences were automatically acquired from the web through Google News using a method introduced by Filippova and Altun (2013). The news articles were from 2013 and 2014.

***BNC News*:** The second dataset contains around 1,500 sentence pairs collected by Clarke and Lapata (2008)[3]. The sentences were from British National Corpus (BNC) and the American News Text corpus before 2008.

***Research Papers*:** The last dataset contains 100 sentences taken from 10 randomly selected papers published at the ACL conference in 2016.

For *Google News* and *BNC News*, we have the ground truth compressed sentences, which are deletion-based compressions, i.e., subsequences of the original sentences. For *Research Papers*, we use it only for manual evaluation in terms of readability and informativeness, as we will explain below.

We evaluate three settings of our method:

BiLSTM: In this setting, we use only the base bi-LSTM model without incorporating any syntactic feature.

BiLSTM+SynFeat: In this setting, we combine word embeddings with POS embeddings and dependency embeddings as input to the bi-LSTM model and use the predictions of $\boldsymbol{y}$ from the bi-LSTM model.

BiLSTM+SynFeat+ILP: In this setting, on top of **BiLSTM+SynFeat**, we solve the ILP problem as described in Section 2.4 to predict the final label sequence $\boldsymbol{y}$.

In the experiments, our model was trained using the Adam (Kingma and Ba, 2015) algorithm with a learning rate initialized at 0.001. The dimension of the hidden layers of bi-LSTM is 100. Word embeddings are initialized from GloVe 100-dimensional pre-trained embeddings (Pennington et al., 2014). POS and dependency embeddings are randomly initialized with 40-dimensional vectors. The embeddings are all updated during training. Dropping probability for dropout layers between stacked LSTM layers is 0.5. The batch size is set as 30. For the ILP part, λ is set to 0.5, β and γ are turned by the validation data and finally they are set to 0.7 and 0.2, respectively. We utilize an open source ILP solver[4] in our method.

We compare our methods with a few baselines:

LSTM: This is the basic LSTM-based deletion method proposed by (Filippova et al., 2015). We report both the performance they achieved using close to two million training sentence pairs and the performance of our re-implementation of their model trained on the 8,000 sentence pairs.

LSTM+: This is advanced version of the model proposed by Filippova et al. (2015), where the authors incorporated some dependency parse tree information into the LSTM model and used the prediction on the previous word to help the prediction on the current word.

Traditional ILP: This is the ILP-based method proposed by Clarke and Lapata (2008). This method does not use neural network models and

[2]Available at http://storage.googleapis.com/sentencecomp/compression-data.json.

[3]Available at http://jamesclarke.net/research/resources/.

[4]gnu.org/software/glpk

is an unsupervised method that relies heavily on the syntactic structures of the input sentences[5].

Abstractive seq2seq: This is an abstractive sequence-to-sequence model trained on 3.8 million Gigaword title-article pairs as described in Section 1.

3.2 Automatic Evaluation

With the two datasets *Google News* and *BNC News* that have the ground truth compressed sentences, we can perform automatic evaluation. We first split the *Google News* dataset into a training set, a validation set and a test set. We took the first 1,000 sentence pairs from *Google News* as the test set, following the same practice as Filippova et al. (2015). We then use 8,000 of the remaining sentence pairs for training and the other 1,000 sentence pairs for validation. For the *NBC News* dataset, we use it only as a test set, applying the sentence compression models trained from the 8,000 sentence pairs from *Google News*.

We use the ground truth compressed sentences to compute accuracy and F1 scores. Accuracy is defined as the percentage of tokens for which the predicted label y_i is correct. F1 scores are derived from precision and recall values, where precision is defined as the percentage of retained words that overlap with the ground truth, and recall is defined as the percentage of words in the ground truth compressed sentences that overlap with the generated compressed sentences.

We report both in-domain performance and cross-domain performance in Table 1. From the table, we have the following observations: (1) For the abstractive sequence-to-sequence model, it was trained on the Gigaword data, so for both *Google News* and *NBC News*, the performance shown is cross-domain performance. We can see that indeed this abstractive method performed poorly in cross-domain settings. (2) In the in-domain setting, with the same amount of training data (8,000), our BiLSTM method with syntactic features (BiLSTM+SynFeat and BiLSTM+SynFeat+ILP) performs similarly to or better than the LSTM+ method proposed by Filippova et al. (2015), in terms of both F1 and accuracy. This shows that our method is comparable to the LSTM+ method in the in-domain setting. (3) In the in-domain setting, even compared with the performance of LSTM+ trained on 2 million sentence pairs, our method trained on 8,000 sentence pairs does not perform substantially worse. (4) In the out-of-domain setting, our BiLSTM+SynFeat and BiLSTM+SynFeat+ILP methods clearly outperform the LSTM and LSTM+ methods. This shows that by incorporating more syntactic features, our methods learn a sentence compression model that is less domain-dependent. (5) The Traditional ILP method also works better than the LSTM and LSTM+ methods in the out-of-domain setting. This is probably because the Traditional ILP method relies heavily on syntax, which is less domain-dependent compared with lexical patterns. But the Traditional ILP method performs worse in the in-domain setting than both the LSTM and LSTM+ methods and our methods.

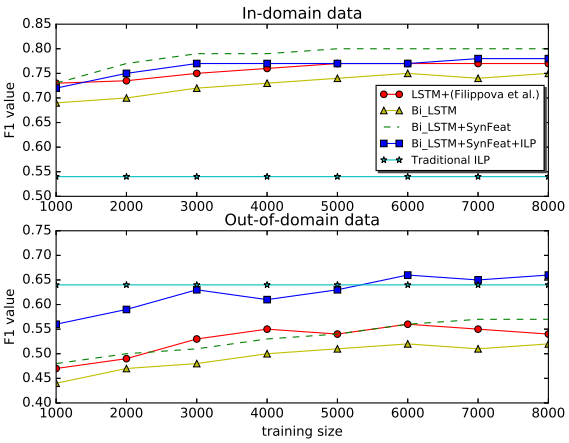


Figure 4: F1 scores with different sizes of training data for in-domain and cross-domain settings.

Overall, Table 1 shows that our proposed method combines both the strength of neural network models in the in-domain setting and the strength of the syntax-based methods in the cross-domain setting. Therefore, our method works reasonably well for both in-domain and out-of-domain data.

We also notice that on *Google News*, adding the ILP layer decreased the sentence compression performance. After some analysis, we think the reason is that some of the constraints used in the ILP layer have led to less deletion but the ground truth compressed sentences in the *Google News* data tend to be shorter compared with those in the *NBC News* data.

We also conduct additional experiments to see the effect of the training data size on our meth-

[5]We use an open source implementation: https://github.com/cnap/sentence-compression.

	size of training data	*Google News* F1	Acc	CR	*NBC News* F1	Acc	CR
LSTM (Filippova et al., 2015)	2 million	0.80	-	0.39	-	-	-
LSTM+ (Filippova et al., 2015)	2 million	**0.82**	-	0.38	-	-	-
Traditional ILP (Clarke and Lapata, 2008)	N/A	0.54	0.56	0.62	0.64	0.56	0.56
Abstractive seq2seq	3.8M	0.09	0.02	0.16	0.14	0.06	0.21
LSTM (our implementation)	8000	0.74	0.75	0.45	0.51	0.48	0.37
LSTM+ (our implementation)	8000	0.77	0.78	0.47	0.54	0.51	0.38
BiLSTM	8000	0.75	0.76	0.43	0.52	0.50	0.34
BiLSTM+SynFeat	8000	0.80	**0.82**	0.43	0.57	0.54	0.37
BiLSTM+SynFeat+ILP	8000	0.78	0.78	0.57	**0.66**	**0.58**	0.53

Table 1: Automatic evaluation of our sentence compression methods. CR standards for compression rate and is defined as the average percentage of words that are retained after compression.

ods and the LSTM+ method. Figure 4 shows the F1 scores on the in-domain *Google News* data and the out-of-domain *NBC News* data when we train the models using different amounts of sentence pairs. We can see that in the in-domain setting, our method does not have any advantage over the LSTM+ method. But in the cross-domain setting, our method that uses ILP to impose syntax-based constraints clearly performs better than LSTM+ when the amount of training data is relatively small.

3.3 Manual Evaluation

The evaluation above does not look at the readability of the compressed sentences. In order to evaluate whether sentences generated by our method are readable, we adopt the manual evaluation procedure by Filippova et al. (2015) to compare our method with LSTM+ and Traditional ILP in terms of readability and informativeness. We asked two raters to score a randomly selected set of 100 sentences from the *Research Papers* dataset. The compressed sentences were randomly ordered and presented to the human raters to avoid any bias. The raters were asked to score the sentences on a five-point scale in terms of both readability and informativeness. We show the average scores of the three methods we compare in Table 3. We can see that our BiLSTM+SynFeat+ILP method clearly outperforms the two baseline methods in the manual evaluation.

We also show a small sample of input sentences from the *Research Papers* dataset and the automatically compressed sentences by different methods in Table 2. As we can see from the table, a general weakness of the LSTM+ method is that the compressed sentences may not be grammatical. In comparison, our method does better in terms of preserving grammaticality.

4 Related Work

Sentence compression can be seen as sentence-level summarization. Similar to document summarization, sentence compression methods can be divided into extractive compression and abstractive compression methods, based on whether words in the compressed sentence all come from the source sentence. In this paper, we focus on deletion-based sentence compression, which is a spacial case of extractive sentence compression.

An early work on sentence compression was done by Jing (2000), who proposed to use several resources to decide whether a phrase in a sentence should be removed or not. Knight and Marcu (2000) proposed to apply a noisy-channel model from machine translation to the sentence compression task, but their model encountered the problem that many SCFG rules have unreliable probability estimates with inadequate data. Galley and McKeown (2007) tried to solve this problem by utilizing parent annotation, Markovization and lexicalization, which have all been shown to improve the quality of the rule probability estimates. Cohn and Lapata (2007) formulated sentence compression as a tree-to-tree rewrite problem. They utilized a synchronous tree substitution grammar (STSG) to license the space of all possible rewrites. Each rule has a weight learned from the training data. For prediction, an algorithm was used to search for the best scoring compression using the gram-

Although dynamic oracles are widely used in dependency parsing and available for most standard transition systems , no dynamic oracle parsing model has yet been proposed for phrase structure grammars **T**: Although are used for transition systems model has been proposed for structure grammars . **S**: Although dynamic oracles are . **B**: Although oracles are used no model has been proposed for structure grammars .
As described above , we used Bayesian Optimization to find optimal hyperparameter configurations in fewer steps than in regular grid search . **T**: As described we used Optimization to find configurations in steps in search . **S**: As described above Optimization to find optimal hyperparameter configurations steps than in grid search . **B**: As described , we used Bayesian Optimization to find optimal hyperparameter configurations in steps.
Following the phrase structure of a source sentence , we encode the sentence recursively in a bottom-up fashion to produce a vector representation of the sentence and decode it while aligning the input phrases and words with the output . **T**: Following structure of a sentence we encode sentence recursively to produce a representation of the sentence and decode it while aligning phrases and words with output . **S**: Following the structure of a source sentence encode the sentence recursively in a bottom-up fashion . **B**: Following the structure , we encode the sentence recursively in a bottom-up fashion to produce a vector representation and decode it .

Table 2: Some input sentences from the *Research Papers* dataset and the automatically compressed sentences using different methods. T: Traditional ILP method. S: LSTM+. B: BiLSTM+SynFeat+ILP.

	readability	informativeness
Traditional ILP	3.94	3.33
LSTM+	3.69	3.07
BiLSTM+SynFeat+ILP	4.29	3.46

Table 3: Manual evaluation.

mar rules. Besides, Cohn and Lapata (2008) extended this model to abstractive sentence compression, which includes substitution, reordering and insertion. McDonald (2006) proposed a graph-based sentence compression method. The general idea is that each word pair in the original sentence has a score. The task then becomes how to find a compressed sentence with a length limit according word pair scores. Their method is similar to graph-based dependency parsing. Clarke and Lapata (2008) first used an ILP framework for sentence compression. In the paper, the author put forward three models. The first model is a language model reformulated by ILP. As the first model treats all the words equally, the second model uses a corpus to learn an importance score for each word and then incorporates it in the ILP model. The Last model, which is based on (McDonald, 2006), replaces the decoder with an ILP model and adds many linguistic constraints such as dependency parsing compared with the previous two ILP models. Filippova and Strube (2008) represented sentences with dependency parse trees and an ILP-based method was used to decide whether the dependencies were preserved or not. Different from most previous work that treats sentence extraction and sentence compression separately, Berg-Kirkpatrick et al. (2011) jointly model the two processes in one ILP problem. Bigrams and subtrees are represented by some features, and feature are learned on some training data. The ILP problem maximizes the coverage of weighted bigrams and deleted subtrees of the summary.

In recent years, neural network models, especially sequence-to-sequence models, have been applied to sentence compression. Our work is based on the deletion-based LSTM model for sentence compression by Filippova et al. (2015). There has also been much interest in applying sequence-to-sequence models for abstractive sentence compression (Rush et al., 2015; Chopra et al., 2016). As we pointed out in Section 1, in a cross-domain setting, abstractive sentence compression may not be suitable.

5 Conclusions

In this paper, we studied how to modify an LSTM model for deletion-based sentence compression so that the model works well in a cross-domain setting. We hypothesized that incorporation of syntactic information into the training of the LSTM model would help. We thus proposed two ways to incorporate syntactic information, one through directly adding POS tag embeddings and dependency type embeddings, and the other through the objective function and constraints of an Integer Linear Programming (ILP) model. The experiments showed that our proposed bi-LSTM model with syntactic features and an ILP layer works

well in both in-domain and cross-domain settings. In comparison, the original LSTM model does not work well in the cross-domain setting, and a traditional ILP method does not work well in the in-domain setting. Therefore, our proposed method is relatively more robust than these baselines. We also manually evaluated the compressed sentences generated by our method and found that the method works better than the baselines in terms of both readability and informativeness.

Acknowledgment

This work is supported by DSO grant DSOCL15223. The work was conducted during the first author's visit to the Singapore Management University.

References

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research* .

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *Joint Meeting of Conference on Empirical Methods in Natural Language and Conference on Computational Natural Language Learning*.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics*.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*.

Michel Galley and Kathleen McKeown. 2007. Lexicalized markov grammars for sentence compression. In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization step one: Sentence compression. In *Proceedings of the 17th National Conference on Artificial Intelligence*.

Ryan T McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of European Chapter of the Association for Computational Linguistics Valencia*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Transductive Non-linear Learning for Chinese Hypernym Prediction

Chengyu Wang[1], **Junchi Yan**[2,1], **Aoying Zhou**[1], **Xiaofeng He**[1*]
[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University
[2] IBM Research – China
chywang2013@gmail.com, yanesta13@163.com
{ayzhou,xfhe}@sei.ecnu.edu.cn

Abstract

Finding the correct hypernyms for entities is essential for taxonomy learning, fine-grained entity categorization, knowledge base construction, etc. Due to the flexibility of the Chinese language, it is challenging to identify hypernyms in Chinese accurately. Rather than extracting hypernyms from texts, in this paper, we present a transductive learning approach to establish mappings from entities to hypernyms in the embedding space directly. It combines linear and non-linear embedding projection models, with the capacity of encoding arbitrary language-specific rules. Experiments on real-world datasets illustrate that our approach outperforms previous methods for Chinese hypernym prediction.

1 Introduction

A hypernym of an entity characterizes the *type* or the *class* of the entity. For example, the word `country` is the hypernym of the entity `Canada`. The accurate prediction of hypernyms benefits a variety of NLP tasks, such as taxonomy learning (Wu et al., 2012; Fu et al., 2014), fine-grained entity categorization (Ren et al., 2016), knowledge base construction (Suchanek et al., 2007), etc.

In previous work, the detection of hypernyms requires lexical, syntactic and/or semantic analysis of relations between entities and their respective hypernyms from a language-specific knowledge source. For example, Hearst (1992) is the pioneer work to extract *is-a* relations from a text corpus based on handcraft patterns. The following-up work mostly focuses on *is-a* relation extraction using automatically generated patterns (Snow et al., 2004; Ritter et al., 2009; Sang and Hofmann, 2009; Kozareva and Hovy, 2010) and relation inference based on distributional similarity measures (Kotlerman et al., 2010; Lenci and Benotto, 2012; Shwartz et al., 2016).

While these approaches have relatively high precision over English corpora, extracting hypernyms for entities is still challenging for Chinese. From the linguistic perspective, Chinese is a lower-resourced language with very flexible expressions and grammatical rules (Wang et al., 2015). For instance, there are no word spaces, explicit tenses and voices, and distinctions between singular and plural forms in Chinese. The order of words can be changed flexibly in sentences. Hence, as previous research indicates, hypernym extraction methods for English are not necessarily suitable for the Chinese language (Fu et al., 2014; Wang et al., 2015; Wang and He, 2016).

Based on such conditions, several classification methods are proposed to distinguish *is-a* and *not-is-a* relations based on Chinese encyclopedias (Lu et al., 2015; Li et al., 2015). Similar to Princeton WordNet, a few Chinese wordnets have also been developed (Huang et al., 2004; Xu et al., 2008; Wang and Bond, 2013). The most recent approaches for Chinese *is-a* relation extraction (Fu et al., 2014; Wang and He, 2016) use word embedding based linear projection models to map embeddings of hyponyms to those of their hypernyms, which outperform previous algorithms.

However, we argue that these projection-based methods may have three potential limitations: (i) Only positive *is-a* relations are used for projection learning. The distinctions between *is-a* and *not-is-a* relations in the embedding space are not modeled. (ii) These methods lack the capacity to encode linguistic rules, which are designed by linguists and usually have high precision. (iii) It assumes that the linguistic regularities of *is-a* rela-

*Corresponding author.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1394–1404
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1128

tions can be solely captured by single or multiple linear projection models.

In this paper, we address these limitations by a two-stage transductive learning approach. It distinguishes *is-a* and *not-is-a* relations given a Chinese word/phrase pair as input. In the initial stage, we train linear projection models on positive and negative training data separately and predict *is-a* relations jointly. In the transductive learning stage, the initial prediction results, linguistic rules and the non-linear mappings from entities to hypernyms are optimized simultaneously in a unified framework. This optimization problem can be efficiently solved by blockwise gradient descent. We evaluate our method over two public datasets and show that it outperforms state-of-the-art approaches for Chinese hypernym prediction.

The rest of this paper is organized as follows. We summarize the related work in Section 2. Our approach is introduced in Section 3. Experimental results are presented in Section 4. We conclude our paper in Section 5.

2 Related Work

In this section, we overview the related work on hypernym prediction and discuss the challenges of Chinese hypernym detection.

Pattern based methods identify *is-a* relations from texts by handcraft or automatically generated patterns. Hearst patterns (Hearst, 1992) are lexical patterns in English that are employed to extract *is-a* relations for taxonomy construction (Wu et al., 2012). Automatic approaches mostly use iterative learning paradigms such that the system learns new *is-a* relations and patterns simultaneously. A few relevant studies can be found in (Caraballo, 1999; Etzioni et al., 2004; Sang, 2007; Pantel and Pennacchiotti, 2006; Kozareva and Hovy, 2010). To avoid "semantic drift" in iterations, Snow et al. (2004) train a hypernym classifier based on syntactic features based on parse trees. Carlson et al. (2010) exploit multiple learners to extract relations via coupled learning. These approaches are not effective for Chinese for two reasons: i) Chinese *is-a* relations are expressed in a highly flexible manner (Fu et al., 2014) and ii) the accuracy of basic NLP tasks such as dependency parsing still need improvement for Chinese (Li et al., 2013).

Inference based methods take advantage of distributional similarity measures (DSM) to infer relations between words. They assume that a hypernym may appear in all contexts of the hyponyms and a hyponym can only appear in part of the contexts of its hypernyms. In previous work, Kotlerman et al. (2010) design directional DSMs to model the asymmetric property of *is-a* relations. Other DSMs are introduced in (Bhagat et al., 2007; Szpektor et al., 2007; Lenci and Benotto, 2012; Santus et al., 2014). Shwartz et al. (2016) combine dependency parsing and DSM to improve the performance of hypernymy detection. The reason why DSM is not effective for Chinese is that the contexts of entities in Chinese are flexible and sparse.

Encyclopedia based methods take encyclopedias as knowledge sources to construct taxonomies. Ponzetto and Strube (2007) design features from multiple aspects to predict *is-a* relations between entities and categories in English Wikipedia. The taxonomy in YAGO (Suchanek et al., 2007) is constructed by linking conceptual categories in Wikipedia to WordNet synsets (Miller, 1995). For Chinese, Li et al. (2015) propose an SVM-based approach to build a large Chinese taxonomy from Wikipedia. Similar classification based algorithms are presented in (Fu et al., 2013; Lu et al., 2015). Due to the lack of Chinese version of WordNet, several Chinese semantic dictionaries have been conducted, such as Sinica BOW (Huang et al., 2004), SEW (Xu et al., 2008), COW (Wang and Bond, 2013), etc. These approaches have higher accuracy than mining hypernym relations from texts directly. However, they heavily rely on existing knowledge sources and are difficult to extend to different domains.

To tackle these challenges, **word embedding based methods** directly model the task of hypernym prediction as learning a mapping from entity vectors to their respective hypernym vectors in the embedding space. The vectors can be pretrained by neural language models (Mikolov et al., 2013). For the Chinese language, Fu et al. (2014) train piecewise linear projection models based on a Chinese thesaurus. The state-of-the-art method (Wang and He, 2016) combines an iterative learning procedure and Chinese Hearst-style patterns to improve the performance of projection models. They can reduce data noise by avoiding direct parsing of Chinese texts, but still capture the linguistic regularities of *is-a* relations based on word embeddings. Additionally, several work aims to study how to combine word embeddings for re-

lation classification, such as (Mirza and Tonelli, 2016). In our paper, we extend these approaches by modeling non-linear mappings from entities to hypernyms and adding linguistic rules via a unified transductive learning framework.

3 Proposed Approach

This section begins with a brief overview of our approach. After that, the detailed steps and the learning algorithm are introduced in detail.

3.1 Overview

Given a word/phrase pair (x_i, y_i), the goal of our task is to learn a classification model to predict whether y_i is the hypernym of x_i.

As illustrated in Figure 1, our approach has two stages: initial stage and transductive learning stage. The input is a positive *is-a* set D^+, a negative *is-a* set D^- and an unlabeled set D^U, all of which are the collections of word/phrase pairs.

Denote $\mathbf{x}_i$ as the embedding vector of word x_i, pre-trained and stored in a lookup table. In the initial stage, we train a linear projection model over D^+ such that for each $(x_i, y_i) \in D^+$, a projection matrix maps the entity vector $\mathbf{x}_i$ to its hypernym vector $\mathbf{y}_i$. A similar model is also trained over D^-. Based on the two models, we estimate the prediction score and the confidence score for each $(x_i, y_i) \in D^U$. In the transductive learning stage, a joint optimization problem is formed to learn the final prediction score for each $(x_i, y_i) \in D^U$. It aims to minimize the prediction errors based on the human labeled data, the initial model prediction and linguistic rules. It also employs non-linear mappings to capture linguistic regularities of *is-a* relations other than linear projections.

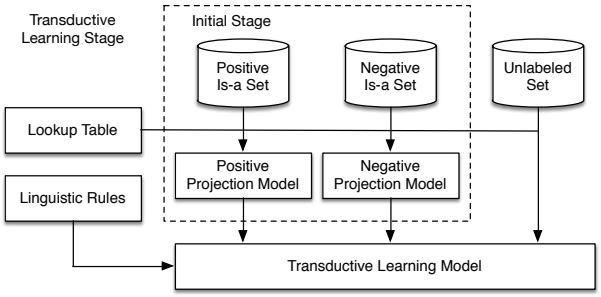


Figure 1: General framework of our approach.

3.2 Initial Model Training

The initial stage models how entities are mapped to their hypernyms or non-hypernyms by projection learning. We first train a *Skip-gram* model (Mikolov et al., 2013) to learn word embeddings over a large text corpus. Inspired by (Fu et al., 2014; Wang and He, 2016), for each $(x_i, y_i) \in D^+$, we assume there is a positive projection model such that $\mathbf{M}^+\mathbf{x}_i \approx \mathbf{y}_i$ where $\mathbf{M}^+$ is an $|\mathbf{x}_i| \times |\mathbf{x}_i|$ projection matrix[1]. However, this model does not capture the semantics of *not-is-a* relations. Thus, we learn a negative projection model $\mathbf{M}^-\mathbf{x}_i \approx \mathbf{y}_i$ where $(x_i, y_i) \in D^-$. This approach is equivalent to learning two separate translation models within the same semantic space. For parameter estimation, we minimize the two following objectives:

$$J(\mathbf{M}^+) = \frac{1}{2} \sum_{(x_i,y_i)\in D^+} \|\mathbf{M}^+\mathbf{x}_i - \mathbf{y}_i\|_2^2 + \frac{\lambda}{2}\|\mathbf{M}^+\|_F^2$$

$$J(\mathbf{M}^-) = \frac{1}{2} \sum_{(x_i,y_i)\in D^-} \|\mathbf{M}^-\mathbf{x}_i - \mathbf{y}_i\|_2^2 + \frac{\lambda}{2}\|\mathbf{M}^-\|_F^2$$

where $\lambda > 0$ is a Tikhonov regularization parameter (Golub et al., 1999).

In the testing phase, for each $(x_i, y_i) \in D^U$, denote $d^+(x_i, y_i) = \|\mathbf{M}^+\mathbf{x}_i - \mathbf{y}_i\|_2$ and $d^-(x_i, y_i) = \|\mathbf{M}^-\mathbf{x}_i - \mathbf{y}_i\|_2$. The *prediction score* is defined as:

$$score(x_i, y_i) = \tanh(d^-(x_i, y_i) - d^+(x_i, y_i))$$

where $score(x_i, y_i) \in (-1, 1)$. Higher prediction score indicates there is a larger probability of an *is-a* relation between x_i and y_i. We choose the hyperbolic tangent function rather than the sigmoid function to avoid the widespread saturation of sigmoid function (Menon et al., 1996). Because the semantics of Chinese is-a and not-is-a relations are complicated and difficult to model (Fu et al., 2014), we do not impose explicit connections between $\mathbf{M}^+$ and $\mathbf{M}^-$ and let the algorithm learn the parameters automantically.

The difference between $d^+(x_i, y_i)$ and $d^-(x_i, y_i)$ can be also used to indicate whether the models are confident enough to make a prediction.

[1] We have also examined piecewise linear projection models proposed in (Fu et al., 2014; Wang and He, 2016) as the initial models for transductive learning. However, we found that this practice is less efficient and the performance does not improve significantly.

In this paper, we calculate the *confidence score* as:

$$conf(x_i, y_i) = \frac{|d^+(x_i, y_i) - d^-(x_i, y_i)|}{\max\{d^+(x_i, y_i), d^-(x_i, y_i)\}}$$

where $conf(x_i, y_i) \in (0, 1)$. Higher confidence score means that there is a larger probability that the models can predict whether there is an *is-a* relation between x_i and y_i correctly. This score gives different data instances different weights in the transductive learning stage.

3.3 Transductive Non-linear Learning

Although linear projection methods are effective for Chinese hypernym prediction, it does not encode non-linear transformation and only leverages the positive data. We present an optimization framework for non-linear mapping utilizing both labeled and unlabeled data and linguistic rules by transductive learning (Gammerman et al., 1998; Chapelle et al., 2006).

Let F_i be the final prediction score of the word/phrase pair (x_i, y_i). In the initialization stage of our algorithm, we set $F_i = 1$ if $(x_i, y_i) \in D^+$, $F_i = -1$ if $(x_i, y_i) \in D^-$ and set F_i randomly in $(-1, 1)$ if $(x_i, y_i) \in D^U$. In matrix representation, denote $\mathbf{F}$ as the $m \times 1$ final prediction vector where $m = |D^+| + |D^-| + |D^U|$. F_i is the ith element in $\mathbf{F}$. The three components in our transductive learning model are as follows:

3.3.1 Initial Prediction

Denote $\mathbf{S}$ as an $m \times 1$ initial prediction vector. We set $S_i = 1$ if $(x_i, y_i) \in D^+$, $S_i = -1$ if $(x_i, y_i) \in D^-$ and $S_i = score(x_i, y_i)$ if $(x_i, y_i) \in D^U$. In order to encode the confidence of model prediction, we define $\mathbf{W}$ as an $m \times m$ diagonal weight matrix. The element in the ith row and the jth column of $\mathbf{W}$ is set as follows:

$$W_{i,j} = \begin{cases} conf(x_i, y_i) & i = j, (x_i, y_i) \in D^U \\ 1 & i = j, (x_i, y_i) \in D^+ \cup D^- \\ 0 & \text{Otherwise} \end{cases}$$

The objective function is defined as: $\mathscr{O}_s = \|\mathbf{W}(\mathbf{F} - \mathbf{S})\|_2^2$, which encodes the hypothesis that the final prediction should be similar to the initial prediction for unlabeled data or human labeling for training data. The weight matrix $\mathbf{W}$ gives the largest weight (i.e., 1) to all the pairs in $D^+ \cup D^-$ and a larger weight to the pair $(x_i, y_i) \in D^U$ if the initial prediction is more confident.

3.3.2 Linguistic Rules

Although linguistic rules can only cover a few circumstances, they are effective to guide the learning process. For Chinese hypernym prediction, Li et al. (2015) study the word formation of conceptual categories in Chinese Wikipedia. In our model, let C be the collection of linguistic rules. γ_i is the true positive (or negative) rate with respect to the respective positive (or negative) rule $c_i \in C$, estimated over the training set. Considering the word formation of Chinese entities and hypernyms, we design one positive rule (i.e., P1) and two negative rules (i.e., N1 and N2), shown in Table 1.

Let $\mathbf{R}$ be an $m \times 1$ linguistic rule vector and R_i is the ith element in $\mathbf{R}$. For training data, we set $R_i = 1$ if $(x_i, y_i) \in D^+$ and $R_i = -1$ if $(x_i, y_i) \in D^-$, which follows the same settings as those in $\mathbf{S}$. For unlabeled pairs that do not match any linguistic rules in C, we update $R_i = F_i$ in each iteration of the learning process, meaning no loss for errors imposed in this part.

For other conditions, denote $C_{(x_i,y_i)} \subseteq C$ as the collection of rules that (x_i, y_i) matches. If $C_{(x_i,y_i)}$ are positive rules, we set R_i as follows:

$$R_i = \max\{F_i, \max_{c_j \in C_{(x_i,y_i)}} \gamma_j\}$$

Similarly, if $C_{(x_i,y_i)}$ are negative rules, we have:

$$R_i = -\max\{-F_i, \max_{c_j \in C_{(x_i,y_i)}} \gamma_j\}$$

which means F_i receives a penalty only if $F_i < \max_{c_j \in C_{(x_i,y_i)}} \gamma_j$ for pairs that match positive rules or $F_i > -\max_{c_j \in C_{(x_i,y_i)}} \gamma_j$ for negative rules[2]. The objective function is: $\mathscr{O}_r = \|\mathbf{F} - \mathbf{R}\|_2^2$. In this way, our model can integrate arbitrary "soft" constraints, making it robust to false positives or negatives introduced by these rules.

3.3.3 Non-linear Learning

TransLP is a transductive label propagation framework (Liu and Yang, 2015) for link prediction, previously used for applications such as text classification (Xu et al., 2016). In our work, we extend their work for our task, modeling non-linear mappings from entities to hypernyms.

[2]We do not consider the cases where a pair matches both positive and negative rules because such cases are very rare, and even non-existent in our datasets. However, our method can deal with these cases by using some simple heuristics. For example, we can update R_i using either of the following two ways: i) $R_i = F_i$ and ii) $R_i = F_i + \sum_{c_j \in C_{(x_i,y_i)}} \gamma_j$.

P1	The head word of the entity x matches that of the candidate hypernym y. For example, 动物 (Animal) is the correct hypernym of 哺乳动物 (Mammal).
N1	The head word of the entity x matches the non-head word of the candidate hypernym y. For example, 动物学 (Zoology) is not a hypernym of 哺乳动物 (Mammal).
N2	The head word of the candidate hypernym y matches an entry in a Chinese lexicon extended based on the lexicon used in Li et al. (2015). It consists of 184 non-taxonomic, thematic words such as 政治 (Politics), 军事 (Military), etc.

Table 1: Three linguistic rules used in our work for Chinese hypernym prediction.

For *is-a* relations, we find that if y is the hypernym of x, it is likely that y is the hypernym of entities that are semantically close to x. For example, if we know `United States` is a `country`, we can infer `country` is the hypernym of similar entities such as `Canada`, `Australia`, etc. This intuition can be encoded in the similarity of the two pairs $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$:

$$\text{sim}(p_i, p_j) = \begin{cases} \cos(\mathbf{x}_i, \mathbf{x}_j) & y_i = y_j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $\mathbf{x}_i$ is the embedding vector of x_i[3].

This similarity indicates there exists a non-linear mapping from entities to hypernyms, which can not be encoded in linear projection based methods (Fu et al., 2014; Wang and He, 2016). Based on *TransLP* (Liu and Yang, 2015), this intuition can be model as propagating class labels (*is-a* or *not-is-a*) of labeled word/phrase pairs to similar unlabeled ones based on Eq. (1). For example, the score of *is-a* relations between `United State` and `country` will propagate to pairs such as `(Canada, country)` and `(Australia, country)` by random walks.

Denote $\mathbf{F}^*$ as the optimal solution of the problem $\min \mathscr{O}_s + \mathscr{O}_r$. Inspired by (Liu and Yang, 2015; Xu et al., 2016), we can add a Gaussian prior $N(\mathbf{F}^*, \mathbf{\Sigma})$ to $\mathbf{F}$ where $\mathbf{\Sigma}$ is the covariance matrix and $\Sigma_{i,j} = \text{sim}(p_i, p_j)$. Hence the optimization objective of this part is defined as: $\mathscr{O}_n = \mathbf{F}^T\mathbf{\Sigma}^{-1}\mathbf{F}$ which is linearly proportional to the negative likelihood of the Gaussian random field prior. This means we minimize the training error and encourage $\mathbf{F}$ to have a smooth propagation with respect to the similarities among pairs defined by Eq. (1) at the same time.

[3]We only consider the similarity between entities and not candidate hypernyms because the similar rule for candidate hypernyms is not true. For example, nouns close to `country` in our *Skip-gram* model are `region`, `department`, etc. They are not all correct hypernyms of `United States`, `Canada`, `Australia`, etc.

3.3.4 Joint Optimization

By combining the three components together, we minimize the following function:

$$J(\mathbf{F}) = \mathscr{O}_s + \mathscr{O}_r + \frac{\mu_1}{2}\mathscr{O}_n + \frac{\mu_2}{2}\|\mathbf{F}\|_2^2 \quad (2)$$

where $\|\mathbf{F}\|_2^2$ imposes an additional smooth l_2-regularization on $\mathbf{F}$. μ_1 and μ_2 are regularization parameters that can be tuned manually.

Based on the convexity of the optimization problem, we can learn the optimal values of $\mathbf{F}$ is via gradient descent. The derivative of $\mathbf{F}$ with respect to $J(\mathbf{F})$ is:

$$\frac{\mathrm{d}J(\mathbf{F})}{\mathrm{d}\mathbf{F}} = \mathbf{W}^2(\mathbf{F}-\mathbf{S})+(\mathbf{F}-\mathbf{R})+\mu_1\mathbf{\Sigma}^{-1}\mathbf{F}+\mu_2\mathbf{F}$$

which is computationally expensive when m is large. After $\mathbf{W}^2$, $\mathbf{S}$, $\mathbf{R}$ and $\mathbf{\Sigma}^{-1}$ are pre-computed, the runtime complexity of the loop of gradient descent is $O(tm^2)$ where t is the number of iterations.

To speed up the learning process, we introduce a *blockwise gradient descent* technique. From the definition of Eq. (2), we can see that the optimal values of F_i and F_j with respect to (x_i, y_i) and (x_j, y_j) are irrelevant if $y_i \neq y_j$. Therefore, the original optimization problem can be decomposed and solved separately according to different candidate hypernyms.

Let H be the collection of candidate hypernyms in D^U. For each $h \in H$, denote D_h as the collection of word/phase pairs in $D^+ \cup D^- \cup D^U$ that share the same candidate hypernym h. The original problem can be decomposed into $|H|$ optimization subproblems over D_h for each $h \in H$. Denote $\mathbf{W}_h$, $\mathbf{S}_h$, $\mathbf{R}_h$, $\mathbf{F}_h$ and $\mathbf{\Sigma}_h$ as the weight matrix, the initial prediction vector, the rule prediction vector, the final prediction vector and the entity similarity covariance matrix with respect D_h. The objective function can be rewritten as:

$J(\mathbf{F}) = \sum_{h \in H} \tilde{J}(\mathbf{F}_h)$ where

$$\tilde{J}(\mathbf{F}_h) = \|\mathbf{W}_h(\mathbf{F}_h - \mathbf{S}_h)\|_2^2 + \|\mathbf{F}_h - \mathbf{R}_h\|_2^2 + \frac{\mu_1}{2}\mathbf{F}_h^T \mathbf{\Sigma}_h^{-1} \mathbf{F}_h + \frac{\mu_2}{2}\|\mathbf{F}_h\|_2^2$$

We additionally use $^{(n)}$ to denote the values of matrices or vectors in the nth iteration. $\mathbf{F}_h^{(n)}$ is iteratively updated based on the following equation:

$$\mathbf{F}_h^{(n+1)} = \mathbf{F}_h^{(n)} - \eta \cdot \frac{\mathrm{d}\tilde{J}(\mathbf{F}_h^{(n)})}{\mathrm{d}\mathbf{F}_h^{(n)}}$$

where η is the learning rate. To this end, we present the learning algorithm in Algorithm 1.

Algorithm 1 Learning Algorithm

1: Initialize $\mathbf{W}_h$ and $\mathbf{S}_h$ based on the initial prediction model;
2: Randomly initialize $\mathbf{F}_h^{(0)}$;
3: Compute $\mathbf{\Sigma}_h^{-1}$ based on entity similarities;
4: Initialize counter $n = 1$;
5: **for** each linguistic rule $c_i \in C$ **do**
6: Estimate γ_i over the training set;
7: **end for**
8: **while** $\|\mathbf{F}_h^{(n)} - \mathbf{F}_h^{(n+1)}\|_2 < 10^{-3}$ **do**
9: Compute $\mathbf{R}_h^{(n)}$ based on C and $\mathbf{F}_h^{(n)}$;
10: Calculate $\frac{\mathrm{d}\tilde{J}(\mathbf{F}_h^{(n)})}{\mathrm{d}\mathbf{F}_h^{(n)}} = \mathbf{W}_h^2(\mathbf{F}_h^{(n)} - \mathbf{S}_h) + (\mathbf{F}_h^{(n)} - \mathbf{R}_h^{(n)}) + \mu_1 \mathbf{\Sigma}_h^{-1}\mathbf{F}_h^{(n)} + \mu_2 \mathbf{F}_h^{(n)}$;
11: Compute $\mathbf{F}_h^{(n+1)}$ for the next iteration: $\mathbf{F}_h^{(n+1)} = \mathbf{F}_h^{(n)} - \eta \cdot \frac{\mathrm{d}\tilde{J}(\mathbf{F}_h^{(n)})}{\mathrm{d}\mathbf{F}_h^{(n)}}$;
12: Update counter $n = n + 1$;
13: **end while**
14: **return** Final prediction vector $\mathbf{F}_h^{(n+1)}$;

The runtime complexity of this algorithm is $O(\sum_{h \in D_h} t_h |D_h|^2)$ where t_h is the number of iterations to solve the subproblem over D_h. Although we do not know the upper bounds on the numbers of iterations of these two learning techniques, the runtime complexity can be reduced by blockwise gradient descent for two reasons: i) $\sum_{h \in D_h} |D_h| \leq m$ and ii) t_h has a large probability to be smaller than t due to the smaller number of data instances. This technique can be also viewed as optimizing Eq. (2) based on blockwise matrix computation.

Finally, for each $(x_i, y_i) \in D^U$, we predict that y_i is a hypernym of x_i if $F_i > \theta$ where $\theta \in (-1, 1)$ is a threshold tuned on the development set.

4 Experiments

In this section, we conduct experiments to evaluate our method. Section 4.1 to Section 4.5 report the experimental steps on Chinese datasets. We present the performance on English datasets in Section 4.6 and a discussion in Section 4.7.

4.1 Experimental Data

We have two collections of Chinese word/phase pairs as ground truth datasets. Each pair is labeled with an *is-a* or *not-is-a* tag. The first one (denoted as *FD*) is from Fu et al. (2014), containing 1,391 *is-a* pairs and 4,294 *not-is-a* pairs, which is the first publicly available dataset to evaluate this task. The second one (denoted as *BK*) is larger in size and crawled from Baidu Baike by ourselves, consisting of <entity, category> pairs. For each pair in *BK*, we ask multiple human annotators to label the tag and discard the pair with inconsistent labels by different annotators. In total, it contains 3,870 *is-a* pairs and 3,582 *not-is-a* pairs[4].

The Chinese text corpus is extracted from the contents of 1.2M entity pages from Baidu Baike[5], a Chinese online encyclopedia. It contains approximately 1.1B words. We use the open source toolkit *Ansj*[6] for Chinese word segmentation. Chinese words/phrases in our test sets may consist of multiple Chinese characters. We treat such word/phrase as a whole to learn embeddings, instead of using character-level embeddings.

In the following experiments, we use 60% of the data for training, 20% for development and 20% for testing, partitioned randomly. By rotating the 5-fold subsets of the datasets, we report the performance of each method on average.

4.2 Parameter Analysis

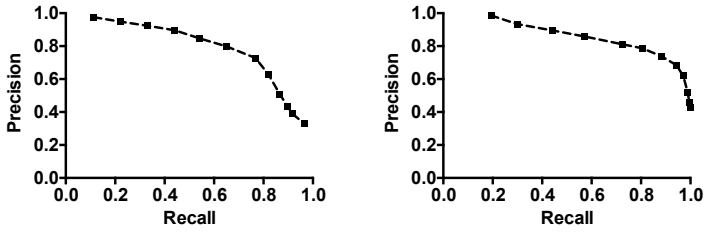
The word embeddings are pre-trained by ourselves on the Chinese corpus. In total, we obtain the 100-dimensional embedding vectors of 5.8M distinct words. The regularization parameters are set to $\lambda = 10^{-3}$ and $\mu_1 = \mu_2 = 10^{-4}$, fine tuned on the development set.

The choice of θ reflects the precision-recall trade-off in our model. A larger value of θ means we pay more attention to precision rather than recall. Figure 2 illustrates the precision-recall curves

[4] https://chywang.github.io/data/acl17.zip
[5] https://baike.baidu.com/
[6] https://github.com/NLPchina/ansj_seg/

Dataset	FD			BK		
Method	P	R	F	P	R	F
Fu et al. (2014) (S)	64.1	56.0	59.8	71.4	64.8	67.9
Fu et al. (2014) (P)	66.4	59.3	62.6	72.7	67.5	70.0
Li et al. (2015)	54.3	38.4	45.0	61.2	47.5	53.5
Mirza and Tonelli (2016) (C)	67.7	**75.2**	69.7	80.3	75.9	78.0
Mirza and Tonelli (2016) (A)	65.3	60.7	62.9	72.7	65.6	68.9
Mirza and Tonelli (2016) (S)	71.9	60.6	65.7	78.4	60.7	68.4
Wang and He (2016)	69.3	64.5	66.9	73.9	69.8	71.8
Ours (Initial)	70.7	69.2	69.9	81.7	78.5	80.0
Ours	**72.8**	70.5	**71.6**	**83.6**	**80.6**	**82.1**

Table 2: Performance comparison on test sets for Chinese hypernym prediction (%).



(a) Dataset: FD (b) Dataset: BK

Figure 2: Precision-recall curve with respect to the tuning of θ on development sets.

on both datasets. It can be seen that the performance of our method is generally better in *BK* than *FD*. The most probable cause is that *BK* is a large dataset with more "balanced" numbers of positive and negative data. Finally, θ is set to 0.05 on *FD* and 0.1 on *BK*.

4.3 Performance

In a series of previous work (Fu et al., 2013, 2014; Wang and He, 2016), several pattern-based, inference-based and encyclopedia-based *is-a* relation extraction methods for English have been implemented for the Chinese language. As their experiments show, these methods achieve the F-measure of lower than 60% in most cases, which are not suggested to be strong baselines for Chinese hypernym prediction. Interested readers may refer to their papers for the experimental results.

To make the convincing conclusion, we employ two recent state-of-the-art approaches for Chinese *is-a* relation identification (Fu et al., 2014; Wang and He, 2016) as baselines. We also take the word embedding based classification approach (Mirza and Tonelli, 2016)[7] and Chinese Wikipedia based SVM model (Li et al., 2015) as baselines to predict *is-a* relations between words[8]. The experimental results are illustrated in Table 2.

For Fu et al. (2014), we test the performance using a linear projection model (denoted as S in Table 2) and piecewise projection models (P). It shows that the semantics of *is-a* relations are better modeled by multiple projection models, with a slight improvement in F-measure. By combining iterative projection models and pattern-based validation, the most recent approach (Wang and He, 2016) increases the F-measure by 4% and 2% in two datasets. In this method, the pattern-based statistics are calculated using the same corpus over which we train word embedding models. The main reason of the improvement may be that the projection models have a better generalization power by applying an iterative learning paradigm.

Mirza and Tonelli (2016) is implemented using three different strategies in combining the word vectors of a pair: i) concatenation $\mathbf{x}_i \oplus \mathbf{y}_i$ (de-

[7] Although the experiments in their paper are mostly related to temporal relations, the method can be applied to *is-a* relations without modification.

[8] Previously, these methods used different knowledge sources to train models and thus the results in their papers are not directly comparable with ours. To make fair comparison, we take the training data as the same knowledge source to train models for all methods.

Candidate Hypernym	P	T	Candidate Hypernym	P	T
Entity: 乙烯 (Ethylene)			**Entity**: 孙燕姿 (Stefanie Sun)		
化学品 (Chemical)	√	√	歌手 (Singer)	√	√
有机化学 (Organic Chemistry)	×	×	明星 (Star)	√	√
有机物 (Organics)	√	√	人物 (Person)	√	√
气体 (Gas)	√	√	金曲奖 (Golden Melody Award)	√	×
自然科学 (Natural Science)	×	×	音乐人 (Musician)	√	√
Entity: 显卡 (Graphics Card)			**Entity**: 核反应堆 (Nuclear Reactor)		
硬件 (Hardware)	√	√	建筑学 (Architecture)	×	×
电子产品 (Electronic Product)	√	√	核科学 (Nuclear Science)	×	×
电脑硬件 (Computer Hardware)	√	√	核能 (Nuclear Energy)	√	×
数码 (Digit)	×	×	自然科学 (Natural Science)	×	×

Table 3: Examples of model prediction. (P: prediction result, T: ground truth, √: positive, ×: negative)

TP/TN Rate	Rule P1	Rule N1	Rule N2
Dataset FD	98.6	92.3	94.1
Dataset BK	97.6	96.8	97.3

Table 4: TP/TN rates of three linguistic rules (%).

noted as C), ii) addition $\mathbf{x}_i + \mathbf{y}_i$ (A) and iii) subtraction $\mathbf{x}_i - \mathbf{y}_i$ (S). As seen, the classification models using addition and subtraction have similar performance in two datasets, while the concatenation strategy outperforms previous two approaches. Although Li et al. (2015) achieve a high performance in their dataset, this method does not perform well in ours. The most likely cause is that the features in that work are designed specifically for the Chinese Wikipedia category system. Our initial model has a higher accuracy than all the baselines. By utilizing the transductive learning framework, we boost the F-measure by 1.7% and 2.1%, respectively. Therefore, our method is effective to predict hypernyms of Chinese entities. We further conduct statistical tests which show our method significantly ($p < 0.01$) improves the F-measure over the state-of-the-art method (Wang and He, 2016).

4.4 Effectiveness of Linguistic Rules

To illustrate the effectiveness of linguistic rules, we present the true positive (or negative) rate by using one positive (or negative) rule solely, shown in Table 4. These values serve as γ_is in the transductive learning stage. The results indicate that these rules have high precision (over 90%) over both datasets for our task.

We state that currently we only use a few handcraft linguistic rules in our work. The proposed approach is a general framework that can encode arbitrary numbers of rules and in any language.

4.5 Error Analysis and Case Studies

We analyze correct and error cases in the experiments. Some examples of prediction results are shown in Table 3. We can see that our method is generally effective. However, some mistakes occur mostly because it is difficult to distinguish strict *is-a* and *topic-of* relations. For example, the entity Nuclear Reactor is semantically close to Nuclear Energy. The error statistics show that such kind of errors account for approximately 80.2% and 78.6% in two test sets, respectively.

Based on the literature study, we find that such problem has been also reported in (Fu et al., 2013; Wang and He, 2016). To reduce such errors, we employ the Chinese thematic lexicon based on Li et al. (2015) in the transductive learning stage but the coverage is still limited. Two possible solutions are: i) adding more negative training data of this kind; and ii) constructing a large-scale thematic lexicon automatically from the Web.

4.6 Experiments on English Datasets

To examine how our method can benefit hypernym prediction for the English language, we use two standard datasets in this paper. The first one is a benchmark dataset for distributional semantic evaluation, i.e., *BLESS* (Baroni and Lenci, 2011). Because the number of pairs in *BLESS* is relatively small, we also use the *Shwartz* (Shwartz et al., 2016) dataset. In the experiments, we treat the HYPER relations as positive data (1,337 pairs) and randomly sample 30% of the RANDOM relations as negative data (3,754 pairs) in *BLESS*. To create a relatively balanced dataset, we take the random split of *Shwartz* as input and use only 30% of the negative pairs. The dataset contains 14,135 positive pairs and 16,956 negative pairs. We use English Wikipedia as the text corpus to estimate the

Dataset	BLESS			Shwartz		
Method	P	R	F	P	R	F
Lenci and Benotto (2012)	42.8	38.6	40.6	38.5	50.1	43.5
Santus et al. (2014)	59.2	52.3	55.4	51.2	71.5	59.6
Fu et al. (2014) (S)	65.3	62.4	63.8	65.6	66.1	65.8
Fu et al. (2014) (P)	68.1	64.2	66.1	62.3	71.9	67.3
Mirza and Tonelli (2016) (C)	79.4	**84.1**	81.7	79.3	**80.9**	**80.1**
Mirza and Tonelli (2016) (A)	80.7	72.3	76.3	79.1	79.6	79.4
Mirza and Tonelli (2016) (S)	78.0	81.2	79.6	**80.5**	77.5	79.0
Wang and He (2016)	76.2	75.4	75.8	75.1	76.3	75.6
Ours (Initial)	79.3	76.3	77.7	77.2	76.8	77.0
Ours	**84.4**	79.5	**81.9**	79.1	77.5	78.3

Table 5: Performance comparison on test sets for English hypernym prediction (%).

statistics, and the pre-trained embedding vectors of English words[9].

For comparison, we test all the baselines over English datasets except Li et al. (2015). This is because most features in Li et al. (2015) can only be used in the Chinese environment. To implement Wang and He (2016) for English, we use the original Hearst patterns (Hearst, 1992) to perform relation selection and do not consider *not-is-a* patterns. We also take two recent DSM based approaches (Lenci and Benotto, 2012; Santus et al., 2014) as baselines. As for our own method, we do not use linguistic rules in Table 1 for English. The results are illustrated in Table 5. As seen, our method is superior to all the baselines over *BLESS*, with an F-measure of 81.9%. In *Shwartz*, while the approach (Mirza and Tonelli, 2016) has the highest F-measure of 80.1%, our method is generally comparable to theirs and outperforms others. The results suggest that although our method is not necessarily the state-of-the-art for English hypernym prediction, it has several potential applications. Refer to Section 4.7 for discussion.

4.7 Discussion

From the experiments, we can see that the proposed approach outperforms the state-of-the-art methods for Chinese hypernym prediction. Although the English language is not our focus, our approach still has relatively high performance. Additionally, our work has potential values for the following applications:

- **Domain-specific or Context-sparse Relation Extraction.** If the task is to predict relations between words when it is related to a specific domain or the contexts are sparse, even for English, traditional pattern-based methods are likely to fail. Our method can predict the existence of relations without explicit textual patterns and requires a relatively small amount of pairs as training data.

- **Under-resourced Language Learning.** Our method can be adapted for relation extraction in languages with flexible expressions, few knowledge resources and/or low-performance NLP tools. Our method does not require deep NLP parsing of sentences in a text corpus and thus the performance is not affected by parsing errors.

5 Conclusion

In summary, this paper introduces a transuctive learning approach for Chinese hypernym prediction. By modeling linear projection models, linguistic rules and non-linear mappings, our method is able to identify Chinese hypernyms with high accuracy. Experiments show that the performance of our method outperforms previous approaches. We also discuss the potential applications of our method besides Chinese hypernym prediction. In our work, the candidate Chinese hyponyms and hypernyms are extracted from user generated categories. In the future, we will study how to construct a taxonomy from texts in Chinese.

Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000904.

[9] http://nlp.stanford.edu/projects/glove/

References

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. pages 1—-10.

Rahul Bhagat, Patrick Pantel, and Eduard H. Hovy. 2007. LEDIR: an unsupervised algorithm for learning directionality of inference rules. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 161–170.

Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *27th Annual Meeting of the Association for Computational Linguistics*.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third International Conference on Web Search and Web Data Mining*. pages 101–110.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. 2006. *Transductive Inference and Semi-Supervised Learning*. MIT Press.

Oren Etzioni, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th international conference on World Wide Web*. pages 100–110.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 1199–1209.

Ruiji Fu, Bing Qin, and Ting Liu. 2013. Exploiting multiple sources for open-domain hypernym discovery. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 1224–1234.

Alexander Gammerman, Katy S. Azoury, and Vladimir Vapnik. 1998. Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. pages 148–155.

Gene H. Golub, Per Christian Hansen, and Dianne P. O'Leary. 1999. Tikhonov regularization and total least squares. *SIAM J. Matrix Analysis Applications* 21(1):185–194.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics*. pages 539–545.

Chu-Ren Huang, Ru-Yng Chang, and Hshiang-Pin Lee. 2004. Sinica BOW (bilingual ontological wordnet): Integration of bilingual wordnet and SUMO. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering* 16(4):359–389.

Zornitsa Kozareva and Eduard H. Hovy. 2010. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. pages 1482–1491.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*. pages 543–546.

Hai-Guang Li, Xindong Wu, Zhao Li, and Gong-Qing Wu. 2013. A relation extraction method of chinese named entities based on location and semantic features. *Appl. Intell.* 38(1):1–15.

Jinyang Li, Chengyu Wang, Xiaofeng He, Rong Zhang, and Ming Gao. 2015. User generated content oriented chinese taxonomy construction. In *Web Technologies and Applications - 17th Asia-Pacific Web Conference*. pages 623–634.

Hanxiao Liu and Yiming Yang. 2015. Bipartite edge prediction via transductive learning over product graphs. In *Proceedings of the 32nd International Conference on Machine Learning*. pages 1880–1888.

Weiming Lu, Renjie Lou, Hao Dai, Zhenyu Zhang, Shansong Yang, and Baogang Wei. 2015. Taxonomy induction from chinese encyclopedias by combinatorial optimization. In *Proceedings of the 4th CCF Conference on Natural Language Processing and Chinese Computing*. pages 299–312.

Anil Menon, Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. 1996. Characterization of a class of sigmoid functions with applications to neural networks. *Neural Networks* 9(5):819–835.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the Acm* 38(11):39–41.

Paramita Mirza and Sara Tonelli. 2016. On the contribution of word embeddings to temporal relation classification. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 2818–2828.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Simone Paolo Ponzetto and Michael Strube. 2007. Deriving a large-scale taxonomy from wikipedia. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*. pages 1440–1445.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. AFET: automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1369–1378.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Learning by Reading and Learning to Read, the 2009 AAAI Spring Symposium*. pages 88–93.

Erik F. Tjong Kim Sang. 2007. Extracting hypernym pairs from the web. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Erik F. Tjong Kim Sang and Katja Hofmann. 2009. Lexical patterns or dependency patterns: Which is better for hypernym extraction? In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. pages 174–182.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 38–42.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17, NIPS 2004*. pages 1297–1304.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*. pages 697–706.

Idan Szpektor, Eyal Shnarch, and Ido Dagan. 2007. Instance-based evaluation of entailment rule acquisition. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. page 456–463.

Chengyu Wang, Ming Gao, Xiaofeng He, and Rong Zhang. 2015. Challenges in chinese knowledge graph construction. In *Proceedings of the 31st IEEE International Conference on Data Engineering Workshops*. pages 59–61.

Chengyu Wang and Xiaofeng He. 2016. Chinese hypernym-hyponym extraction from user generated categories. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 1350–1361.

Shan Wang and Francis Bond. 2013. Cbuilding the chinese open wordnet (cow): Starting from core synsets. In *Proceedings of the 11th Workshop on Asian Language Resources: ALR-2013 a Workshop of The 6th International Joint Conference on Natural Language Processing*. pages 10–18.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. pages 481–492.

Renjie Xu, Zhiqiang Gao, Yingji Pan, Yuzhong Qu, and Zhisheng Huang. 2008. An integrated approach for automatic construction of bilingual chinese-english wordnet. In *The Semantic Web, Proceedings of the 3rd Asian Semantic Web Conference*. pages 302–314.

Ruochen Xu, Yiming Yang, Hanxiao Liu, and Andrew Hsi. 2016. Cross-lingual text classification via model translation with limited dictionaries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. pages 95–104.

A Constituent-Centric Neural Architecture for Reading Comprehension

Pengtao Xie[*†] **and Eric P. Xing**[†]
[*]Machine Learning Department, Carnegie Mellon University
[†]Petuum Inc.
pengtaox@cs.cmu.edu, eric.xing@petuum.com

Abstract

Reading comprehension (RC), aiming to understand natural texts and answer questions therein, is a challenging task. In this paper, we study the RC problem on the Stanford Question Answering Dataset (SQuAD). Observing from the training set that most correct answers are centered around constituents in the parse tree, we design a constituent-centric neural architecture where the generation of candidate answers and their representation learning are both based on constituents and guided by the parse tree. Under this architecture, the search space of candidate answers can be greatly reduced without sacrificing the coverage of correct answers and the syntactic, hierarchical and compositional structure among constituents can be well captured, which contributes to better representation learning of the candidate answers. On SQuAD, our method achieves the state of the art performance and the ablation study corroborates the effectiveness of individual modules.

1 Introduction

Reading comprehension (RC) aims to answer questions by understanding texts, which is a challenge task in natural language processing. Various RC tasks and datasets have been developed, including Machine Comprehension Test (Richardson et al., 2013) for multiple-choice question answering (QA) (Sachan et al., 2015; Wang and McAllester, 2015), Algebra (Hosseini et al., 2014) and Science (Clark and Etzioni, 2016) for passing standardized tests (Clark et al., 2016), CNN/Daily Mail (Hermann et al., 2015) and Children's Book Test (Hill et al., 2015) for cloze-style QA (Chen et al., 2016; Shen et al., 2016), WikiQA (Yang et al., 2015), Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) and Microsoft Machine Reading Comprehension (Nguyen et al., 2016) for open domain QA. In this paper, we are specifically interested in solving the SQuAD QA task (Figure 1 shows an example), in light of its following features: (1) large scale: 107,785 questions, 23,215 paragraphs; (2) non-synthetic: questions are generated by crowdworkers; (3) large search space of candidate answers.

> The most authoritative account at the time came from the medical faculty in Paris in a report to the king of France that blamed the heavens. This report became the first and most widely circulated of a series of plague tracts that sought to give advice to sufferers. That the plague was caused by bad air became the most widely accepted theory. Today, this is known as the Miasma theory.
>
> 1. Who was the medical report written for?
> the king of France
>
> 2. What is the newer, more widely accepted theory behind the spread of the plague?
> bad air
>
> 3. What is the bad air theory officially known as?
> Miasma theory

Figure 1: An example of the SQuAD QA task

We study two major problems: (1) how to generate candidate answers? Unlike in multiple-choice QA and cloze-style QA where a small amount of answer choices are given, an answer in SQuAD could be any span in the text, resulting in a large search space with size $O(n^2)$ (Rajpurkar et al., 2016), where n is the number of words in the sentence. This would incur a lot of noise, ambigu-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1405–1414
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1129

ity and uncertainty, making it highly difficult to pick up the correct answer. (2) how to effectively represent the candidate answers? First, long-range semantics spanning multiple sentences need to be captured. As noted in (Rajpurkar et al., 2016), the answering of many questions requires multiple-sentence reasoning. For instance, in Figure 1, the last two sentences in the passages are needed to answer the third question. Second, local syntactic structure needs to be incorporated into representation learning. The study by (Rajpurkar et al., 2016) shows that syntax plays an important role in SQuAD QA: there are a wide range of syntactic divergence between a question and the sentence containing the answer; the answering of 64.1% questions needs to deal with syntactic variation; experiments show that syntactic features are the major contributing factors to good performance.

To tackle the first problem, motivated by the observation in (Rajpurkar et al., 2016) that the correct answers picked up by human are not arbitrary spans, but rather centered around constituents in the parse tree, we generate candidate answers based upon constituents, which significantly reduces the search space. Different from (Rajpurkar et al., 2016) who only consider exact constituents, we adopt a constituent expansion mechanism which greatly improves the coverage of correct answers.

For the representation learning of candidate answers which are sequences of constituents, we first encode individual constituents using a *chain-of-trees LSTM* (CT-LSTM) and tree-guided attention mechanism, then feed these encodings into a chain LSTM (Hochreiter and Schmidhuber, 1997) to generate representations for the constituent sequences. The CT-LSTM seamlessly integrates intra-sentence tree LSTMs (Tai et al., 2015) which capture the local syntactic properties of constituents and an inter-sentence chain LSTM which glues together the sequence of tree LSTMs such that the semantics of each sentence can be propagated to others. The tree-guided attention leverages the hierarchical relations among constituents to learn question-aware representations.

Putting these pieces together, we design a constituent-centric neural network (CCNN), which contains four layers: a chain-of-trees LSTM encoding layer, a tree-guided attention layer and a candidate-answer generation layer, a prediction layer. Evaluation on SQuAD demonstrates the effectiveness of CCNN.

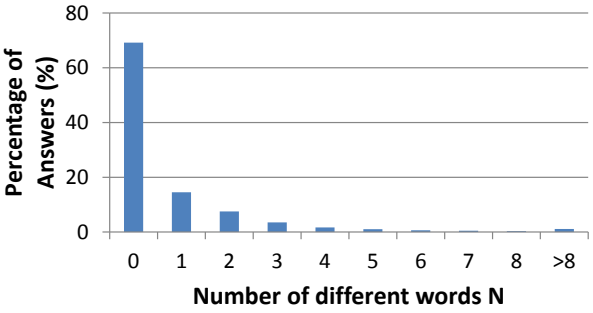


Figure 2: Percentage of answers that differ from their closest constituents by N words

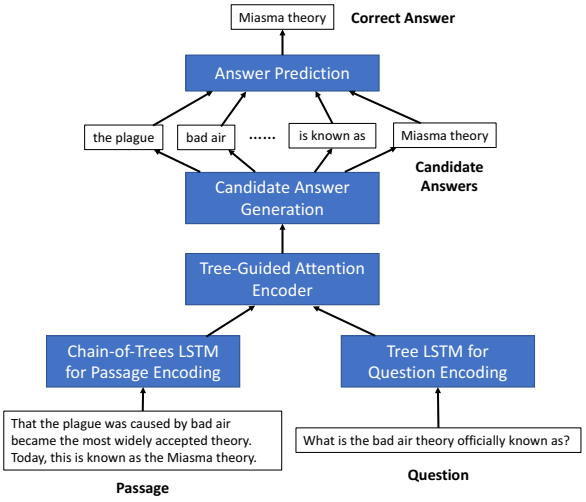


Figure 3: Constituent-centric neural network.

2 Constituent-Centric Neural Network for Reading Comprehension

2.1 Overall Architecture

As observed in (Rajpurkar et al., 2016), almost all correct answers are centered around the constituents. To formally confirm this, we compare the correct answers in the training set with constituents generated by the Stanford parser (Manning et al., 2014): for each correct answer, we find its "closest" constituent – the longest constituent that is a substring of the answer, and count how many words they differ from (let N denote this number). Figure 2 shows the percentage of answers whose N equals to $0, \cdots, 8$ and $N > 8$. As can be seen, $\sim$70% answers are exactly constituents ($N = 0$) and $\sim$97% answers differ from the closest constituents by less equal to 4 words. This observation motivates us to approach the

reading comprehension problem in a constituent-centric manner, where the generation of candidate answers and their representation learning are both based upon constituents.

Specifically, we design a Constituent-Centric Neural Network (CCNN) to perform end-to-end reading comprehension, where the inputs are the passage and question, and the output is a span in the passage that is mostly suitable to answer this question. As shown in Figure 3, the CCNN contains four layers. In the *encoding* layer, the chain-of-trees LSTM and tree LSTM encode the constituents in the passage and question respectively. The encodings are fed to the tree-guided *attention layer* to learn question-aware representations, which are passed to the *candidate-answer generation layer* to produce and encode the candidate answers based on constituent expansion. Finally, the *prediction layer* picks up the best answer from the candidates using a feed-forward network.

2.2 Encoding

Given the passages and questions, we first use the Stanford parser to parse them into constituent parse trees, then the encoding layer of CCNN learns representations for constituents in questions and passages, using tree LSTM (Tai et al., 2015) and chain-of-trees LSTM respectively. These LSTM encoders are able to capture the syntactic properties of constituents and long-range semantics across multiple sentences, which are crucial for SQuAD QA.

2.2.1 Tree LSTM for Question Encoding

Each question is a single sentence, having one constituent parse tree. Internal nodes in the tree represent constituents having more than one word and leaf nodes represent single-word constituent. Inspired by (Tai et al., 2015; Teng and Zhang, 2016), we build a bi-directional tree LSTM which consists of a bottom-up LSTM and a top-down LSTM, to encode these constituents (as shown in Figure 4). Each node (constituent) has two hidden states: $\mathbf{h}_\uparrow$ produced by the LSTM in bottom-up direction and $\mathbf{h}_\downarrow$ produced by the LSTM in top-down direction. Let T denote the maximum number of children an internal node could have. For each particular node, let L ($0 \leq L \leq T$) be the number of children it has, $\mathbf{h}_\uparrow^{(l)}$ and $\mathbf{c}_\uparrow^{(l)}$ be the bottom-up hidden state and memory cell of the l-th ($1 \leq l \leq L$) child (if any) respectively and $\mathbf{h}_\downarrow^{(p)}$

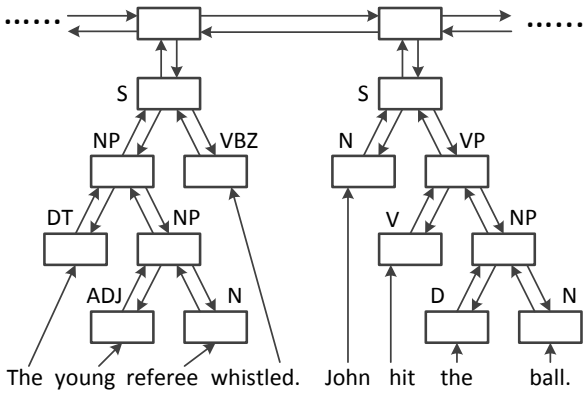


Figure 4: Chain-of-trees LSTM

and $\mathbf{c}_\downarrow^{(p)}$ be the top-down hidden state and memory cell of the parent.

In the bottom-up LSTM, each node has an input gate $\mathbf{i}_\uparrow$, L forget gates $\{\mathbf{f}_\uparrow^{(l)}\}_{l=1}^L$ corresponding to different children, an output gate $\mathbf{o}_\uparrow$ and a memory cell $\mathbf{c}_\uparrow$. For an internal node, the inputs are the hidden states and memory cells of its children and the transition equations are defined as:

$$
\begin{aligned}
&\mathbf{i}_\uparrow = \sigma(\textstyle\sum_{l=1}^{L} \mathbf{W}_\uparrow^{(i,l)} \mathbf{h}_\uparrow^{(l)} + \mathbf{b}_\uparrow^{(i)}) \\
&\forall l, \mathbf{f}_\uparrow^{(l)} = \sigma(\mathbf{W}_\uparrow^{(f,l)} \mathbf{h}_\uparrow^{(l)} + \mathbf{b}_\uparrow^{(f,l)}) \\
&\mathbf{o}_\uparrow = \sigma(\textstyle\sum_{l=1}^{L} \mathbf{W}_\uparrow^{(o,l)} \mathbf{h}_\uparrow^{(l)} + \mathbf{b}_\uparrow^{(o)}) \\
&\mathbf{u}_\uparrow = \tanh(\textstyle\sum_{l=1}^{L} \mathbf{W}_\uparrow^{(u,l)} \mathbf{h}_\uparrow^{(l)} + \mathbf{b}_\uparrow^{(u)}) \\
&\mathbf{c}_\uparrow = \mathbf{i}_\uparrow \odot \mathbf{u}_\uparrow + \textstyle\sum_{l=1}^{L} \mathbf{f}_\uparrow^{(l)} \odot \mathbf{c}_\uparrow^{(l)} \\
&\mathbf{h}_\uparrow = \mathbf{o}_\uparrow \odot \tanh(\mathbf{c}_\uparrow)
\end{aligned}
\tag{1}
$$

where the weight parameters $\mathbf{W}$ and bias parameters $\mathbf{b}$ with superscript l such as $\mathbf{W}_\uparrow^{(i,l)}$ are specific to the l-th child. For a leaf node which represents a single word, it has no forget gate and the input is the wording embedding (Pennington et al., 2014) of this word.

In the top-down direction, the gates, memory cell and hidden state are defined in a similar fashion as the bottom-up direction (Eq.(1)). For an internal node except the root, the inputs are the hidden state $\mathbf{h}_\downarrow^{(p)}$ and memory cell $\mathbf{c}_\downarrow^{(p)}$ of its parents. For a leaf node, in addition to $\mathbf{h}_\downarrow^{(p)}$ and $\mathbf{c}_\downarrow^{(p)}$, the inputs also contain the word embedding. For the root node, the top-down hidden state $\mathbf{h}_\downarrow^{(r)}$ is set to its bottom-up hidden state $\mathbf{h}_\uparrow^{(r)}$. $\mathbf{h}_\uparrow^{(r)}$ captures the semantics of all constituents, which is then replicated as $\mathbf{h}_\downarrow^{r}$ and propagated downwards to each individual constituent.

Concatenating the hidden states of two directions, we obtain the LSTM encoding for each node

$\mathbf{h} = [\mathbf{h}_\uparrow; \mathbf{h}_\downarrow]$ which will be the input of the attention layer. The bottom-up hidden state $\mathbf{h}_\uparrow$ composes the semantics of sub-constituents contained in this constituent and the top-down hidden state $\mathbf{h}_\downarrow$ captures the contextual semantics manifested in the entire sentence.

2.2.2 Chain-of-Trees LSTM for Passage Encoding

To encode the passage which contains multiple sentences, we design a chain-of-trees LSTM (Figure 4). A bi-directional tree LSTM is built for each sentence to capture the local syntactic structure and these tree LSTMs are glued together via a bi-directional chain LSTM (Graves et al., 2013) to capture long-range semantics spanning multiple sentences. The hidden states generated by the bottom-up tree LSTM serves as the input of the chain LSTM. Likewise, the chain LSTM states are fed to the top-down tree LSTM. This enables the encoding of every constituent to be propagated to all other constituents in the passage.

In the chain LSTM, each sentence t is treated as a unit. The input of this unit is generated by the tree LSTM of sentence t, which is the bottom-up hidden state $\mathbf{h}_{\uparrow t}$ at the root. Sentence t is associated with a forward hidden state $\overrightarrow{\mathbf{h}}_t$ and a backward state $\overleftarrow{\mathbf{h}}_t$. In the forward direction, the transition equations among the input gate $\overrightarrow{\mathbf{i}}_t$, forget gate $\overrightarrow{\mathbf{f}}_t$, output gate $\overrightarrow{\mathbf{o}}_t$ and memory cell $\overrightarrow{\mathbf{c}}_t$ are:

$$\begin{aligned}
\overrightarrow{\mathbf{i}}_t &= \sigma(\overrightarrow{\mathbf{W}}^{(i)}\mathbf{h}_{\uparrow t} + \overrightarrow{\mathbf{U}}^{(i)}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}^{(i)}) \\
\overrightarrow{\mathbf{f}}_t &= \sigma(\overrightarrow{\mathbf{W}}^{(f)}\mathbf{h}_{\uparrow t} + \overrightarrow{\mathbf{U}}^{(f)}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}^{(f)}) \\
\overrightarrow{\mathbf{o}}_t &= \sigma(\overrightarrow{\mathbf{W}}^{(o)}\mathbf{h}_{\uparrow t} + \overrightarrow{\mathbf{U}}^{(o)}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}^{(o)}) \\
\overrightarrow{\mathbf{u}}_t &= \tanh(\overrightarrow{\mathbf{W}}^{(u)}\mathbf{h}_{\uparrow t} + \overrightarrow{\mathbf{U}}^{(u)}\overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{b}}^{(u)}) \\
\overrightarrow{\mathbf{c}}_t &= \overrightarrow{\mathbf{i}}_t \odot \overrightarrow{\mathbf{u}}_t + \overrightarrow{\mathbf{f}}_t \odot \overrightarrow{\mathbf{c}}_{t-1} \\
\overrightarrow{\mathbf{h}}_t &= \overrightarrow{\mathbf{o}}_t \odot \tanh(\overrightarrow{\mathbf{c}}_t)
\end{aligned} \tag{2}$$

The backward LSTM is defined in a similar way. Subsequently, $\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$, which encapsulate the semantics of all sentences, are inputted to the root of the top-down tree LSTM and propagated to all the constituents in sentence t.

To sum up, the CT-LSTM encodes a passage in the following way: (1) the bottom-up tree LSTMs compute hidden states $\mathbf{h}_\uparrow$ for each sentence and feed $\mathbf{h}_\uparrow$ of the root node into the chain LSTM; (2) the chain LSTM computes forward and backward states and feed them into the root of the top-down tree LSTMs; (3) the top-down tree LSTMs compute hidden states $\mathbf{h}_\downarrow$. At each constituent C, the bottom-up state $\mathbf{h}_\uparrow$ captures the semantics of sub-constituents in C and the top-down state $\mathbf{h}_\downarrow$ captures the semantics of the entire passage.

2.3 Tree-Guided Attention Mechanism

We propose a tree-guided attention (TGA) mechanism to learn a question-aware representation for each constituent in the passage, which consists of three ingredients: (1) constituent-level attention score computation; (2) tree-guided local normalization; (3) tree-guided attentional summarization. Given a constituent $\mathbf{h}^{(p)}$ in the passage, for each constituent $\mathbf{h}^{(q)}$ in the question, an unnormalized attention weight score a is computed as $a = \mathbf{h}^{(p)} \cdot \mathbf{h}^{(q)}$ which measures the similarity between the two constituents. Then we perform a tree-guided local normalization of these scores. At each internal node in the parse tree, where the unnormalized attention scores of its L children are $\{a_l\}_{l=1}^L$, a local normalization is performed using a softmax operation $\widetilde{a}_l = \exp(a_l)/\sum_{m=1}^L \exp(a_m)$ which maps these scores into a probabilistic simplex. This normalization scheme stands in contrast with the global normalization adopted in word-based attention (Wang and Jiang, 2016; Wang et al., 2016), where a single softmax is globally applied to the attention scores of all the words in the question.

Given these locally normalized attention scores, we merge the LSTM encodings of constituents in the question into an attentional representation in a recursive and bottom-up way. At each internal node, let $\mathbf{h}$ be its LSTM encoding, a and $\{a_l\}_{l=1}^L$ be the normalized attention scores of this node and its L children, and $\{\mathbf{b}_l\}_{l=1}^L$ be the attentional representations (which we will define later) generated at the children, then the attentional representation $\mathbf{b}$ of this node is defined as:

$$\mathbf{b} = a(\mathbf{h} + \sum_{l=1}^{L} a_l \mathbf{b}_l) \tag{3}$$

which takes the weighted representation $\sum_{l=1}^L a_l \mathbf{b}_l$ contributed from its children, adds in its own encoding $\mathbf{h}$, then performs a re-weighting using the attention score a. The attentional representation $\mathbf{b}^{(r)}$ at the root node acts as the final summarization of constituents in the question. We concatenate it to the LSTM encoding $\mathbf{h}^{(p)}$ of the passage constituent and obtain a *concatenated representation* $\mathbf{z} = [\mathbf{h}^{(p)}; \mathbf{b}^{(r)}]$ which will be the input of the candidate answer generation layer.

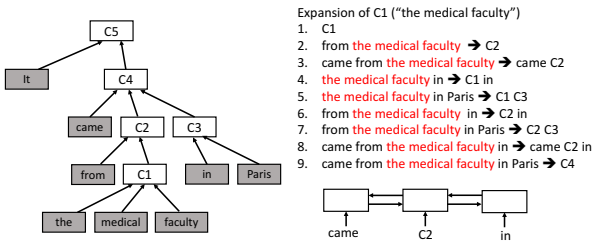


Figure 5: Constituent expansion. (Left) Parse tree of a sentence in the passage. (Top Right) Expansions of constituent C1 and their reductions (denoted by arrow). (Bottom Right) Learning the representation of an expansion using bidirectional chain-LSTM.

Unlike the word-based flat-structure attention mechanism (Wang and Jiang, 2016; Wang et al., 2016) where the attention scores are computed between words and normalized using a single global softmax, and the attentional summary is computed in a flat manner, the tree-guided attention calculates attention scores between constituents, normalizes them locally at each node in the parse tree and computes the attentional summary in a hierarchical way. Tailored to the parse tree, TGA is able to capture the syntactic, hierarchical and compositional structures among constituents and arguably generate better attentional representations, as we will validate in the experiments.

2.4 Candidate Answer Generation

As shown in Figure 2, while most correct answers in the training set are exactly constituents, some of them are not the case. To cover the non-constituent answers, we propose to expand each constituent by appending words adjacent to it. Let C denote a constituent and $S = \text{“}\cdots w_{i-1} w_i C w_j w_{j+1} \cdots\text{”}$ be the sentence containing C. We expand C by appending words preceding C (such as w_{i-1} and w_i) and words succeeding C (such as w_j and w_{j+1}) to C. We define an (l, r)-expansion of a constituent C as follows: append l words preceding C in the sentence to C; append r words succeeding C to C. Let M be the maximum expansion number that $l \leq M$ and $r \leq M$. Figure 5 shows an example. On the left is the constituent parse tree of the sentence “it came from the medical faculty in Paris”. On the upper right are the expansions of the constituent C1 – “the medical faculty”. To expand this constituent, we trace it back to the sentence and look up the M (M=2 in this case) words preceding C1 (which are “came” and “from”) and succeeding C1 (which are “in” and “Paris”). Then combinations of C1 and the preceding/succeeding words are taken to generate constituent expansions. On both the left and right side of C1, we have three choices of expansion: expanding 0,1,2 words. Taking combination of these cases, we obtain 9 expansions, including C1 itself ($(0, 0)$-expansion).

The next step is to perform reduction of constituent expansions. Two things need to be reduced. First, while expanding the current constituent, new constituents may come into being. For instance, in the expansion “came from C1 in Paris”, “in” and “Paris” form a constituent C3; “from” and C1 form a constituent C2; “came”, C2 and C3 form a constituent C4. Eventually, this expansion is reduced to C4. Second, the expansions generated from different constituents may have overlap and the duplicated expansions need to be removed. For example, the $(2, 1)$-expansion of C1 – “came from the medical faculty in” – can be reduced to “came C2 in”, which is the $(1, 1)$-expansion of C2. After reduction, each expansion is a sequence of constituents.

Next we encode these candidate answers and the encodings will be utilized in the prediction layer. In light of the fact that each expansion is a constituent sequence, we build a bi-directional chain LSTM (Figure 5, bottom right) to synthesize the representations of individual constituents therein. Let $E = C_1 \cdots C_n$ be an expansion consisting of n constituents. In the chain LSTM, the input of unit i is the combined representation of C_i. We concatenate the forward hidden state at C_n and backward state at C_1 as the final representation of E.

2.5 Answer Prediction and Parameter Learning

Given the representation of candidate answers, we use a feed-forward network $f : \mathbb{R}^d \to \mathbb{R}$ to predict the correct answer. The input of the network is the feature vector of a candidate answer and the output is a confidence score. The one with the largest score is chosen as the the correct answer.

For parameter learning, we normalize the confidence scores into a probabilistic simplex using softmax and define a cross entropy loss thereupon. Let J_k be the number of candidate answers produced from the k-th passage-question pair and

$\{\mathbf{z}_j^{(k)}\}_{j=1}^{J_k}$ be their representations. Let t_k be the index of the correct answer. Then the cross entropy loss of K pairs is defined as

$$\sum_{k=1}^{K}(-f(\mathbf{z}_{t_k}) + \log \sum_{j=1}^{J_k} \exp(f(\mathbf{z}_j^{(k)}))) \quad (4)$$

Model parameters are learned by minimizing this loss using stochastic gradient descent.

3 Experiments

3.1 Experimental Setup

The experiments are conducted on the Stanford Question Answering Dataset (SQuAD) v1.1, which contains 107,785 questions and 23,215 passages coming from 536 Wikipedia articles. The data was randomly partitioned into a training set (80%), a development set (10%) and an unreleased test set (10%). Rajpurkar et al. (2016) build a leaderboard to evaluate and publish results on the test set. Due to software copyright issues, we did not participate this online evaluation. Instead, we use the development set (which is untouched during model training) as test set. In training, if the correct answer is not in the candidate-answer set, we use the shortest candidate containing the correct answer as the target.

The Stanford parser is utilized to obtain the constituent parse trees for questions and passages. In the parse tree, any internal node which has one child is merged together with its child. For instance, in "(NP (NNS sufferers))", the parent "NP" has only one child "(NNS sufferers)", we merge them into "(NP sufferers)". We use 300-dimensional word embeddings from GloVe (Pennington et al., 2014) to initialize the model. Words not found in GloVe are initialized as zero vectors.

We use a feed-forward network with 2 hidden layers (both having the same amount of units) for answer prediction. The activation function is set to rectified linear. Hyperparameters in CCNN are tuned via 5-fold cross validation (CV) on the training set, summarized in Table 1. We use the ADAM (Kingma and Ba, 2014) optimizer to train the model with an initial learning rate 0.001 and a mini-batch size 100. An ensemble model is also trained, consisting of 10 training runs using the same hyperparameters. The performance is evaluated by two metrics (Rajpurkar et al., 2016): (1) exact match (EM) which measures the percentage of predictions that match any one of the ground truth answers exactly; (2) F1 score which measures the average overlap between the prediction and ground truth answer. In the development set each question has about three ground truth answers. F1 scores with the best matching answers are used to compute the average F1 score.

3.2 Results

Table 2 shows the performance of our model and previous approaches on the development set. CCNN (single model) achieves an EM score of 69.3% and an F1 score of 78.5%, significantly outperforming all previous approaches (single model). Through ensembling, the performance of CCNN is further improved and outperforms the baseline ensemble methods. The key difference between our method and previous approaches is that CCNN is constituent-centric where the generation and encoding of candidate answers are both based on constituents while the baseline approaches are mostly word-based where the candidate answer is an arbitrary span of words and the encoding is performed over individual words rather than at the constituent level. The constituent-centric model-design enjoys two major benefits. First, restricting the candidate answers from arbitrary spans to neighborhoods around the constituents greatly reduces the search space, which mitigates the ambiguity and uncertainty in picking up the correct answer. Second, the tree LSTMs and tree-guided attention mechanism encapsulate the syntactic, hierarchical and compositional structure among constituents, which leads to better representation learning of the candidate answers. We conjecture these are the primary reasons that CCNN outperforms the baselines and provide a validation in the next section.

3.3 Ablation Study

To further understand the individual modules in CCNN, we perform an ablation study. The results are shown in Table 2.

Tree LSTM To evaluate the effectiveness of tree LSTM in learning syntax-aware representations, we replace it with a syntax-agnostic chain LSTM. We build a bi-directional chain LSTM (denoted by A) over the entire passage to encode the individual words. Given a constituent $C = w_i \cdots w_j$, we build another bi-directional chain LSTM (denoted by B) over C where the inputs are the encodings of words $w_i, \cdots, w_j$ generated by LSTM

Parameter	Tuning Range	Best Choice
Maximum expansion number M in constituent expansion	$0, 1, 2, 3, 4, 5$	2
Size of hidden state in all LSTMs	$50, 100, 150, 200, 250, 300$	100
Size of hidden state in prediction network	100, 200, 300, 400, 500	400

Table 1: Hyperparameter Tuning

	Exact Match (EM,%)	F1 (%)
Single model		
Logistic Regression (Rajpurkar et al., 2016)	40.0	51.0
Fine Grained Gating (Yang et al., 2016)	60.0	71.3
Dynamic Chunk Reader (Yu et al., 2016)	62.5	71.2
Match-LSTM with Answer Pointer (Wang and Jiang, 2016)	64.1	73.9
Dynamic Coattentation Network (Xiong et al., 2016)	65.4	75.6
Multi-Perspective Context Matching (Wang et al., 2016)	66.1	75.8
Recurrent Span Representations (Lee et al., 2016)	66.4	74.9
Bi-Directional Attention Flow (Seo et al., 2016)	68.0	77.3
Ensemble		
Fine Grained Gating (Yang et al., 2016)	62.4	73.4
Match-LSTM with Answer Pointer (Wang and Jiang, 2016)	67.6	76.8
Recurrent Span Representations (Lee et al., 2016)	68.2	76.7
Multi-Perspective Context Matching (Wang et al., 2016)	69.4	78.6
Dynamic Coattentation Network (Xiong et al., 2016)	70.3	79.4
Bi-Directional Attention Flow (Seo et al., 2016)	73.3	81.1
CCNN Ablation (single model)		
Replacing tree LSTM with chain LSTM	63.5	73.9
Replacing chain-of-trees LSTM with independent tree LSTMs	64.8	75.2
Removing the attention layer	63.9	74.3
Replacing tree-guided attention with flat attention	65.6	75.9
CCNN (single model)	**69.3**	**78.5**
CCNN (ensemble)	**74.1**	**82.6**

Table 2: Results on the development set

A. In LSTM B, the forward hidden state of w_j and backward state of w_i are concatenated to represent C. Note that the attention mechanism remains intact, which is still guided by the parse tree. This replacement cause 5.8% and 4.6% drop of the EM and F1 scores respectively, which demonstrates the necessity of incorporating syntactic structure (via tree LSTM) into representation learning.

Chain-of-Trees LSTM (CT-LSTM) We evaluate the effectiveness of CT-LSTM by comparing it with a bag of tree LSTMs: instead of using a chain LSTM to glue the tree LSTMs, we treat them as independent. Keeping the other modules intact and replacing CT-LSTM with a bag of independent tree LSTMs, the EM and F1 score drop 4.5% and 3.3% respectively. The advantage of CT-LSTM is that it enables the semantics of one sentence to be propagated to others, which makes multiple-sentence reasoning possible.

Tree-Guided Attention (TGA) Mechanism To evaluate the effectiveness of TGA, we performed two studies. First, we take it off from the architecture. Then constituents in the passage are solely represented by the chain-of-trees LSTM encodings and the question sentence is represented by the tree LSTM encoding at the root of the parse tree. At test time, we concatenate the encodings of a candidate answer and the question as inputs of the prediction network. Removing the attention layer decreases the EM and F1 by 5.4% and 4.2% respectively, demonstrating the effectiveness of attention mechanism for question-aware representation learning.

Second, we compare the tree-structured mech-

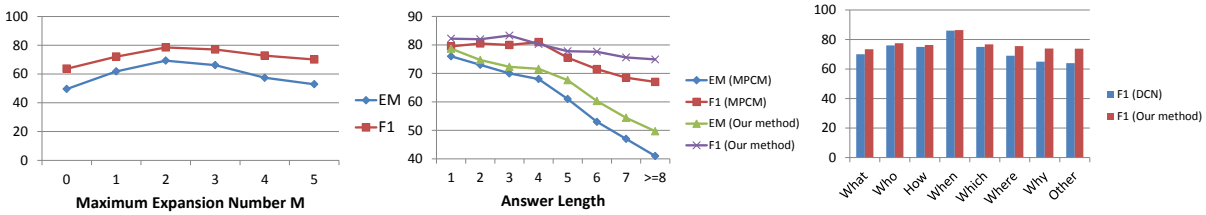


Figure 6: Performance for different (a) M (expansion number), (b) answer length, (c) question type.

anism in TGA with a flat-structure mechanism. For each constituent $\mathbf{h}_i^{(p)}$ in the passage, we compute its unnormalized score $a_{ij} = \mathbf{h}_i^{(p)} \cdot \mathbf{h}_j^{(q)}$ with every constituent $\mathbf{h}_j^{(q)}$ in the question (which has R constituents). Then a global softmax operation is applied to these scores, $\{\tilde{a}_{ij}\}_{j=1}^R = softmax(\{a_{ij}\}_{j=1}^R)$, to project them into a probabilistic simplex. Finally, a flat summarization $\sum_{j=1}^R \tilde{a}_{ij}\mathbf{h}_j^{(q)}$ is computed and appended to $\mathbf{h}_i^{(p)}$. Replacing TGA with flat-structure attention causes the EM and F1 to drop 3.7% and 2.6% respectively, which demonstrates the advantage of the tree-guided mechanism.

Constituent Expansion We study how the maximum expansion number M affects performance. If M is too small, many correct answers are not contained in the candidate set, which results in low recall. If M is too large, excessive candidates are generated, making it harder to pick up the correct one. Figure 6(a) shows how EM and F1 vary as M increases, from which we can see a value of M in the middle ground achieves the best tradeoff.

3.4 Analysis

In this section, we study how CCNN behaves across different answer length (number of words in the answer) and question types, which are shown in Figure 6(b) and (c). In Figure 6(b), we compare with the MPCM method (Wang et al., 2016). As answer length increases, the performance of both methods decreases. This is because for longer answers, it is more difficult to pinpoint the precise boundaries. The decreasing of F1 is slower than EM, because F1 is more elastic to small mismatches. Our method achieves larger improvement over MPCM at longer answers. We conjecture the reason is: longer answers have more complicated syntactic structure, which can be better captured by the tree LSTMs and tree-guided attention mechanism in our method. MPCM is built upon individual words and is syntax-agnostic.

In Figure 6(c), we compare with DCN (Xiong et al., 2016) on 8 question types. Our method achieves significant improvement over DCN on four types: "what", "where", "why" and "other". The answers of questions in these types are typically longer and have more complicated syntactic structure than the other four types where the answers are mostly entities (person, numeric, time, etc.). The syntax-aware nature of our method makes it outperform DCN whose model design does not explicitly consider syntactic structures.

4 Related Works

Several neural network based approaches have been proposed to solve the SQuAD QA problem, which we briefly review from three aspects: candidate answer generation, representation learning and attention mechanism.

Two ways were investigated for candidate answer generation: (1) chunking: candidates are preselected based on lexical and syntactic analysis, such as constituent parsing (Rajpurkar et al., 2016) and part-of-speech pattern (Yu et al., 2016); (2) directly predicting the start and end position of the answer span, using feed-forward neural network (Wang et al., 2016), LSTM (Seo et al., 2016), pointer network (Vinyals et al., 2015; Wang and Jiang, 2016), dynamic pointer decoder (Xiong et al., 2016).

The representation learning in previous approaches is conducted over individual words using the following encoders: LSTM in (Wang et al., 2016; Xiong et al., 2016); bi-directional gated recurrent unit (Chung et al., 2014) in (Yu et al., 2016); match-LSTM in (Wang and Jiang, 2016); bi-directional LSTM in (Seo et al., 2016).

In previous approaches, the attention (Bahdanau et al., 2014; Xu et al., 2015) mechanism is mostly word-based and flat-structured (Kadlec et al., 2016; Sordoni et al., 2016; Wang and Jiang,

2016; Wang et al., 2016; Yu et al., 2016): the attention scores are computed between individual words, are normalized globally and are used to summarize word-level encodings in a flat manner. Cui et al. (2016); Xiong et al. (2016) explored a coattention mechanism to learn question-to-passage and passage-to-question summaries. Seo et al. (2016) proposed to directly use the attention weights as augmented features instead of applying them for early summarization.

5 Conclusions and Future Work

To solve the SQuAD question answering problem, we design a constituent centric neural network (CCNN), where the generation and representation learning of candidate answers are both based on constituents. We use a constituent expansion mechanism to produce candidate answers, which can greatly reduce the search space without losing the recall of hitting the correct answer. To represent these candidate answers, we propose a chain-of-trees LSTM to encode constituents and a tree-guided attention mechanism to learn question-aware representations. Evaluations on the SQuAD dataset demonstrate the effectiveness of the constituent-centric neural architecture.

For future work, we will investigate the wider applicability of chain-of-trees LSTM as a general text encoder that can simultaneously capture local syntactic structure and long-range semantic dependency. It can be applied to named entity recognition, sentiment analysis, dialogue generation, to name a few. We will also apply the tree-guided attention mechanism to NLP tasks that need syntax-aware attention, such as machine translation, sentence summarization, textual entailment, etc. Another direction to explore is joint learning of syntactic parser and chain-of-trees LSTM. Currently, the two are separated, which may lead to suboptimal performance.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858* .

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Peter Clark and Oren Etzioni. 2016. My computer is an honor student-but how intelligent is it? standardized tests as a measure of ai. *AI Magazine* 37(1):5–12.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter D Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *AAAI*. pages 2580–2586.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423* .

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, pages 273–278.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

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

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*. pages 523–533.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* .

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*. volume 3, page 4.

Mrinmaya Sachan, Kumar Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *ACL (1)*. pages 239–249.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. *arXiv preprint arXiv:1609.05284* .

Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245* .

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Zhiyang Teng and Yue Zhang. 2016. Bidirectional tree-structured lstm with head lexicalization. *arXiv preprint arXiv:1611.06788* .

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

Hai Wang and Mohit Bansal Kevin Gimpel David McAllester. 2015. Machine comprehension with syntax, frames, and semantics .

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* .

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*. Citeseer, pages 2013–2018.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. 2016. Words or characters? fine-grained gating for reading comprehension. *arXiv preprint arXiv:1611.01724* .

Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. 2016. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996* .

Cross-lingual Distillation for Text Classification

Ruochen Xu
Carnegie Mellon Universit
ruochenx@cs.cmu.edu

Yiming Yang
Carnegie Mellon Universit
yiming@cs.cmu.edu

Abstract

Cross-lingual text classification(CLTC) is the task of classifying documents written in different languages into the same taxonomy of categories. This paper presents a novel approach to CLTC that builds on model distillation, which adapts and extends a framework originally proposed for model compression. Using soft probabilistic predictions for the documents in a label-rich language as the (induced) supervisory labels in a parallel corpus of documents, we train classifiers successfully for new languages in which labeled training data are not available. An adversarial feature adaptation technique is also applied during the model training to reduce distribution mismatch. We conducted experiments on two benchmark CLTC datasets, treating English as the source language and German, French, Japan and Chinese as the unlabeled target languages. The proposed approach had the advantageous or comparable performance of the other state-of-art methods.

1 Introduction

The availability of massive multilingual data on the Internet makes cross-lingual text classification (CLTC) increasingly important. The task is defined as to classify documents in different languages using the same taxonomy of predefined categories.

CLTC systems build on supervised machine learning require a sufficiently amount of labeled training data for every domain of interest in each language. But in reality, labeled data are not evenly distributed among languages and across domains. English, for example, is a label-rich language in the domains of news stories, Wikipedia pages and reviews of hotels, products, etc. But many other languages do not necessarily have such rich amounts of labeled data. This leads to an open challenge in CLTC, i.e., how can we effectively leverage the trained classifiers in a label-rich *source* language to help the classification of documents in other label-poor *target* languages?

Existing methods in CLTC use either a bilingual dictionary or a parallel corpus to bridge language barriers and to translate classification models (Xu et al., 2016) or text data(Zhou et al., 2016a). There are limitations and challenges in using either type of resources. Dictionary-based methods often ignore the dependency of word meaning and its context, and cannot leverage domain-specific disambiguation when the dictionary on hand is a general-purpose one. Parallel-corpus based methods, although more effective in deploying context (when combined with word embedding in particular), often have an issue of domain mismatch or distribution mismatch if the available source-language training data, the parallel corpus (human-aligned or machine-translation induced one) and the target documents of interest are not in exactly the same domain and genre(Duh et al., 2011). How to solve such domain/distribution mismatch problems is an open question for research.

This paper proposes a new parallel-corpus based approach, focusing on the reduction of domain/distribution matches in CLTC. We call this approach Cross-lingual Distillation with Feature Adaptation or CLDFA in short. It is inspired by the recent work in model compression (Hinton et al., 2015) where a large ensemble model is transformed to a compact (small) model. The assumption of knowledge distillation for model compression is that the knowledge learned by the large model can be viewed as a mapping from in-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1415–1425
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1130

put space to output (label) space. Then, by training with the soft labels predicted by the large model, the small model can capture most of the knowledge from the large model. Extending this key idea to CLTC, if we see parallel documents as different instantiations of the same semantic concepts in different languages, a target-language classifier should gain the knowledge from a well-trained source classifier by training with the target-language part of the parallel corpus and the soft labels made by the source classifier on the source language side. More specifically, we propose to distillate knowledge from the source language to the target language in the following 2-step process:

- Firstly, we train a source-language classifier with both labeled training documents and adapt it to the unlabeled documents from the source-language side of the parallel corpus. The adaptation enforces our classifier to extract features that are: 1) discriminative for the classification task and 2) invariant with regard to the distribution shift between training and parallel data.

- Secondly, we use the trained source-language classifier to obtain the *soft* labels for a parallel corpus, and the target-language part of the parallel corpus to train a target classifier, which yields a similar category distribution over target-language documents as that over source-language documents. We also use unlabeled testing documents in the target language to adapt the feature extractor in this training step.

Intuitively, the first step addresses the potential domain/distribution mismatch between the labeled data and the unlabeled data in the source language. The second step addresses the potential mismatch between the target-domain training data (in the parallel corpus) and the test data (not in the parallel corpus). The soft-label based training of target classifiers makes our approach unique among parallel-corpus based CLTC methods (Section 2.1. The feature adaptation step makes our framework particularly robust in addressing the distributional difference between in-domain documents and parallel corpus, which is important for the success of CLTC with low-resource languages.

The main contributions in this paper are the following:

- We propose a novel framework (CLDFA) for knowledge distillation in CLTC through a parallel corpus. It has the flexibility to be built on a large family of existing monolingual text classification methods and enables the use of a large amount of unlabeled data from both source and target language.

- CLDFA has the same computational complexity as the plug-in text classification method and hence is very efficient and scalable with the proper choice of plug-in text classifier.

- Our evaluation on benchmark datasets shows that our method had a better or at least comparable performance than that of other state-of-art CLTC methods.

2 Related Work

Related work can be outlined with respect to the representative work in CLTC and the recent progress in deep learning for knowledge distillation.

2.1 CLTC Methods

One branch of CLTC methods is to use lexical level mappings to transfer the knowledge from the source language to the target language. The work by Bel et al. (Bel et al., 2003) was the first effort to solve CLTC problem. They translated the target-language documents to source language using a bilingual dictionary. The classifier trained in the source language was then applied on those translated documents. Similarly, Mihalcea et al. (Mihalcea et al., 2007) built cross-lingual classifier by translating subjectivity words and phrases in the source language into the target language. Shi et al. (Shi et al., 2010) also utilized a bilingual dictionary. Instead of translating the documents, they tried to translate the classification model from source language to target language. Prettenhofer and Stein. (Prettenhofer and Stein, 2010) also used the bilingual dictionary as a word translation oracle and built their CLTC system on structural correspondence learning, a theory for domain adaptation. A more recent work by (Xu et al., 2016) extended seminal bilingual dictionaries with unlabeled corpora in low-resource languages. Chen et al. (Chen et al., 2016) used bilingual word embedding to map documents in source and target

language into the same semantic space, and adversarial training was applied to enforce the trained classifier to be language-invariant.

Some recent efforts in CLTC focus on the use of automatic machine translation (MT) technology. For example, Wan (Wan, 2009) used machine translation systems to give each document a source-language and a target-language version, where one version is machine-translated from the another one. A co-training (Blum and Mitchell, 1998) algorithm was applied on two versions of both source and target documents to iterative train classifiers in both languages. MT-based CLTC also include the work on multi-view learning with different algorithms, such as majority voting(Amini et al., 2009), matrix completion(Xiao and Guo, 2013) and multi-view co-regularization(Guo and Xiao, 2012a).

Another branch of CLTC methods focuses on representation learning or the mapping of the induced representations in cross-language settings (Guo and Xiao, 2012b; Zhou et al., 2016a, 2015, 2016b; Xiao and Guo, 2013; Jagarlamudi et al., 2011; De Smet et al., 2011; Vinokourov et al., 2002; Platt et al., 2010; Littman et al., 1998). For example, Meng et al. (Meng et al., 2012) and Lu et al. (Lu et al., 2011) used a parallel corpus to learn word alignment probabilities in a pre-processing step. Some other work attempts to find a language-invariant (or interlingua) representation for words or documents in different languages using various techniques, such as latent semantic indexing (Littman et al., 1998), kernel canonical correlation analysis (Vinokourov et al., 2002), matrix completion(Xiao and Guo, 2013), principal component analysis (Platt et al., 2010) and Bayesian graphical models (De Smet et al., 2011).

2.2 Knowledge Distillation

The idea of distilling knowledge in a neural network was proposed by Hinton et al (Hinton et al., 2015), in which they introduced a student-teacher paradigm. Once the cumbersome teacher network was trained, the student network was trained according to soften predictions of the teacher network. In the field of computer vision, it has been empirically verified that student network trained by distillation performs better than the one trained with hard labels. (Hinton et al., 2015; Romero et al., 2014; Ba and Caruana, 2014). Gupta et al.(Gupta et al., 2015) transfers supervision between images from different modalities(e.g. from RGB image to depth image). There are also some recent works applied distillation in the field of natural language. For example, Lili et al. (Mou et al., 2015) distilled task specific knowledge from a set of high-dimensional embeddings to a low-dimensional space. Zhiting et al. used an iterative distillation method to transfer the structured information of logic rules into the weights of a neural network. Kim et al. (Kim and Rush, 2016) applied knowledge distillation approaches in the field of machine translation to reduce the size of neural machine translation model. Our framework shares the same purpose of existing works that transfer knowledge between models of different properties, such as model complexity, modality, and structured logic. However, our transfer happens between models working on different languages. To the best of knowledge, this is the first work using knowledge distillation to bridge the language gap for NLP tasks.

3 Preliminary

3.1 Task and Notation

CLTC aims to use the training data in the source language to build a model applicable in the target language. In our setting, we have labeled data in source language $L_{src} = \{x_i, y_i\}_{i=1}^{L}$, where x_i is the labeled document in source language and y_i is the label vector. We then have our test data in the target language, given by $T_{tgt} = \{x'_i\}_{i=1}^{T}$. Our framework can also use unlabeled documents from both languages in transductive learning settings. We use $U_{src} = \{x_i\}_{i=1}^{M}$ to denote source-language unlabeled documents,$U_{tgt} = \{x'_i\}_{i=1}^{N}$ to denote target-language unlabeled documents, and $U_{parl} = \{(x_i, x'_i)\}_{i=1}^{P}$ to denote a unlabeled bilingual parallel corpus where x_i and x'_i are paired document translations of each other. We assume that the unlabeled parallel corpus does not overlap with the source-language training documents and the target-language test documents.

3.2 Convolutional Neural Network (CNN) as a Plug-in Classifier

We use a state-of-the-art CNN-based neural network classifier (Kim, 2014) as the plug-in classifier in our framework. Instead of using a bag-of-words representation for each document, the CNN model concatenates the word embeddings (vertical vectors) of each input document into a $n \times k$

matrix, where n is the length (number of word occurrences) of the document, and k is the dimension of word embedding. Denoting by

$$\mathbf{x_{1:n}} = \mathbf{x_1} \oplus \mathbf{x_2} \oplus ... \oplus \mathbf{x_n}$$

as the resulted matrix, with $\oplus$ the concatenation operator. One-dimensional convolutional filter $\mathbf{w} \in R^{hk}$ with window size h operates on every consecutive h words, with non-linear function f and bias b. For window of size h started at index i, the feature after convolutional filter is given by:

$$c_i = f(\mathbf{w} \cdot \mathbf{x_{i:i+h-1}} + b)$$

A max-over-time pooling (Collobert et al., 2011) is applied on c over all possible positions such that each filter extracts one feature. The model uses multiple filters with different window sizes. The concatenated outputs from filters consist the feature of each document. We can see the convolutional filters and pooling layers as feature extractor $\mathbf{f} = G_f(x, \theta_f)$, where θ_f contains parameters for embedding layer and convolutional layer. Theses features are then passed to a fully connected softmax layer to produce probability distributions over labels. We see the final fully connected softmax layer as a label classifier $G_y(\mathbf{f}, \theta_y)$ that takes the output $\mathbf{f}$ from the feature extractor. The final output of model is given by $G_y(G_f(x, \theta_f), \theta_y)$, which is jointly parameterized by $\{\theta_f, \theta_y\}$

We want to emphasize that our choice of the plug-in classifier here is mainly for its simplicity and scalability to demonstrate our framework. There is a large family of neural classifiers for monolingual text classification that could be used in our framework as well, including other convolutional neural networks by (Johnson and Zhang, 2014), the recurrent neural networks by (Lai et al., 2015; Zhang et al., 2016; Johnson and Zhang, 2016; Sutskever et al., 2014; Dai and Le, 2015), the attention mechanism by (Yang et al., 2016), the deep dense network by (Iyyer et al., 2015), and more.

4 Proposed Framework

Let us introduce two versions of our model for cross-language knowledge distillation, i.e., the vanilla version and the full version with feature adaptation. Both are supported by the proposed framework. We denote the former by CLD-KCNN and the latter by CLDFA-KCNN.

4.1 Vanilla Distillation

Without loss of generality, assume we are learning a multi-class classifier for the target language. We have $y \in 1, 2, ..., |v|$ where v is the set of all possible classes. We assume the base classification network produces real number logits q_j for each class. For example, for the case of CNN text classifier, the logits can be produced by a linear transformation which takes features extracted max-pooling layer and outputs a vector of size $|v|$. The logits are converted into probabilities of classes through the softmax layer, by normalizing each q_j with all other logits.

$$p_j = \frac{\exp(q_j/T)}{\sum_{k=1}^{|v|} \exp(q_k/T)} \quad (1)$$

where T is a temperature and is normally set to 1. Using a higher value of T produces a softer probability distribution over classes.

The first step of our framework is to train the source-language classifier on labeled source documents L_{src}. We use standard temperature $T = 1$ and cross-entropy loss as the objective to minimize. For each example and its label (x_i, y_i) from the source training set, we have:

$$\mathcal{L}(\theta_{src}) = - \sum_{(x_i,y_i) \in L_{src}} \sum_{k=1}^{|v|} \mathbb{1}\{y_i = k\} \log p(y = k|x_i; \theta_{src}) \quad (2)$$

where $p(y = k|x; \theta_{src})$ is source model controlled by parameter θ_{src} and $\mathbb{1}\{\cdot\}$ is the indicator function.

In the second step, the knowledge captured in θ_{src} is transferred to the distilled model in the target language by training it on the parallel corpus. The intuition is that paired documents in parallel corpus should have the same distribution of class predicted by the source model and target model. In the simplest version of our framework, for each source-language document in the parallel corpus, we predict a soft class distribution by source model with high temperature. Then we minimize the cross-entropy between soft distribution produced by source model and the soft distribution produced by target model on the paired documents in the target language. More formally, we optimize θ_{tgt} according to the following loss

function for each document pair (x_i, x'_i) in parallel corpus.

$$\mathcal{L}(\theta_{tgt}) = -\sum_{(x_i,x'_i)\in U_{parl}} \sum_{k=1}^{|v|} p(y=k|x_i;\theta_{src}) \log p(y=k|x'_i;\theta_{tgt}) \quad (3)$$

During distillation, the same high temperature is used for training target model. After it has been trained, we set the temperature to 1 for testing.

We can show that under some assumptions, the two-step cross-lingual distillation is equivalent to distilling a target-language classifier in the target-language input space.

Lemma 1. *Assume the parallel corpus* $\{x_i, x'_i\} \in U_{parl}$ *is generated by* $x'_i \sim p(X';\eta)$ *and* $x_i = t(x'_i)$*, where* η *controls the marginal distribution of* x_i *and* t *is a differentiable translation function with integrable derivative. Let* $f_{\theta_{src}}(t(x'))$ *be the function that outputs soft labels of* $p(y = k|t(x');\theta_{src})$*. The distillation given by equation 3 can be interpreted as distillation of a target language classifier* $f_{\theta_{src}}(t(x'))$ *on target language documents sampled from* $p(X';\eta)$*.*

$f_{\theta_{src}}(t(x'))$ is the classifier that takes input of target documents, translates them into source documents through t and makes prediction using the source classifier. If we further assume the testing documents have the same marginal distribution $P(X';\eta)$, then the distilled classifier should have similar generalization power as $f_{\theta_{src}}(t(x'))$.

Theorem 2. *Let source training data* $x_i \in L_{src}$ *has marginal distribution* $p(X;\lambda)$*. Under the assumptions of lemma 1, further assume* $p(t(x');\lambda) = p(x';\eta)$*,* $p(y|t(x')) = p(y|x')$ *and* $t'(x') \approx C$*, where* C *is a constant. Then* $f_{\theta_{src}}(t(x'))$ *actually minimizes the expected loss in target language data* $E_{x'\sim p(X;\eta),y\sim p(Y|x')}[L\big(y, f(t(x'))\big)]$.

Proof. By definition of equation 2, $f_{\theta_{src}}(x)$ minimizes the expected loss $E_{x\sim p(X;\lambda),y\sim p(Y|x)}[L\big(y, f(x)\big)]$, where L is cross-entropy loss in our case. Then we can write

$$\begin{aligned}
&E_{x\sim p(X;\lambda),y\sim p(Y|x)}[L\big(y, f(x)\big)] \\
=&\int p(x;\lambda) \sum_y p(y|x) L\big(y, f(x)\big) dx \\
=&\int p(t(x');\lambda) \sum_y p(y|t(x')) L\big(y, f(t(x'))\big) t'(x') dx' \\
\approx& C \int p(x';\eta) \sum_y p(y|x') L\big(y, f(t(x'))\big) dx' \\
=& C E_{x'\sim p(X;\eta),y\sim p(Y|x')}[L\big(y, f(t(x'))\big)]
\end{aligned}$$

□

4.2 Distillation with Adversarial Feature Adaptation

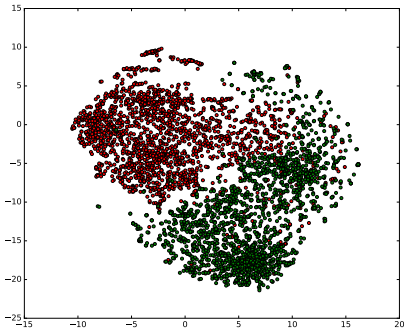


Figure 1: Extracted features for source-language documents in the English-Chinese Yelp Hotel Review dataset. Red dots represent features of the documents in L_{src} and green dots represent the features of documents in U_{parl}, which is a general-purpose parallel corpus.

Although vanilla distillation is intuitive and simple, it cannot handle distribution mismatch issues. For example, the marginal feature distributions of source-language documents in L_{src} and U_{parl} could be different, so are the distributions of target-language documents in U_{parl} and T_{tgt}. According to theorem 2, the vanilla distillation works for the best performance under unrealistic assumption: $p(t(x')|\lambda) = p(x'|\eta)$. To further illustrate our point, we trained a CNN classifier according to equation 2 and used the features extracted by G_f to present the source-language documents in both L_{src} and U_{parl}. Then we projected the high-dimensional features onto a 2-dimensional space via t-Distributed Stochastic Neighbor Embedding (t-SNE)(Maaten and Hinton, 2008). This resulted

the visualization of the project data in Figures 1 and 2.

It is quite obvious in Figure 1 that the general-purpose parallel corpus has a very different feature distribution from that of the labeled source training set. Even for machine-translated parallel data from the same domain, as shown in figure 2, there is still a non-negligible distribution shift from the source language to the target language for the extracted features. Our interpretation of this observation is that when the MT system (e.g. Google Translate) is a general-purpose one, it non-avoidably add translation ambiguities which would lead the distribution shift from the original domain. To address the distribution divergence brought by either a general-purpose parallel corpus or an imperfect MT system, we seek to adapt the features extraction part of our neural classifier such that the feature distributions on both sides should be close as possible in the newly induced feature space. We adapt the adversarial training method by (Ganin and Lempitsky, 2014) to the cross-lingual settings in our problems.

Given a set of training set of $L = \{x_i, y_i\}_{i=1,...,N}$ and an unlabeled set $U = \{x'_i\}_{i=1,...,M}$, our goal is to find a neural classifier $G_y(G_f(x, \theta_f), \theta_y)$, which has good discriminative performance on L and also extracts features which have similar distributions on L and U. One way to maximize the similarity of two distributions is to maximize the loss of a discriminative classifier whose job is to discriminate the two feature distributions. We denote this classifier by $G_d(\cdot, \theta_d)$, which is parameterized by θ_d.

At training time, we seek θ_f to minimize the loss of G_y and maximize the loss of G_d. Meanwhile, θ_y and θ_d are also optimized to minimize their corresponding loss. The overall optimization could be summarized as follows:

$$E(\theta_f, \theta_y, \theta_d) = \sum_{x_i, y_i \in L} L_y(y_i, G_y(G_f(x_i, \theta_f), \theta_y))$$
$$- \alpha \sum_{x_i \in L} L_d(0, G_d(G_f(x_i, \theta_f), \theta_d))$$
$$- \alpha \sum_{x_j \in U} L_d(1, G_d(G_f(x_j, \theta_f), \theta_d))$$

where L_y is the loss function for true labels y, L_d is loss function for binary labels indicating the source of data and α is the hyperparameter that controls the relative importance of two losses. We optimize θ_f, θ_y for minimizing E and optimize θ_d for maximizing E. We jointly optimize $\theta_f, \theta_y, \theta_d$ through the gradient reversal layer(Ganin and Lempitsky, 2014).

We use this feature adaptation technique to firstly adapt the source-language classifier to the source-language documents of the parallel corpus. When training the target-language classifier by matching soft labels on the parallel corpus, we also adapt the classifier to the target testing documents. We use cross-entropy loss functions as L_y and L_d for both feature adaptation.

5 Experiments and Discussions

5.1 Dataset

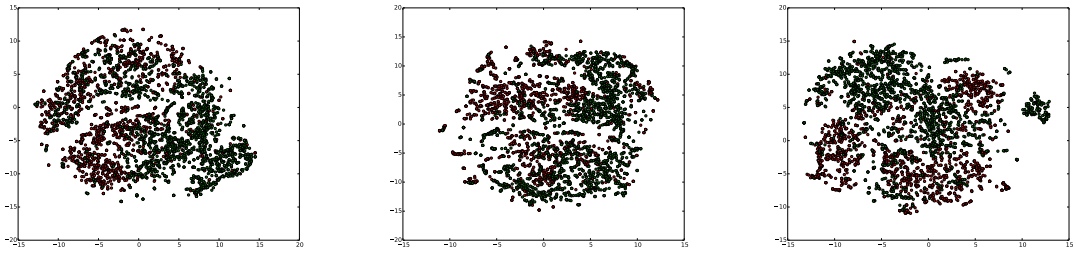
Our experiments used two benchmark datasets, as described below.

(1) Amazon Reviews

Language	Domain	# of Documents
English	book	50000
	DVD	30000
	music	25220
German	book	165470
	DVD	91516
	music	60392
French	book	32870
	DVD	9358
	music	15940
Japanese	book	169780
	DVD	68326
	music	55892

Table 1: Dataset Statistics for the Amazon reviews dataset

We used the multilingual multi-domain Amazon review dataset created by Prettenhofer and Stein (Prettenhofer and Stein, 2010). The dataset contains Amazon reviews in three domains: book, DVD and music. Each domain has the reviews in four different languages: English, German, French and Japanese. We treated English as the source language and the rest three as the target languages, respectively. This gives us 9 tasks (the product of the 3 domains and the 3 target languages) in total. For each task, there are 1000 positive and 1000 negative reviews in English and the target language, respectively. (Prettenhofer and Stein, 2010) also provides 2000 parallel reviews per task,



(a) Germany:DVD (b) French:Music (c) Japanese:Book

Figure 2: Extracted features for the source-language documents in the Amazon Reviews dataset. Red dots represent the features of the labeled training documents in L_{src}, and green dots represent the features of the documents in U_{parl}, which are the machine-translated documents from a target language. Below each figure is the target language and the domain of review (Section 5.1).

that were generated using Google Translate [1], and used by us for cross-language distillation. There are also several thousands of unlabeled reviews in each language. The statistics of unlabeled data is summarized in Table 1. All the reviews are tokenized using standard regular expressions except for Japanese, for which we used a publicly available segmenter [2].

(2) English-Chinese Yelp Hotel Reviews

This dataset was firstly used for CLTC by (Chen et al., 2016). The task is to make sentence-level sentiment classification with 5 labels(rating scale from 1 to 5), using English as the source language and Chinese as the target language. The labeled English data consists of balanced labels of 650k Yelp reviews from Zhang et al. (Zhang et al., 2015). The Chinese data includes 20k labeled Chinese hotel reviews and 1037k unlabeled ones from (Lin et al., 2015). Following the approach by (Chen et al., 2016), we use 10k of labeled Chinese data as validation set and another 10k hotel reviews as held-out test data. We a random sample of 500k parallel sentences from UM-courpus(Tian et al., 2014), which is a general-purpose corpus designed for machine translation.

5.2 Baselines

We compare the proposed method with other state-of-the-art methods as outlined below.

(1) Parallel-Corpus based CLTC Methods

Methods in this category all use an unlabeled parallel corpus. Methods named **PL-LSI** (Littman et al., 1998), **PL-OPCA** (Platt et al., 2010) and **PL-KCAA** (Vinokourov et al., 2002) learn latent document representations in a shared low-dimensional space by performing the Latent Semantic Indexing (LSI), the Oriented Principal Component Analysis (OPCA) and a kernel (namely KCAA) for the parallel text. **PL-MC** (Xiao and Guo, 2013) recovers missing features via matrix Completion, and also uses LSI to induce a latent space for parallel text. All these methods train a classifier in the shared feature space with labeled training data from both the source and target languages.

(2) MT-based CLTC Methods

The methods in this category all use an MT system to translate each test document in the target language to the source language in the testing phase. The prediction on each translated document is made by a source-language classifier, which can be a Logistic Regression model (**MT+LR**) (Chen et al., 2016) or a deep averaging network (**MT+DAN**) (Chen et al., 2016).

(3) Adversarial Deep Averaging Network

Similar to our approach, the adversarial Deep Averaging Network (**ADAN**) also exploits adversarial training for CLTC (Chen et al., 2016). However, it does not have the parallel-corpus based knowledge distillation part (which we do). Instead, it uses averaged bilingual embeddings of words as its input and adapts the feature extractor to produce similar features in both languages.

We also include the results of **mSDA** for the Yelp Hotel Reviews dataset. **mSDA** (Chen et al., 2012) is a domain adaptation method based on

[1] translate.google.com

[2] https://pypi.python.org/pypi/tinysegmenter

Target Language	Domain	PL-LSI	PL-KCCA	PL-OPCA	PL-MC	CLD-KCNN	CLDFA-KCNN
German	book	77.59	79.14	74.72	79.22	82.54	**83.95***
	DVD	79.22	76.73	74.59	81.34	82.24	**83.14***
	music	73.81	79.18	74.45	**79.39**	74.65	79.02
French	book	79.56	77.56	76.55	81.92	81.6	**83.37**
	DVD	77.82	78.19	70.54	81.97	82.41	**82.56**
	music	75.39	78.24	73.69	79.3	83.01	**83.31***
Janpanese	book	72.68	69.46	71.41	72.57	74.12	**77.36***
	DVD	72.55	74.79	71.84	76.6	79.67	**80.52***
	music	73.44	73.54	74.96	76.21	73.69	**76.46**
Averaged Accuracy		75.78	76.31	73.64	78.72	79.33	**81.08***

Table 2: Accuracy scores of methods on the Amazon Reviews dataset: the best score in each row (a task) is highlighted in bold face. If the score of CLDFA-KCNN is statistically significantly better (in one-sample proportion tests) than the best among the baseline methods, it is marked using a star.

Model	Accuracy
mSDA	31.44%
MT-LR	34.01%
MT-DAN	39.66%
ADAN	41.04%
CLD-KCNN	40.96%
CLDFA-KCNN	**41.82%**

Table 3: Accuracy scores of methods on the English-Chinese Yelp Hotel Reviews dataset

stacked denoising autoencoders, which has been proved to be effective in cross-domain sentiment classification evaluations. We show the results reported by (Chen et al., 2012), where they used bilingual word embedding as input for mSDA.

5.3 Implementation Detail

We pre-trained both the source and target classifier with unlabeled data in each language. We ran word2vec(Mikolov et al., 2013) [3] on the tokenized unlabeled corpus. The learned word embeddings are used to initialize the word embedding look-up matrix, which maps input words to word embeddings and concatenates them into input matrix.

We fine-tuned the source-language classifier on the English training data with 5-fold cross-validation. For English-Chinese Yelp-hotel review dataset, the temperature T(Section 4.1) in distillation is tuned on validation set in the target language. For Amazon review dataset, since there is no default validation set, we set temperature from low to high in $\{1, 3, 5, 10\}$ and take the average among all predictions.

[3]https://code.google.com/archive/p/word2vec/

5.4 Main Results

In tables 2 and 3 we compare the results of our methods (the vanilla version CLD-KCNN and the full version CLDFA-KCNN) with those of other methods based on the published results in the literature. The baseline methods are different in these two tables as they were previously evaluated (by their authors) on different benchmark datasets. Clearly, CLDFA-KCNN outperformed the other methods on all except one task in these two datasets, showing that knowledge distillation is successfully carried out in our approach. Noticing that CLDFA-KCNN outperformed CLD-KCNN, showing the effectiveness of adversarial feature extraction in reducing the distribution mismatch between the parallel corpus and the train/test data in the target domain. We should also point out that in Table 2, the four baseline methods (PL-LSI, PL-KCCA, PL-OPCA and PL-MC) were evaluated under the condition of using additional 100 labeled target documents for training, according to the author's report (Xiao and Guo, 2013). On the other hand, our methods (CLD-KCNN and CLDFA-KCNN) were evaluated under a tougher condition, i.e., not using any labeled data in the target domains.

We also test our framework when a few training documents in the target language are available. A simple way to utilize the target-language supervision is to fit the target-language model with labeled target data after optimizing with our cross-lingual distillation framework. The performance of CLD-KCNN and CLDFA-KCNN trained with different sizes of labeled target-language data is shown in figure 3. We also compare the performance of training the same classifier using only

the target-language labels(**Target Only** in figure 3). As we can see, our framework can efficiently utilize the extra supervision and improve the performance over the training using only the target-language labels. The margin is most significant when the size of the target-language label is relatively small.

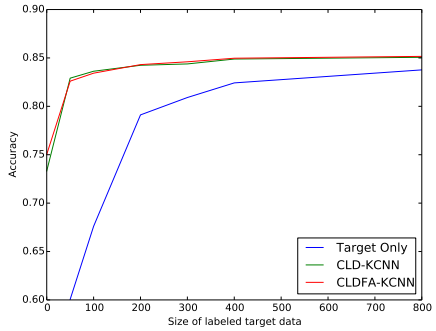


Figure 3: Accuracy scores of methods using varying sizes of target-language labeled data on the Amazon review dataset. The target language is German and the domain is music. The parallel corpus has a fixed size of 1000 and the size of the labeled target-language documents is shown on the x-axis

6 Conclusion

This work introduces a novel framework for distillation of discriminative knowledge across languages, providing effective and efficient algorithmic solutions for addressing domain/distribution mismatch issues in CLTC. The excellent performance of our approach is evident in our evaluation on two CLTC benchmark datasets, compared to that of other state-of-the-art methods.

Acknowledgement

We thank the reviewers for their helpful comments. This work is supported in part by Defense Advanced Research Projects Agency Information Innovation Oce (I2O), the Low Resource Languages for Emergent Incidents (LORELEI) Program, Issued by DARPA/I2O under Contract No. HR0011-15-C-0114, by the National Science Foundation (NSF) under grant IIS-1546329.

References

Massih Amini, Nicolas Usunier, and Cyril Goutte. 2009. Learning from multiple partially observed views-an application to multilingual text categorization. In *Advances in neural information processing systems*. pages 28–36.

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in neural information processing systems*. pages 2654–2662.

Nuria Bel, Cornelis HA Koster, and Marta Villegas. 2003. Cross-lingual text categorization. *Research and Advanced Technology for Digital Libraries* pages 126–139.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*. ACM, pages 92–100.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683* .

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Weinberger, and Claire Cardie. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614* .

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*. pages 3079–3087.

Wim De Smet, Jie Tang, and Marie-Francine Moens. 2011. Knowledge transfer across multilingual corpora via latent topics. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pages 549–560.

Kevin Duh, Akinori Fujino, and Masaaki Nagata. 2011. Is machine translation ripe for cross-lingual sentiment classification? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 429–433.

Yaroslav Ganin and Victor Lempitsky. 2014. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495* .

Yuhong Guo and Min Xiao. 2012a. Cross language text classification via subspace co-regularized multi-view learning. *arXiv preprint arXiv:1206.6481* .

Yuhong Guo and Min Xiao. 2012b. Transductive representation learning for cross-lingual text classification. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, pages 888–893.

Saurabh Gupta, Judy Hoffman, and Jitendra Malik. 2015. Cross modal distillation for supervision transfer. *arXiv preprint arXiv:1507.00448* .

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*.

Jagadeesh Jagarlamudi, Raghavendra Udupa, Hal Daumé III, and Abhijit Bhole. 2011. Improving bilingual projections via sparse covariance matrices. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 930–940.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058* .

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *Proceedings of The 33rd International Conference on Machine Learning*. pages 526–534.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947* .

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*. pages 2267–2273.

Yiou Lin, Hang Lei, Jia Wu, and Xiaoyu Li. 2015. An empirical study on sentiment classification of chinese review using word embedding. *arXiv preprint arXiv:1511.01665* .

Michael L Littman, Susan T Dumais, and Thomas K Landauer. 1998. Automatic cross-language information retrieval using latent semantic indexing. In *Cross-language information retrieval*, Springer, pages 51–62.

Bin Lu, Chenhao Tan, Claire Cardie, and Benjamin K Tsou. 2011. Joint bilingual sentiment classification with unlabeled parallel corpora. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 320–330.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Ge Xu, and Houfeng Wang. 2012. Cross-lingual mixture model for sentiment classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 572–581.

Rada Mihalcea, Carmen Banea, and Janyce M Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections .

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Lili Mou, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Distilling word embeddings: An encoding approach. *arXiv preprint arXiv:1506.04488* .

John C Platt, Kristina Toutanova, and Wen-tau Yih. 2010. Translingual document representations from discriminative projections. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 251–261.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1118–1127.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550* .

Lei Shi, Rada Mihalcea, and Mingjun Tian. 2010. Cross language text classification by model translation and semi-supervised learning. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1057–1067.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Liang Tian, Derek F Wong, Lidia S Chao, Paulo Quaresma, Francisco Oliveira, and Lu Yi. 2014. Um-corpus: A large english-chinese parallel corpus for statistical machine translation. In *LREC*. pages 1837–1842.

Alexei Vinokourov, John Shawe-Taylor, and Nello Cristianini. 2002. Inferring a semantic representation of text via cross-language correlation analysis. In *NIPS*. volume 1, page 4.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL*

and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1. Association for Computational Linguistics, pages 235–243.

Min Xiao and Yuhong Guo. 2013. A novel two-step method for cross language representation learning. In *Advances in Neural Information Processing Systems*. pages 1259–1267.

Ruochen Xu, Yiming Yang, Hanxiao Liu, and Andrew Hsi. 2016. Cross-lingual text classification via model translation with limited dictionaries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, pages 95–104.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* .

Rui Zhang, Honglak Lee, and Dragomir Radev. 2016. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361* .

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. pages 649–657.

Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. ACL.

Xinjie Zhou, Xianjun Wan, and Jianguo Xiao. 2016a. Cross-lingual sentiment classification with bilingual document representation learning .

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016b. Attention-based lstm network for cross-lingual sentiment classification .

Understanding and Predicting Empathic Behavior in Counseling Therapy

Verónica Pérez-Rosas[1], **Rada Mihalcea**[1], **Kenneth Resnicow**[2]
Satinder Singh[1] **and Lawrence An**[3]
[1]Computer Science and Engineering, [2]School of Public Health
[3]Center for Health Communications Research
University of Michigan
{vrncapr,mihalcea,kresnic,baveja,lcan}@umich.edu

Abstract

Counselor empathy is associated with better outcomes in psychology and behavioral counseling. In this paper, we explore several aspects pertaining to counseling interaction dynamics and their relation to counselor empathy during motivational interviewing encounters. Particularly, we analyze aspects such as participants' engagement, participants' verbal and nonverbal accommodation, as well as topics being discussed during the conversation, with the final goal of identifying linguistic and acoustic markers of counselor empathy. We also show how we can use these findings alongside other raw linguistic and acoustic features to build accurate counselor empathy classifiers with accuracies of up to 80%.

1 Introduction

Behavioral counseling is an important tool to address public health issues such as mental health, substance abuse, and nutrition problems among others. This has motivated increased interest in the study of mechanisms associated with successful interventions. Among them, counselor empathy has been identified as a key intervention component that relates to positive therapy outcomes.

Displaying empathic behavior helps counselors to build rapport with their clients. Empathy levels experienced during counseling have a significant effect on treatment outcomes, as clients who perceive their counselor as empathic are more likely to improve than the ones who do not (Moyers and Miller, 2013).

In this paper, we apply quantitative approaches to understand the dynamics of the counseling interactions and their relation to counselor empathy. We focus our analysis on counseling conducted using Motivational Interviewing (MI), a well-established evidence-based counseling style, where counselor empathy is defined as the active interest and effort to understand the client's perspective (Miller and Rollnick, 2013).

We address four main research questions. First, are there differences in how the counselor and the client engage during empathic conversations? We explore this question by conducting turn-by-turn word frequency analyses of participant's interactions across the counseling conversations. Second, are there differences in verbal and vocal mimicry patterns occurring during high and low empathy interactions? We address this question by measuring the degree of language matching, verbal and nonverbal coordination, and power dynamics expressed during the interaction. Third, are there content differences in counselor discourse during high and low empathy interactions? We answer this question by applying topic modeling to identify the topics that are more salient in high and low empathy interventions (or in both). Finally, fourth, can we build accurate classifiers of counselor empathy? We show how the linguistic and acoustic empathy markers identified in our analyses, together with other raw features, can be used to construct classifiers able to predict counselor empathy with accuracies of up to 80%.

2 Related Work

There have been several efforts to study the role of empathy during counseling interactions. (Xiao et al., 2012) applied a text-based approach to discriminate empathic from non-empathic encounters using word-frequency analysis. They conducted a set of experiments aiming to predict empathy at the utterance and session level on a manually annotated dataset. Results showed that empathy can

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1426–1435
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1131

be predicted at reasonable accuracy levels, comparable to human assessments. (Gibson et al., 2015) presented a more refined approach for this task, which in addition to n-grams included features derived from the Linguistic Inquire Word Count, LIWC (Tausczik and Pennebaker, 2010) as well as psycholinguistic norms.

Other research has focused on exploring aspects related to counselor empathy skills, such as their ability to match the client language. (Lord et al., 2015) analyzed the language coordination between client and counselor using Language Style Synchrony (LSS), a measure of the degree of similarity in word usage among speakers in adjacent talking turns. They found that empathy scores are positively related to LSS, and that higher levels of LSS are likely to result in higher empathy scores.

Another line of work has explored the use of the acoustic component to predict empathy levels during counseling encounters. (Xiao et al., 2014) presented a study on the automatic evaluation of counselor empathy based on the analysis of correlation between prosody patterns and the degree of empathy showed by the therapist during the counseling interactions. More recently, (Xiao et al., 2015) addressed the empathy prediction task by deriving language models from transcripts obtained by an automatic speech recognition system, thus eliminating the need of human intervention during speaker segmentation and transcription.

Most of this previous research has focused on the prediction task, and explored a variety of linguistic and acoustic representations for this goal. While some of this work has explored the linguistic accommodation between speakers, previous methods have not fully explored the conversational aspects of the counseling interaction.

In this paper, we seek to explore how conversational aspects such as engagement, accommodation, and discourse topics are related to counselor empathy by using strategies such as turn-by-turn word frequency analysis, language coordination, power dynamics analysis, and topic modeling. Furthermore, we build accurate empathy classifiers that rely on acoustic and linguistic cues inspired by our conversational analyses.

3 Counseling Empathy Dataset

The dataset used in this study consists of 276 MI audio-recorded sessions from: two clinical research studies on smoking cessation and medication adherence (Catley et al., 2012; Goggin et al., 2013); recordings of MI students from a graduate-level MI course; wellness coaching phone calls; brief medical encounters in dental practice and student counseling. The dataset was obtained from a previous study conducted by the authors. Further details can be found in (Pérez-Rosas et al., 2016).

The counseling sessions target three behavior changes: diet changes (72 sessions), smoking cessation (95 sessions), medication adherence (93 sessions). In addition, there are 16 sessions on miscellaneous topics. The full set comprises 97.8 hours of audio with an average session length of 20.8 minutes with a standard deviation of 11.5 minutes.

3.1 Data Preprocessing

Before conducting our analysis on the collected dataset, we performed several preprocessing steps to ensure the confidentiality of the data and to enable automatic text and audio feature extraction.

First, all the counseling recordings were subjected to an anonymization process. This includes manually trimming the audio to remove introductions, and inserting silences to replace references to participant's name and location.

Next, 162 sessions for which transcripts were not readily available were transcribed via Mechanical Turk (Marge et al., 2010) using the following guidelines: 1) transcribe speech turn by turn, 2) clearly identify the speaker (either client or counselor), 3) include speech disfluencies, such as false starts, repetitions of whole words or parts of words, and fillers. Transcriptions were manually verified at random points to avoid spam and ensure their quality.

Since sessions were recorded in natural conditions, we applied speech enhancement methods to remove noise and improve the speech signal quality. We started by converting the audio signal from a stereo to a mono channel and to a uniform sample rate of 16k. We then applied the Mean Square Error estimation of spectral amplitude for audio denoising, as implemented in the Voicebox Speech Processing toolbox (Brookes, 2003). To allow for a turn-by-turn audio analysis of the counseling interaction, we processed the speech signal to separate client and counselor speech segments. To accomplish this task, we used on automatic speech-to-text forced alignment API.[1] We

[1] YouTube Data API

then used the automatically-obtained time stamps to segment the audio and derive speaker-specific speech segments for each counseling dyad.

3.2 Data Annotation

Empathy assessments were obtained using the Motivational Interviewing Treatment Integrity (MITI) coding scheme version 4.1 (Moyers, 2014). Each session was assigned an empathy score using a 5-point Likert scale, which measures the extent to which the clinician understands or makes an effort to grasp the client's perspective and feelings. The coding was conducted by two independent teams of three coders who had previous experience in MI and MI coding. Annotations were conducted using the session audio recording along with its transcript. The inter-rater reliability, measured in a random sample of 20 double coded sessions using the Intra-Class Correlation Coefficient was 0.60,[2] suggesting that the annotators showed moderate agreement on empathy assessments. The reported annotation agreement was calculated on the original 5-scale empathy score and it is within the ranges reported in previous Motivational Interviewing studies (0.60-0.62). Because of the skewed frequency distribution of the empathy scores in the dataset, we decided to conduct our analyses using empathy as a binary outcome, by classifying scores from 1 to 3 as low empathy, and scores of 4 and 5 as high empathy. This resulted in 179 high empathy sessions and 97 low empathy sessions.

4 Empathic vs Non-Empathic Interactions: Counselor Engagement

We start by exploring differences in verbal exchange length between low and high empathy encounters as an indirect measure of participants engagement during the conversation. In this analysis, we account for the time dimension by segmenting the conversation into five equal portions. First, we look at the ratio of words exchanged between the counselor and the client for the different fractions of the conversation.[3] As shown in Figure 1, low empathy interactions present noticeably

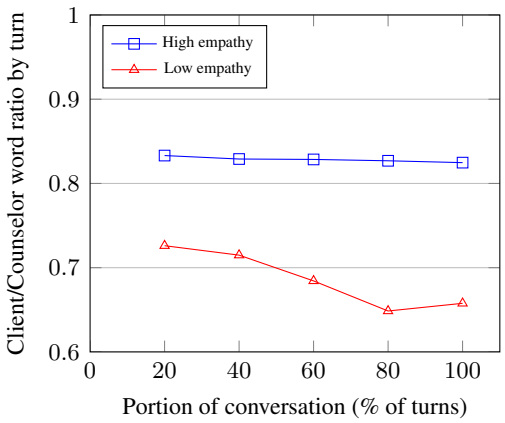


Figure 1: Word ratio by turn between clients and counselors as the conversation progresses.

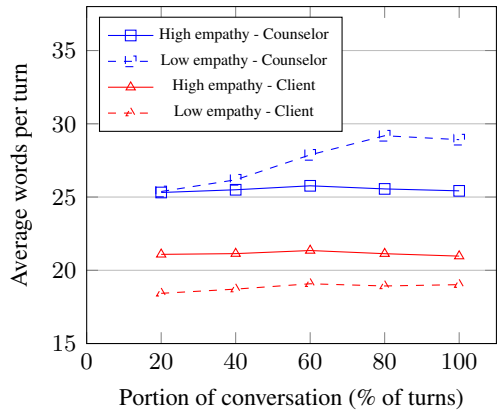


Figure 2: Average words per turn by counselors and clients as the conversation progresses.

lower ratio of words exchanged between counselors and clients across the interaction, while high empathy exchanges show consistently higher levels of interaction. This can be further observed in Figure 2, which shows that more empathic counselors speak considerably less than their clients, and that their less empathic counterparts. This is in line with findings in MI literature indicating that counselors who reduce the amount of time they talk with their clients are likely to allow more time for the patient to talk and explore their concerns, thus improving the perception of empathy and understanding.

5 The Role of Verbal and Nonverbal Accommodation in Empathy

Accommodation in health care communication involves counselor and client coordination including participation in communication and decision making, and shared understanding (D'Agostino

[2]ICC scores were obtained using a two-way mixed model with absolute agreement.

[3]This ratio is calculated for each pair of turns in the conversation, and it is simply measured as the number of words uttered by the counselor divided by the number of words uttered by the client. The turn-level word ratios are then averaged for all the turns included in a portion of the conversation.

and Bylund, 2014). We analyze the accommodation and its relation to empathy by exploring verbal and nonverbal behaviors exhibited by counseling participants during MI encounters. In addition to accommodation assessments, we explore the direction of the accommodation phenomena, i.e., whether the counselor is mirroring or leading the client.

5.1 Verbal Accommodation

In order to explore how verbal accommodation phenomena in our dataset relate to the MITI empathy assessments, we use two methods that are drawn from the Conversation Accommodation Theory. The first one is the Linguistic Style Matching (LSM) proposed in (Gonzales et al., 2009) to quantify to which extent one speaker, i.e., the counselor, matches the language of the other, i.e., the client. The second one is the Linguistic Style Coordination (LSC) metric proposed in (Danescu-Niculescu-Mizil et al., 2011), which quantifies the degree to which one individual immediately echoes the linguistic style of the person they are responding to. Both metrics are evaluated across eight linguistic markers from the LIWC dictionary (Tausczik and Pennebaker, 2010) (i.e., quantifiers, conjunctions, adverbs, auxiliary verbs, prepositions, articles, personal pronouns and impersonal pronouns).

LSM produces a score ranging between 0 and 1 indicating how much one person uses types of words comparable to the other person, while LSC generates a coordination score in the range of -1 to 1 indicating the degree of immediate coordination between speakers. While both measures are designed to analyze verbal synchrony, they can reveal different aspects of the counseling interaction. We use LSM to explore the potential match of language between counselors and clients across the counseling interaction, and we use LSC to quantify whether the counselor use of a specific linguistic marker in a given turn increases the probability of the client using the same marker during their reply. In addition, we use LSC to investigate power differences during the conversation based on the amount of coordination displayed by either counselor or client, under the assumption that the speaker who accommodates less holds the most power during the conversation (Danescu-Niculescu-Mizil et al., 2012).

Figure 3 shows the average LSM scores for

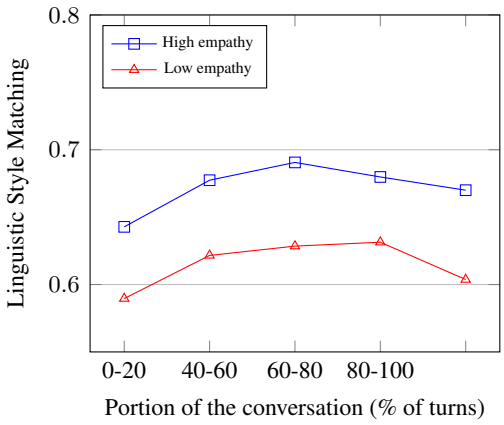


Figure 3: Linguistic style matching across five equal segments of the conversation duration.

eight linguistic markers measured on five equal segments of the conversation duration. As expected, we observe an increasing trend of language style matching during the counseling interaction in both high-empathic and low-empathic encounters, as people usually match their language unconsciously and regardless of the outcome of the conversation (Niederhoffer and Pennebaker, 2002). Interestingly, counselors and clients present a higher degree of language matching during high empathy encounters, while speakers in low empathy encounters show lower levels of style matching.

We evaluate the immediate LSC in two directions: coordination of counselors toward clients, and coordination of clients toward counselors. Results indicate low levels of immediate coordination in both cases, with values ranging between -0.06 and 0.1. Nonetheless, the results also suggest that clients coordinate more than counselors, with LSC(client,counselor)=-0.030 compared to LSC(counselor, client)=-0.038, which further suggests that counselors have more power (control) during the conversation.[4]

Analyses of the LSC levels from counselors to clients on different linguistic markers across high-empathic and low-empathic interactions provide interesting findings. While counselors generally show lower levels of coordination in the use of prepositions, auxiliary verbs, and personal pronouns (Figure 4), low-empathic counselors show higher LSC levels than their high-empathic counterparts. This can be attributed to the use of con-

[4]The differences in coordination showed during the analyses are statistically significant (two tailed t-test, p=0.0156)

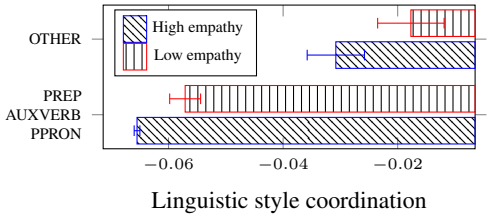


Figure 4: Linguistic style coordination from counselors to clients. OTHER include: quantifiers, conjunctions, adverbs, articles, and impersonal pronouns.

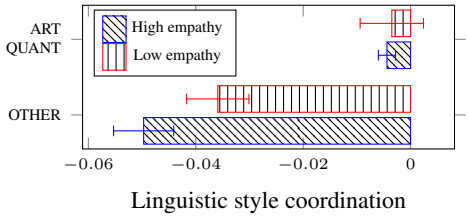


Figure 5: Linguistic style coordination from clients to counselors. OTHER include: conjunctions, adverbs, auxiliary verbs, prepositions, personal pronouns and impersonal pronouns.

frontational language (e.g., *I*, *could*, *should*, and *have*), which is often associated with low empathy. Similar analyses on the client side, shown in Figure 5, indicate significant differences in the use of linguistic markers by the client (except for articles and quantifiers). In particular, during low empathy encounters, clients coordinate more on the use of conjunctions, adverbs, auxiliary verbs, prepositions, personal pronouns, and impersonal pronouns.

5.2 Nonverbal Accommodation

Empathy is also shown through nonverbal channels such as visual and acoustics (Regenbogen et al., 2012). We explore the role of nonverbal mirroring in empathy by looking at vocal synchrony patterns shared between counselors and clients during the counseling interaction. We focus our analysis on vocal pitch, which is defined as the psychological perception of the voice frequency in terms of how high or how low it sounds. Pitch carries information about the speaker's emotional state, and has been shown to be related to the perception of empathy in psychotherapy (Reich et al., 2014).

We evaluate speech synchrony during turn-taking trajectories in the conversation. We consider two cases: sequences where the counselor replies to the client statements (e.g., rephrasing), and sequences where the counselor leads the interaction (e.g., asking questions). Starting with the turn-by-turn segmentation,[5] we extract pitch (F0) on each speaker-specific segment using OpenEar (Eyben et al., 2009).[6] We then measure the correlation of all pitch values during counselor following turns and during counselor leading turns across the entire therapy session.[7]

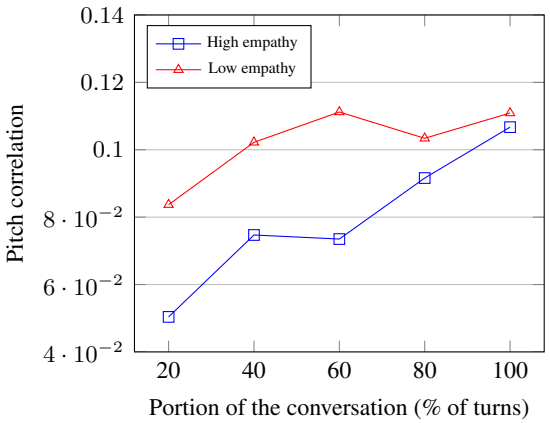


Figure 6: Pitch correlation among participants during counselor following turns as the conversation progresses.

Figures 6 and 7 show the trends in pitch synchrony across high-empathic and low-empathic encounters in the dataset. In the first figure, we observe that when replying to clients, counselors who are given low empathy scores show higher vocal synchrony levels than counselors who receive higher empathy scores. A potential explanation for this finding is that a counselor who mirrors the client pitch might amplify the emotional distress of the client, or may suggest the counselor's lack of confidence or knowledge (Reich et al., 2014).

On the other hand, we observe the opposite trend for the counselor leading sequences, where higher vocal synchrony levels are observed during high empathy interactions, which can be at-

[5]On average, there are approximately 40 counselor-client turns per conversation

[6]The feature extraction was done at audio-frame level every 10ms with a 25ms Hamming window.

[7]The terms of "counselor following" and "counselor leading" simply refer to how the correlation is computed. In "counselor following," we consider the set of counselor utterances and the previous client utterances; in "counselor leading," we consider the set of counselor utterances and the following client utterances.

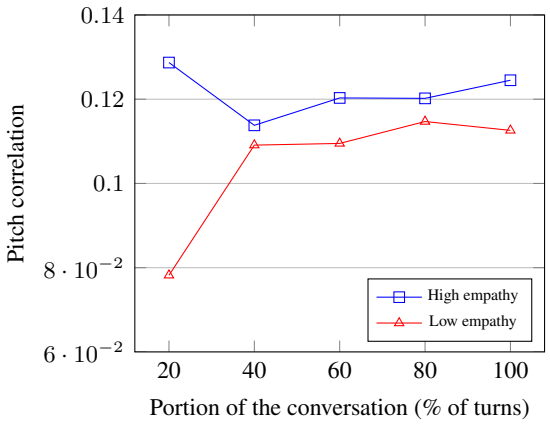


Figure 7: Pitch correlation among participants during counselor leading turns as the conversation progresses.

tributed to clients mirroring the counselor speech. The similarity is noticeably higher at the beginning of the conversation and gradually decreases as the conversation progresses. Moreover, the differences are not significant for the 40-100% turns, but results for the first 20% suggest significant differences at least in the beginning of the conversation ($p < 0.05$). This further confirm similarities during verbal and nonverbal accommodation, similar to how in section 5.1 we found that during high-empathic encounters, counselors hold control of the conversation and clients accommodate more than counselors.

6 Topics Discussed during Counseling Interaction and their Relation to Empathy

We also conduct content analysis on the counseling interactions, to identify themes discussed in high-empathic and low-empathic encounters. For this task, we employ the Meaning Extraction Method (MEM) (Chung and Pennebaker, 2008), a topic extraction method that identifies the most common words used in a set of documents, and clusters them into coherent themes by analyzing their co-occurrences. MEM has been used in the past in the psychotherapy domain to analyze salient topics in depression forums (Ramirez-Esparza et al., 2008) and also to investigate differences in topics discussed by patients given their therapy outcomes, i.e., therapeutic gain or unsuccessful therapy (Wolf et al., 2010).

Our analyses are conducted on counselor turns only, thus all the client turns are removed from each session transcript. We use the Meaning Extraction Helper tool (Boyd, 2016) to conduct the text preprocessing tasks, which include tokenizing and lemmatizing the words in each session, as well as removing function words. We keep only words who appear in at least 10% of the transcripts with a minimum frequency of 50. From the resulting list, we remove adjectives, adverbs, and verbs and keep only nouns as they usually refer to one definite class thus helping us to identify less ambiguous topics. Using the resulting noun list, with 339 entries, we generate a binary vector for each document, indicating the presence or absence of each noun in the document. We then run a Principal Component Analysis (PCA), followed by varimax rotation on the document matrix to find clusters of co-occurring nouns.

Behavior target	Sample words
Medication adherence	Adherence, dose, window, target, adherent, maintain, track
Smoking cessation	Cigarette, nicotine, risk, addiction, smoke, withdrawal
Weight management	diet, weight, eat, food, meal, lose, gain, cook, exercise

Table 1: Three behavior change targets in the dataset

The initial PCA shows that the first three components consist mainly of domain specific nouns. Notably, this accurately captures the presence of the three main behavior change targets discussed in the dataset, i.e., medication adherence, smoking cessation, and weight management; sample words from each component are shown in Table 1.

In order to identify topics potentially related to the counseling skill, we decided to remove the domain words from the analysis, which resulted in 250 nouns. Next, we use the same PCA configuration on the binary document matrix and rerun the experiment, which this time leads to 98 components. Following PCA literature recommendations (Velicer and Fava, 1998), we retain only components with at least three variables with loadings greater than 0.30, which leads to 14 components. We then re-run PCA forcing a 14 components solution; these components explain 35% of the total variance in the original matrix. Finally, we use the method proposed in (Wilson et al., 2016) to measure the degree to which a particular MEM topic (component) is used during high-empathic and low-emphatic encounters.

Topic	Sample nouns	Score
Concerns	Concern, scare, overwhelm, diagnose	2.52
Importance	Importance, reason, maintain, sense, increase	1.41
Inform	Information, schedule, discuss, read	1.14
Reflections	sound, start, look, mention, past, notice	1.19
Change	Health, past, experience, decision, realize, difficult, impact	1.27
Goals	Reach, choose, period, stick, idea, study, record	1.57
Motivation	Plan, motivate, routine, motivation, group, progress, fun	1.10
Support	Family, care, worry, job, lifestyle, job, focus, issue	0.92
Feelings	Worry, deal, stuck, struggle, leave	0.91
Guide	Stop, reduce, attempt, spend, replacement	0.79
Resistance	Trouble, barrier, fear, reach, involve, cover	073
Persuade	Routine, track, strategy, recommend, affect	0.64
Persuade	Stop, increase, decrease, benefit, consequences	0.455
Plan	Activity, strategy, barrier, couple	0.292

Table 2: Topics extracted by the MEM from MI sessions, along with sample nouns and salient topic scores.

Table 2 shows the scores assigned to each topic. In this table, scores greater than 1 correspond to topics salient in high empathy encounters while scores lower than 1 indicate topics salient in low empathy encounters. From this table, we can derive interesting observations. First, during high-empathic encounters, counselors seem to pay more attention to patient concerns, provide information, use reflective language, and talk about change. Second, during less empathic encounters, counselors seem to persuade and direct more, as well as focus on client's resistance to change. Interestingly, topics that are identified as dominant in less empathic interactions are also related to MI non-adherent behavior, which means the counselors are not following the MI strategy (Rollnick et al., 2008). Finally, regardless of the empathy shown during the encounter, counselors discuss patients' support system and feelings at similar rates (values closer to 1), which is expected when following the MI strategy.

7 Prediction of Counselor Empathy

In the previous sections, we provided evidence of important differences in linguistic and verbal behaviors exhibited by counselors and clients during high-empathic and low-empathic MI encounters. In this section, we explore the use of linguistic and acoustic cues to build a computational model that predicts counselor empathy during MI encounters.

The feature set consists of the cues identified during our exploratory analyses as potential indicators of counselor empathy, as well as additional text and audio features used during standard NLP and acoustic feature extraction.

The text-based features are extracted from the manual transcripts of the sessions, while the audio-based features are extracted from audio segments obtained by force-aligning each session transcript with its corresponding audio. However, as future work, we are considering to automatize this process by conducting automatic speaker diarization and transcription via automatic speech recognition.

During our experiments, we first explore the predictive power of each cue separately, followed by an integrated model that attempts to combine the linguistic and acoustic cues to improve the prediction of counselor empathy.

All the experiments are performed using a Random Forest (Breiman, 2001) classifier. Given the large number of features, we use feature selection based on information gain to identify the best set of features during each experiment. During this process we keep at least 20% of the features in each set. Evaluations are conducted using leave-one-session-out cross-validation. The feature selection algorithm is run on each training fold before the model is trained, and the final model includes the best subset of attributes. As a reference value, we use a majority class baseline, obtained by selecting high empathy as the default class, which corresponds to 64% accuracy.

7.1 Linguistic and Acoustic Features

Engagement: These features represent the participant's engagement during the conversation as described in Section 4. They are evaluated at 20% increments of the conversation duration and also at conversation (session) level. The features are listed in Table 3.

Linguistic accommodation: We measure the LSM and LSC metrics as described in section 5.2 over 74 LIWC categories and measured at 20% increments of the encounter duration.

[8]Calculated using the LSC metric

Feature	C	T
Counselor talk time based on syllable counting	✓	
Length of conversation setter, length of setter response, ratio between setter and response	✓	
Counselor turns, client turns	✓	✓
Average words during client and counselor turns	✓	✓
Ratio of counselor and client words in each turn	✓	✓
Rate of verbal mirroring on each LIWC category [8]	✓	

Table 3: Engagement features extracted at a) (C) conversation level, and b) (T) 20% increments of the conversation duration, in percentage of turns.

Nonverbal accommodation: This set includes the counselor-leading and counselor-following synchrony scores, calculated as described in section 5.2, and evaluated at 20% increments of the encounter duration.

Discourse topics: These features consist of the 14 topics identified in section 6 as frequently discussed during the MI encounters. The features are obtained by calculating the product of the principal components matrix and the binary document-term matrix.

Raw linguistic features: We extract a large set of linguistic features derived from the session transcript to model the counselor language. We include: unigrams and bigrams (ngrams), represented as a vector containing their frequencies in the session; psycholinguistic-inspired features that capture differences in semantic meaning (lexicons), represented as the total frequency counts of all the words in a lexicon-category that are present in the transcript; syntactic features that encode syntax patterns in the counselor statements (CFG), represented as a vector containing the frequency of lexicalized and unlexicalized production rules from the Context Free Grammar parse trees[9] of each transcript. The final linguistic features set consists of 13,648 features.

Raw acoustic features: This feature set includes a large number of speech features extracted with the OpenEar toolkit (Eyben et al., 2009). We use a predefined feature set, EmoLarge, which consists of a set of 6,552 features used for emotion recognition tasks. The features are derived from 25 low-level speech descriptors including intensity, loudness, 12 Mel frequency coefficients, pitch (F0), probability of voicing, F0 envelope, zero-crossing rate, and 8 line spectral frequencies.

[9]Extracted with the Stanford parser (Klein and Manning, 2003).

Feature set	Empathy		
		F-score	
	Acc.	HE	LE
Linguistic			
Engagement	71.01%	0.80	0.40
Ling Accom	73.19%	0.82	0.44
Topics	75.72%	0.83	0.57
N-grams	78.62%	0.86	0.58
Lexicons	76.09%	0.84	0.55
CFG	76.09%	0.84	0.53
All linguistic	**80.07%**	**0.87**	0.62
Acoustic			
Nonverb Accom	64.86%	0.79	0.00
Raw acoustic	73.91%	0.82	0.53
All acoustic	75.72%	0.83	0.56
Linguistic+ Acoustic			
Ling+acoustic (early)	76.81%	0.84	0.56
Ling+acoustic (late)	79.35%	0.86	**0.71**

Table 4: Overall prediction results and F-scores for high empathy (HE) and low empathy (LE) using linguistic and acoustic feature sets.

7.2 Classification Results

Classification results for each feature set are shown in Table 4. For the linguistic and acoustic modalities, almost all the feature sets provide classification accuracies above the baseline, with good F-scores for both high and low empathy. The only exception are the nonverbal accommodation features, which have an accuracy comparable to the baseline (64.86% vs. 64%).

When combining all the feature sets for each modality, we observe performance gains in the range of 10 to 15%, as compared to the models that use one feature set at a time.

We also conduct multimodal experiments where we combine linguistic and acoustic features using either early fusion by concatenating all the feature vectors, or late fusion by aggregating the outputs of each classifier using a rule-based score level fusion that assigns a weight of 0.8 to the linguistic classifier, and 0.2 to the acoustic classifier.[10]

Overall, the results show performance gains when using late fusion as compared to early fusion. While the late fusion model does not outperform the best linguistic model in terms of accuracy and high empathy F-score, the multimodal late fusion classifier has significantly better F-score performance in the classification of low empathy encounters, thus suggesting potential benefits of fus-

[10]Weights empirically determined on a development set by evaluating increments of 0.2 for each classifier weight.

ing acoustic and linguistic cues during the prediction of counselor empathy.

8 Conclusions

In this paper, we presented an extensive analysis of counselors and clients behaviors during MI encounters, and found significant differences in the way counselors and clients behave during high and low empathy encounters. We specifically explored the engagement, coordination, and discourse of counselors during MI interventions. Our main findings include:

Engagement: Empathic counselors show more engagement during the conversation by a) showing levels of verbal interaction consistent with their client, and b) reducing their relative talking time with clients.

Coordination: Empathic counselors match the linguistic style of their clients across the session, but maintain control of the conversation by coordinating less at immediate conversation turn-level.

Conversation content: Empathic counselors use reflective language and talk about behavior change, while less empathic counselors persuade more and focus on client resistance toward change.

The results of these analyses were used to build accurate counselor empathy classifiers that rely on linguistic and acoustic cues, with accuracies of up to 80%.

In the future, we plan to build upon the acquired knowledge and the developed classifiers to create automatic tools that provide accurate evaluative feedback of counseling practice.

Acknowledgments

We are grateful to Prof. Berrin Yanikoglu for her very useful input on the machine learning framework. This material is based in part upon work supported by the University of Michigan under the M-Cube program, by the National Science Foundation (grant #1344257), the John Templeton Foundation (grant #48503), and the Michigan Institute for Data Science. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the University of Michigan, the National Science Foundation, the John Templeton Foundation, or the Michigan Institute for Data Science.

References

Ryan. L. Boyd. 2016. Meh: Meaning extraction helper (version 1.4.14) [software].

Leo Breiman. 2001. Random forests. *Machine learning* 45(1):5–32.

Michael Brookes. 2003. Voicebox: Speech processing toolbox for matlab.

Delwyn Catley, Kari J Harris, Kathy Goggin, Kimber Richter, Karen Williams, Christi Patten, Ken Resnicow, Edward Ellerbeck, Andrea Bradley-Ewing, Domonique Malomo, et al. 2012. Motivational interviewing for encouraging quit attempts among unmotivated smokers: study protocol of a randomized, controlled, efficacy trial. *BMC public health* 12(1):456.

Cindy K Chung and James W Pennebaker. 2008. Revealing dimensions of thinking in open-ended self-descriptions: An automated meaning extraction method for natural language. *Journal of Research in Personality* 42(1):96–132.

Thomas A D'Agostino and Carma L Bylund. 2014. Nonverbal accommodation in health care communication. *Health communication* 29(6):563–573.

Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words!: linguistic style accommodation in social media. In *Proceedings of the 20th international conference on World Wide Web*. ACM, pages 745–754.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of WWW*. pages 699–708.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2009. Openear introducing the munich open-source emotion and affect recognition toolkit. In *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*. IEEE, pages 1–6.

James Gibson, Nikolaos Malandrakis, Francisco Romero, David C Atkins, and Shrikanth Narayanan. 2015. Predicting therapist empathy in motivational interviews using language features inspired by psycholinguistic norms. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Kathy Goggin, Mary M Gerkovich, Karen B Williams, Julie W Banderas, Delwyn Catley, Jannette Berkley-Patton, Glenn J Wagner, James Stanford, Sally Neville, Vinutha K Kumar, et al. 2013. A randomized controlled trial examining the efficacy of motivational counseling with observed therapy for antiretroviral therapy adherence. *AIDS and Behavior* 17(6):1992–2001.

Amy L Gonzales, Jeffrey T Hancock, and James W Pennebaker. 2009. Language style matching as a predictor of social dynamics in small groups. *Communication Research* .

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 423–430.

Sarah Peregrine Lord, Elisa Sheng, Zac E Imel, John Baer, and David C Atkins. 2015. More than reflections: Empathy in motivational interviewing includes language style synchrony between therapist and client. *Behavior therapy* 46(3):296–303.

Matthew Marge, Satanjeev Banerjee, Alexander Rudnicky, et al. 2010. Using the amazon mechanical turk for transcription of spoken language. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, pages 5270–5273.

William R Miller and Stephen Rollnick. 2013. *Motivational interviewing: Helping people change, Third edition*. The Guilford Press.

Theresa B. Manuel Jennifer K. Ernst Denise Moyers. 2014. *Motivational Interviewing Treatment Integrity Coding Manual 4.1. Unpublished manual.*.

Theresa B Moyers and William R Miller. 2013. Is low therapist empathy toxic? *Psychology of Addictive Behaviors* 27(3):878.

Kate G Niederhoffer and James W Pennebaker. 2002. Linguistic style matching in social interaction. *Journal of Language and Social Psychology* 21(4):337–360.

Verónica Pérez-Rosas, Rada Mihalcea, Kenneth Resnicow, Lawrence An, and Satinder Singh. 2016. Building a motivational interviewing dataset. In *NAACL Workshop on Clinical Psychology*.

Nairan Ramirez-Esparza, Cindy K. Chung, Ewa Kacewicz, and James W. Pennebaker. 2008. The psychology of word use in depression forums in english and in spanish: Testing two text analytic approaches. In *In Proc. ICWSM 2008*.

Christina Regenbogen, Daniel A Schneider, Andreas Finkelmeyer, Nils Kohn, Birgit Derntl, Thilo Kellermann, Raquel E Gur, Frank Schneider, and Ute Habel. 2012. The differential contribution of facial expressions, prosody, and speech content to empathy. *Cognition & emotion* 26(6):995–1014.

Catherine M Reich, Jeffrey S Berman, Rick Dale, and Heidi M Levitt. 2014. Vocal synchrony in psychotherapy. *Journal of Social and Clinical Psychology* 33(5):481.

Stephen Rollnick, William R Miller, Christopher C Butler, and Mark S Aloia. 2008. Motivational interviewing in health care: helping patients change behavior. *COPD: Journal of Chronic Obstructive Pulmonary Disease* 5(3):203–203.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology* 29(1):24–54.

Wayne F Velicer and Joseph L Fava. 1998. Affects of variable and subject sampling on factor pattern recovery. *Psychological methods* 3(2):231.

Steven R. Wilson, Rada Mihalcea, Ryan L. Boyd, and James W. Pennebaker. 2016. *Cultural influences on the measurement of personal values through words*, AI Access Foundation, volume SS-16-01 - 07, pages 314–317.

Markus Wolf, Cindy K Chung, and Hans Kordy. 2010. Inpatient treatment to online aftercare: e-mailing themes as a function of therapeutic outcomes. *Psychotherapy Research* 20(1):71–85.

Bo Xiao, Daniel Bone, Maarten Van Segbroeck, Zac E Imel, David C Atkins, Panayiotis G Georgiou, and Shrikanth S Narayanan. 2014. Modeling therapist empathy through prosody in drug addiction counseling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Bo Xiao, Dogan Can, Panayiotis G Georgiou, David Atkins, and Shrikanth S Narayanan. 2012. Analyzing the language of therapist empathy in motivational interview based psychotherapy. In *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*. IEEE, pages 1–4.

Bo Xiao, Zac E Imel, Panayiotis G Georgiou, David C Atkins, and Shrikanth S Narayanan. 2015. " rate my therapist": Automated detection of empathy in drug and alcohol counseling via speech and language processing. *PloS one* 10(12):e0143055.

Leveraging Knowledge Bases in LSTMs for Improving Machine Reading

Bishan Yang
Machine Learning Department
Carnegie Mellon University
bishan@cs.cmu.edu

Tom Mitchell
Machine Learning Department
Carnegie Mellon University
tom.mitchell@cs.cmu.edu

Abstract

This paper focuses on how to take advantage of external knowledge bases (KBs) to improve recurrent neural networks for machine reading. Traditional methods that exploit knowledge from KBs encode knowledge as discrete indicator features. Not only do these features generalize poorly, but they require task-specific feature engineering to achieve good performance. We propose KBLSTM, a novel neural model that leverages continuous representations of KBs to enhance the learning of recurrent neural networks for machine reading. To effectively integrate background knowledge with information from the currently processed text, our model employs an attention mechanism with a sentinel to adaptively decide whether to attend to background knowledge and which information from KBs is useful. Experimental results show that our model achieves accuracies that surpass the previous state-of-the-art results for both entity extraction and event extraction on the widely used ACE2005 dataset.

1 Introduction

Recurrent neural networks (RNNs), a neural architecture that can operate over text sequentially, have shown great success in addressing a wide range of natural language processing problems, such as parsing (Dyer et al., 2015), named entity recognition (Lample et al., 2016), and semantic role labeling (Zhou and Xu, 2015)). These neural networks are typically trained end-to-end where the input is only text or a sequence of words and a lot of background knowledge is disregarded.

The importance of background knowledge in natural language understanding has long been recognized (Minsky, 1988; Fillmore, 1976). Earlier NLP systems mostly exploited restricted linguistic knowledge such as manually-encoded morphological and syntactic patterns. With the advanced development of knowledge base construction, large amounts of semantic knowledge become available, ranging from manually annotated semantic networks like WordNet [1] to semi-automatically or automatically constructed knowledge graphs like DBPedia [2] and NELL [3]. While traditional approaches have exploited the use of these knowledge bases (KBs) in NLP tasks (Ratinov and Roth, 2009; Rahman and Ng, 2011; Nakashole and Mitchell, 2015), they require a lot of task-specific engineering to achieve good performance.

One way to leverage KBs in recurrent neural networks is by augmenting the dense representations of the networks with the symbolic features derived from KBs. This is not ideal as the symbolic features have poor generalization ability. In addition, they can be highly sparse, e.g., using WordNet synsets can easily produce millions of indicator features, leading to high computational cost. Furthermore, the usefulness of knowledge features varies across contexts, as general KBs involve polysemy, e.g., "Clinton" can refer to a person or a town. Incorporating KBs irrespective of the textual context could mislead the machine reading process.

Can we train a recurrent neural network that learns to adaptively leverage knowledge from KBs to improve machine reading? In this paper, we propose KBLSTM, an extension to bidirec-

[1] https://wordnet.princeton.edu
[2] http://wiki.dbpedia.org/
[3] http://rtw.ml.cmu.edu/rtw/kbbrowser/

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1436–1446
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1132

tional Long Short-Term Memory neural networks (BiLSTMs) (Hochreiter and Schmidhuber, 1997; Graves et al., 2005) that is capable of leveraging symbolic knowledge from KBs as it processes each word in the text. At each time step, the model retrieves KB concepts that are potentially related to the current word. Then, an attention mechanism is employed to dynamically model their semantic relevance to the reading context. Furthermore, we introduce a sentinel component in BiLSTMs that allows flexibility in deciding whether to attend to background knowledge or not. This is crucial because in some cases the text context should override the context-independent background knowledge available in general KBs.

In this work, we leverage two general, readily available knowledge bases: WordNet (WordNet, 2010) and NELL (Mitchell et al., 2015). WordNet is a manually created lexical database that organizes a large number of English words into sets of synonyms (i.e. synsets) and records conceptual relations (e.g., hypernym, part_of) among them. NELL is an automatically constructed web-based knowledge base that stores beliefs about entities. It is organized based on an ontology of hundreds of semantic categories (e.g., person, fruit, sport) and relations (e.g., personPlaysInstrument). We learn distributed representations (i.e., embeddings) of WordNet and NELL concepts using knowledge graph embedding methods. We then integrate these learned embeddings with the state vectors of the BiLSTM network to enable knowledge-aware predictions.

We evaluate the proposed model on two core information extraction tasks: entity extraction and event extraction. For entity extraction, the model needs to recognize all mentions of entities such as person, organization, location, and other things from text. For event extraction, the model is required to identify event mentions or event triggers[4] that express certain types of events, e.g., elections, attacks, and travels. Both tasks are challenging and often require the combination of background knowledge and the text context for accurate prediction. For example, in the sentence "Maigret left viewers in tears.", knowing that "Maigret" can refer to a TV show can greatly help disambiguate its meaning. However, knowledge bases may hurt performance if used blindly. For example, in the sentence "Santiago is charged with murder.", methods that rely heavily on KBs are likely to interpret "Santiago" as a location due to the popular use of Santiago as a city. Similarly for events, the same word can trigger different types of events, for example, "release" can be used to describe different events ranging from book publishing to parole. It is important for machine learning models to learn to decide which knowledge from KBs is relevant given the context.

Extensive experiments demonstrate that our KBLSTM models effectively leverage background knowledge from KBs in training BiLSTM networks for machine reading. They achieve significant improvement on both entity and event extraction compared to traditional feature-based methods and LSTM networks that disregard knowledge in KBs, resulting in new state-of-the-art results for entity extraction and event extraction on the widely used ACE2005 dataset.

2 Related Work

Essential to RNNs' success on natural language processing is the use of Long Short-Term Memory neural networks (Hochreiter and Schmidhuber, 1997) (LSTMs) or Gated Recurrent Unit (Cho et al., 2014) (GRU) as they are able to handle long-term dependencies by adaptively memorizing values for either long or short durations. Their bidirectional variants BiLSTM (Graves et al., 2005) or BiGRU further allow the incorporation of both past and future information. Such ability has been shown to be generally helpful in various NLP tasks such as named entity recognition (Huang et al., 2015; Chiu and Nichols, 2016; Ma and Hovy, 2016), semantic role labeling (Zhou and Xu, 2015), and reading comprehension (Hermann et al., 2015; Chen et al., 2016). In this work, we also employ the BiLSTM architecture.

In parallel to the development for text processing, neural networks have been successfully used to learn distributed representations of structured knowledge from large KBs (Bordes et al., 2011, 2013; Socher et al., 2013; Yang et al., 2015; Guu et al., 2015). Embedding the symbolic representations into continuous space not only makes KBs more easy to use in statistical learning approaches, but also offers strong generalization ability. Many attempts have been made on connecting distributed representations of KBs with text in the

[4] An event also consists of participants whose types depend on the event triggers. In this work, we focus on predicting event triggers and leave the prediction of event participants for future work.

context of knowledge base completion (Lao et al., 2011; Gardner et al., 2014; Toutanova et al., 2015), relation extraction (Weston et al., 2013; Chang et al., 2014; Riedel et al., 2013), and question answering (Miller et al., 2016). However, these approaches model text using shallow representations such as subject/relation/object triples or bag of words. More recently, Ahn et al. (2016) proposed a neural knowledge language model that leverages knowledge bases in RNN language models, which allows for better representations of words for language modeling. Unlike their work, we leverage knowledge bases in LSTMs and applies it to information extraction.

The architecture of our KBLSTM model draws on the development of attention mechanisms that are widely employed in tasks such as machine translation (Bahdanau et al., 2015) and image captioning (Xu et al., 2015). Attention allows the neural networks to dynamically attend to salient features of the input. With a similar motivation, we employ attention in KBLSTMs to allow for dynamic attention to the relevant knowledge given the text context. Our model is also closely related to a recent model of caption generation introduced by Lu et al. (2017), where a visual sentinel is introduced to allow the decoder to decide whether to attend to image information when generating the next word. In our model, we introduce a sentinel to control the tradeoff between background knowledge and information from the text.

3 Method

In this section, we present our KBLSTM model. We first describe several basic recurrent neural network frameworks for machine reading, including basic RNNs, LSTMs, and bidirectional LSTMs (Sec. § 3.1). We then introduce our extension to bidirectional LSTMs that allows for the incorporation of KB information at each time step of reading (Sec. § 3.2). The KB information is encoded using continuous representations (i.e., embeddings) which are learned using knowledge embedding methods (Sec. § 3.3).

3.1 RNNs, LSTMs, and Bidirectional LSTMs

RNNs are a class of neural networks that take a sequence of inputs and compute a hidden state vector at each time step based on the current input and the entire history of inputs. The hidden state vector can be computed recursively using the following equation (Elman, 1990):

$$\mathbf{h}_t = F(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t)$$

where $\mathbf{x}_t$ is the input at time step t, $\mathbf{h}_t$ is the hidden state at time step t, $\mathbf{U}$ and $\mathbf{W}$ are weight matrices, and F is a nonlinear function such as tanh or ReLu. Depending on the applications, RNNs can produce outputs based on the hidden state of each time step or just the last time step.

A Long Short-Term Memory network (Hochreiter and Schmidhuber, 1997) (LSTM) is a variant of RNNs which was design to better handle cases where the output at time t depends on much earlier inputs. It has a memory cell and three gating units: an input gate that controls what information to add to the current memory, a forget gate which controls what information to remove from the previous memory, and an output gate which controls what information to output from the current memory. Each gate is implemented as a logistic function σ that takes as input the previous hidden state and the current input, and outputs values between 0 and 1. Multiplication with these values controls the flow of information into or out of the memory. In equations, the updates at each time step t are:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{h}_{t-1} + \mathbf{U}_i\mathbf{x}_t) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{x}_t) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{x}_t) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot tanh(\mathbf{W}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{x}_t) \\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t)
\end{aligned}$$

where $\mathbf{i}_t$ is the input gate, $\mathbf{f}_t$ is the forget gate, $\mathbf{o}_t$ is the output gate, $\mathbf{c}_t$ is the memory cell, and $\mathbf{h}_t$ is the hidden state. $\odot$ denotes element-wise multiplication. $\mathbf{W}_i, \mathbf{U}_i, \mathbf{W}_f, \mathbf{U}_f, \mathbf{W}_o, \mathbf{U}_o, \mathbf{W}_c, \mathbf{U}_c$ are weight matrices to be learned.

Bidirectional LSTMs (Graves et al., 2005) (BiLSTMs) are essentially a combination of two LSTMs in two directions: one operates in the forward direction and the other operates in the backward direction. This leads to two hidden states $\overrightarrow{\mathbf{h}_t}$ and $\overleftarrow{\mathbf{h}_t}$ at time step t, which can be viewed as a summary of the past and the future respectively. Their concatenation $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$ provides a whole summary of the information about the input around time step t. Such property is attractive in NLP tasks, since information of both previous words and future words can be helpful for interpreting the meaning of the current word.

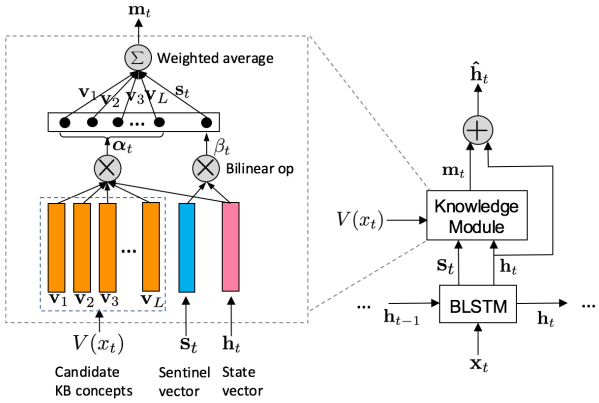


Figure 1: Architecture of the KBLSTM model. As each time step t, the knowledge module retrieves a set of candidate KB concepts $V(x_t)$ that are related to the current input x_t, and then computes a knowledge state vector $\mathbf{m}_t$ that integrates the embeddings of the candidate KB concepts $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_L$ and the current context vector $\mathbf{s}_t$. See Section § 3.2 for details.

3.2 Knowledge-aware Bidirectional LSTMs

Our model (referred to as KBLSTM) extends BiLSTMs to allow flexibility in incorporating symbolic knowledge from KBs. Instead of encoding knowledge as discrete features, we encode it using continuous representations. Concretely, we learn embeddings of concepts in KBs using a knowledge graph embedding method. (We will describe the details in Section § 3.3). The KBLSTM model then retrieves the embeddings of candidate concepts that are related to the current word being processed and integrates them into its state vector to make knowledge-aware predictions. Figure 1 depicts the architecture of our model.

The core of our model is the knowledge module, which is responsible for transferring background knowledge into the BiLSTMs. The knowledge at time step t consists of candidate KB concepts $V(x_t)$ for input x_t. (We will describe how to obtain the candidate KB concepts from NELL and WordNet in Section § 3.3). Each candidate KB concept $i \in V(x_t)$ is associated with a vector embedding $\mathbf{v}_i$. We compute an attention weight α_{ti} for concept vector $\mathbf{v}_i$ via a bilinear operator, which reflects how relevant or important concept i is to the current reading context $\mathbf{h}_t$:

$$\alpha_{ti} \propto \exp(\mathbf{v}_i^T \mathbf{W}_v \mathbf{h}_t) \quad (1)$$

where $\mathbf{W}_v$ is a parameter matrix to be learned.

Note that the candidate concepts in some cases are misleading. For example, a KB may store the fact that "Santiago" is a city but miss the fact that it can also refer to a person. Incorporating such knowledge in the sentence "Santiago is charged with murder." could be misleading. To address this issue, we introduce a knowledge sentinel that records the information of the current context and use a mixture model to allow for better tradeoff between the impact of background knowledge and information from the context. Specifically, we compute a sentinel vector $\mathbf{s}_t$ as:

$$\mathbf{b}_t = \sigma(\mathbf{W}_b \mathbf{h}_{t-1} + \mathbf{U}_b \mathbf{x}_t) \quad (2)$$

$$\mathbf{s}_t = \mathbf{b}_t \odot tanh(\mathbf{c}_t) \quad (3)$$

where $\mathbf{W}_b$ and $\mathbf{U}_b$ are weight parameters to be learned. The weight on the local context is computed as:

$$\beta_t \propto \exp(\mathbf{s}_t^T \mathbf{W}_s \mathbf{h}_t) \quad (4)$$

where $\mathbf{W}_s$ is a parameter matrix to be learned. The mixture model is defined as:

$$\mathbf{m}_t = \sum_{i \in V(x_t)} \alpha_{ti} \mathbf{v}_i + \beta_t \mathbf{s}_t \quad (5)$$

where $\sum_{i \in V(x_t)} \alpha_{ti} + \beta_t = 1$. $\mathbf{m}_t$ can be viewed as a knowledge state vector that encodes external KB information with respect to the input at time t. We combine it with the state vector $\mathbf{h}_t$ of BiLSTMs to obtain a knowledge-aware state vector $\hat{\mathbf{h}}_t$:

$$\hat{\mathbf{h}}_t = \mathbf{h}_t + \mathbf{m}_t \quad (6)$$

If $V(x_t) = \emptyset$, we set $\mathbf{m}_t = 0$. $\hat{\mathbf{h}}_t$ can be used for predictions in the same way as the original state vector $\mathbf{h}_t$ (see Section § 4 for details).

3.3 Embedding Knowledge Base Concepts

Now we describe how to learn embeddings of concepts in KBs. We consider two KBs: WordNet and NELL, which are both knowledge graphs that can be stored in the form of RDF[5] triples. Each triple consists of a subject entity, a relation, and an object entity. Examples of triples in WordNet are (*location*, *hypernym_of*, *city*), and (*door*, *has_part*, *lock*), where both the subject and object entities are synsets in WordNet. Examples of triples in NELL are (*New York*, *located_in*, *United States*)

[5] https://www.w3.org/TR/rdf11-concepts/

and (*New York*, *is_a*, *city*), where the subject entity is a noun phrase that can refer to a real-world entity and the object entity can be either a noun phrase entity or a concept category.

In this work, we refer to the synsets in WordNet and the concept categories in NELL as *KB concepts*. They are the key components of the ontologies and provide generally useful information for language understanding. As our KBLSTM model reads through each word in a sentence, it retrieves knowledge from NELL by searching for entities with the current word and collecting the related concept categories as candidate concepts; and it retrieves knowledge from WordNet by treating the synsets of the current word as candidate concepts.

We employ a knowledge graph embedding approach to learn the representations of the candidate concepts. Denote a KB triple as (e_1, r, e_2), we want to learn embeddings of the subject entity e_1, the object entity e_2, and the relation r, so that the relevance of the triple can be measured by a scoring function based on the embeddings. We employ the BILINEAR model described in (Yang et al., 2015).[6] It computes the score of a triple (e_1, r, e_2) via a bilinear function: $S_{(e_1,r,e_2)} = \mathbf{v}_{e_1}^T \mathbf{M}_r \mathbf{v}_{e_2}$, where $\mathbf{v}_{e_1}$ and $\mathbf{v}_{e_2}$ are vector embeddings for e_1 and e_2 respectively, and $\mathbf{M}_r$ is a relation-specific embedding matrix. We train the embeddings using the max-margin ranking objective:

$$\sum_{q=(e_1,r,e_2)\in\mathcal{T}} \sum_{q'=(e_1,r,e_2')\in\mathcal{T}'} \max\{0, 1 - S_q + S_{q'}\} \tag{7}$$

where $\mathcal{T}$ denotes the set of triples in the KB and $\mathcal{T}'$ denotes the "negative" triples that are not observed in the KB.

For WordNet, we train the concept embeddings using the preprocessed data provided by (Bordes et al., 2013), which contains 151,442 triples with 40,943 synsets and 18 relations. We use the same data splits for training, development, and testing. During training, we use AdaGrad (Duchi et al., 2011) to optimize objective 7 with an initial learning rate of 0.05 and a mini-batch size of 100. At each gradient step, we sample 10 negative object entities with respect to the positive triple. Our implementation of the BILINEAR model achieves top-10 accuracy of 91% for predicting missing object entities on the WordNet test set, which is comparable with previous work (Yang et al., 2015).

For NELL, we train the concept embeddings using a subset of the NELL data[7]. We filter noun phrases with annotation confidence less than 0.9 in order to remove erroneous labels introduced during the automatic construction of NELL (Wijaya, 2016). This results in 180,107 noun phrases and 258 concept categories in total. We randomly split 80% of the data for training, 10% for development and 10% for testing. For each training example, we enumerate all the unobserved concept categories as negative labels. Instead of treating each entity as a unit, we represent it as an average of its constituting word vectors concatenated with its head word vector. The word vectors are initialized with pre-trained paraphrastic embeddings (Wieting et al., 2015) and fine-tuned during training. Using the same optimization parameters as before, the BILINEAR model achieves 88% top-1 accuracy for predicting concept categories of given noun phrases on the test set.

4 Experiments

4.1 Entity Extraction

We first apply our model to entity extraction, a task that is typically addressed by assigning each word/token BIO labels ($Begin$, $Inside$, and $Outside$) (Ratinov and Roth, 2009) indicating the token's position within an entity mention as well as its entity type.

To allow tagging over phrases instead of words, we address entity extraction in two steps. The first step detects mention chunks, and the second step assigns entity type labels to mention chunks (including an O type indicating *other* types). In the first stage, we train a BiLSTM network with a conditional random field objective (Huang et al., 2015) using gold-standard BIO labels regardless of entity types, and only predict each token's position within an entity mention. This produces a sequence of chunks for each sentence. In the second stage, we train another supervised BiLSTM model to predict type labels for the previously extracted chunks. Each chunk is treated as a unit input to the BiLSTMs and the input vector is computed by averaging the input vectors of individual words within the chunk concatenated with its head word vector. The BiLSTMs can be trained

[6] We also experimented with TransE (Bordes et al., 2013) and NTN (Socher et al., 2013), and found that they perform significantly worse than the Bilinear method, especially on predicting the "is_a" facts in NELL.

[7] http://rtw.ml.cmu.edu/rtw/resources

with a softmax objective that minimizes the cross-entropy loss for each individual chunk. It computes the probability of the correct type for each chunk:

$$p_{y_t} = \frac{\exp(\mathbf{w}_{y_t}^T \mathbf{h}_t)}{\sum_{y'_t} \exp(\mathbf{w}_{y'_t}^T \mathbf{h}_t)} \quad (8)$$

The BiLSTMs can also be trained with a CRF objective (referred to as BiLSTM-CRF) that minimizes the negative log-likelihood for the entire sequence. It computes the probability of the correct types for a sequence of chunks:

$$p_{\mathbf{y}} = \frac{\exp(g(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp(g(\mathbf{x}, \mathbf{y}'))} \quad (9)$$

where $g(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{l} P_{t,y_t} + \sum_{t=0}^{l} A_{y_t,y_{t+1}}$, $P_{t,y_t} = \mathbf{w}_{y_t}^T \mathbf{h}_t$ is the score of assigning the t-th chunk with tag y_t and $A_{y_t,y_{t+1}}$ is the score of transitioning from tag y_t to y_{t+1}. By replacing $\mathbf{h}_t$ in Eq. 8 and Eq. 9 with the knowledge-aware state vector $\hat{\mathbf{h}}_t$ (Eq. 6), we can compute the objective for KBLSTM and KBLSTM-CRF respectively.

4.1.1 Implementation Details

We evaluate our models on the ACE2005 corpus (LDC, 2005) and the OntoNotes 5.0 corpus (Hovy et al., 2006) for entity extraction. Both datasets consist of text from a variety of sources such as newswire, broadcast conversations, and web text. We use the same data splits and task settings for ACE2005 as in Li et al. (2014) and for OntoNotes 5.0 as in Durrett and Klein (2014).

At each time step, our models take as input a word vector and a capitalization feature (Chiu and Nichols, 2016). We initialize the word vectors using pretrained paraphrastic embeddings (Wieting et al., 2015), as we find that they significantly outperforms randomly initialized embeddings. The word embeddings are fine-tuned during training. For the KBLSTM models, we obtain the embeddings of KB concepts from NELL and WordNet as described in Section § 3.3. These embeddings are kept fix during training.

We implement all the models using Theano on a single GPU. We update the model parameters on every training example using Adam with default settings (Kingma and Ba, 2014) and apply dropout to the input layer of the BiLSTM with a rate of 0.5. The word embedding dimension is set to 300 and the hidden vector dimension is set to 100. We train models on ACE2005 for about 5 epochs and on OntoNotes 5.0 for about 10 epochs with early stopping based on development results.

For each experiment, we report the average results over 10 random runs. We also apply the Wilcoxon rank sum test to compare our models with the baseline models.

Model	P	R	F1
BiLSTM	83.5	86.4	84.9
BiLSTM-CRF	87.3	84.7	86.0
BiLSTM-Fea	86.1	84.7	85.4
BiLSTM-Fea-CRF	87.7	86.1	86.9
KBLSTM	87.8	86.6	87.2
KBLSTM-CRF	**88.1**	**87.8**	**88.0***

Table 1: Entity extraction results on the ACE2005 test set with gold-standard mention boundaries.

4.1.2 Results

We compare our KBLSTM and KBLSTM-CRF models with the following baselines: BiLSTM, a vanilla BiLSTM network trained using the same input, and BiLSTM-Fea, a BiLSTM network that combines its hidden state vector with discrete KB features (i.e., indicators of candidate KB concepts) to produce the final state vector. We also include their variants BiLSTM-CRF and BiLSTM-Fea-CRF, which are trained using the CRF objective instead of the standard softmax objective.

We first report results on entity extraction with gold-standard boundaries for multi-word mentions. This allows us to focus on the performance of entity type prediction without considering mention boundary errors and the noise they introduce in retrieving candidate KB concepts. Table 1 shows the results.[8] We find that the CRF objective generally outperforms the softmax objective. Our KBLSTM-CRF model significantly improves over its counterpart BiLSTM-Fea-CRF. This demonstrates the effectiveness of KBLSTM architecture in leveraging KB information.

Table 2 breaks down the results of the KBLSTM-CRF and the BiLSTM-Fea-CRF using different KB settings. We find that the KBLSTM-CRF outperforms the BiLSTM-Fea-CRF in all the settings and that incorporating both KBs leads to the best performance.

Next, we evaluate our models on entity extraction with predicted mention boundaries. We first train a BiLSTM-CRF to perform mention

[8] $*$ indicates $p < 0.05$ when comparing to the BiLSTM-based models.

Model	KB	P	R	F1
BiLSTM-Fea-CRF	NELL	87.2	86.1	86.6
	WordNet	86.4	86.0	86.2
	Both	87.7	86.1	86.9
KBLSTM-CRF	NELL	87.4	87.6	87.5
	WordNet	87.1	87.4	87.3
	Both	**88.1**	**87.8**	**88.0**

Table 2: Ablation results with different KBs.

Model	P	R	F1
Li and Ji (2014)	85.2	76.9	80.8
Yang and Mitchell (2016)	83.5	80.2	81.8
Miwa and Bansal (2016)	82.9	83.9	83.4
BiLSTM	82.5	83.1	82.8
BiLSTM-CRF	84.6	82.5	83.6
BiLSTM-Fea	84.3	83.2	83.7
BiLSTM-Fea-CRF	84.7	83.5	84.1
KBLSTM	85.5	85.2	85.3
KBLSTM-CRF	**85.4**	**86.0**	**85.7***

Table 3: Entity extraction results on the ACE2005 test set.

Model	P	R	F1
Ratinov and Roth (2009)	82.0	84.9	83.4
Durrett and Klein (2014)	85.2	82.8	84.0
BiLSTM-CNN	82.5	82.4	82.5
BiLSTM-CNN+emb	85.9	86.3	86.1
BiLSTM-CNN+emb+lexicon	86.0	86.5	86.2
BiLSTM	84.9	86.3	85.6
BiLSTM-CRF	85.3	86.6	85.9
BiLSTM-Fea	85.2	86.4	85.8
BiLSTM-Fea-CRF	85.2	86.8	86.0
KBLSTM	**86.3**	86.2	86.2
KBLSTM-CRF	86.1	**86.8**	**86.4***

Table 4: Entity extraction results on the OntoNotes 5.0 test set.

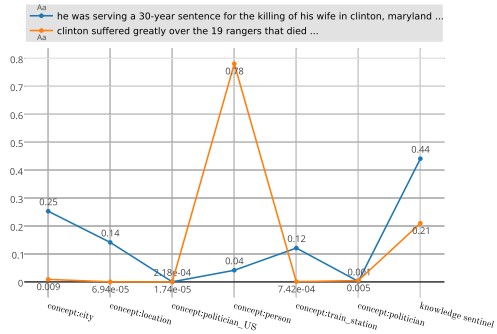
chunking using the same setting as described in Section 4.1.1. We then treat the predicted chunks as units for entity type labeling. Table 3 reports the full entity extraction results on the ACE2005 test set. We compare our models with the state-of-the-art feature-based linear models Li et al. (2014), Yang and Mitchell (2016), and the recently proposed sequence- and tree-structured LSTMs (Miwa and Bansal, 2016). Interestingly, we find that using BiLSTM-CRF without any KB information already gives strong performance compared to previous work. The KBLSTM-CRF model demonstrates the best performance among all the models and achieves the new state-of-the-art performance on the ACE2005 dataset.

We also report the entity extraction results on the OntoNotes 5.0 test set in Table 4. We compare our models with the existing feature-based models Ratinov and Roth (2009) and Durrett and Klein (2014), which both employ heavy feature engineering to bring in external knowledge. BiLSTM-CNN (Chiu and Nichols, 2016) employs a hybrid BiLSTM and Convolutional neural network (CNN) architecture and incorporates rich lexicon features derived from SENNA and DBPedia. Our KBLSTM-CRF model shows competitive results compared to their results. It also presents significant improvements compared to the BiLSTM and BiLSTM-Fea models. Note that the benefit of adding KB information is smaller on OntoNotes compared to ACE2005. We believe that there are two main reasons. One is that NELL has a lower coverage of entity mentions in OntoNotes than in ACE2005 (57% vs. 65%). Second, OntoNotes has a significantly larger amount of training data, which allows for more accurate models without much help from external resources.

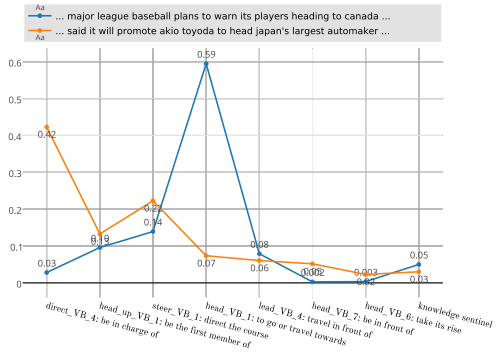
4.2 Event Extraction

We now apply our model to the task of event extraction. Event extraction is concerned with detecting *event triggers*, i.e., a word that expresses the occurrence of a pre-defined type of event. Event triggers are mostly verbs and eventive nouns but can occasionally be adjectives and other content words. Therefore, the task is typically addressed as a classification problem where the goal is to label each word in a sentence with an event type or an O type if it does not express any of the defined events. It is straightforward to apply the BiLSTM architecture to event extraction. Similarly to the models for entity extraction, we can train the BiLSTM network with both the softmax objective and the CRF objective.

We evaluate our models on the portion ACE2005 corpus that has event annotations. We use the same data split and experimental setting as in Li et al. (2013). The training procedure is the same as in Section 4.1.1, and we train all the models for about 5 epochs. For the KBLSTM models, we integrate the learned embeddings of WordNet synsets during training.



(a) The X-axis represents relevant NELL concepts for the entity mention **clinton**. The Y-axis represents the concept weights and the knowledge sentinel weight.



(b) The X-axis represents relevant WordNet concepts for the event trigger **head**. The Y-axis represents the concept weights and the knowledge sentinel weight.

Figure 2: Visualization of the attention weights for KB features learned by KBLSTM-CRF. Higher weights imply higher importance.

Model	P	R	F1
JOINTBEAM	74.0	56.7	64.2
JOINTEVENTENTITY	**75.1**	63.3	68.7
DMCNN	71.8	63.8	69.0
JRNN	66.0	73.0	69.3
BiLSTM	71.3	59.3	64.7
BiLSTM-CRF	64.2	66.6	65.4
BiLSTM-Fea	68.4	62.7	65.5
BiLSTM-Fea-CRF	65.5	66.7	66.1
KBLSTM	70.1	67.3	68.7
KBLSTM-CRF	71.6	67.8	**69.7***

Table 5: event extraction results on the ACE2005 test set.

4.2.1 Results

We compare our models with the prior state-of-the-art approaches for event extraction, including neural and non-neural ones: JOINTBEAM refers to the joint beam search approach with local and global features (Li et al., 2013); JOINTENTITYEVENT refers to the graphical model for joint entity and event extraction (Yang and Mitchell, 2016); DMCNN is the dynamic multi-pooling CNNs in Chen et al. (2015); and JRNN is an RNN model with memory introduced by Nguyen et al. (2016). The first block in Table 5 shows the results of the feature-based linear models (taken from Yang and Mitchell (2016)). The second block shows the previously reported results for the neural models. Note that they both make use of gold-standard entity annotations. The third block shows the results of our models. We can see that our KBLSTM models significantly outperform the BiLSTM and BiLSTM-Fea models, which again confirms their effectiveness in leveraging KB information. The KBLSTM-CRF model achieves the best performance and outperforms the previous state-of-the-art methods without having access to any gold-standard entities.

4.3 Model Analysis

In order to better understand our model, we visualize the learned attention weights $\boldsymbol{\alpha}$ for KB concepts and the sentinel weight β that measures the tradeoff between knowledge and context. Figure 2a visualizes these weights for the entity mention "clinton". In the first sentence, "clinton" refers to a LOCATION while in the second sentence, "clinton" refers to a PERSON. Our model learns to attend to different word senses for 'clinton' (KB concepts associated with 'clinton') in different sentences. Note that the weight on the knowledge sentinel is higher in the first sentence. This is because the local text alone is indicative of the entity type for "clinton": the word "in" is more likely to be followed by a location than a person. We find that BiLSTM-Fea-CRF models often make wrong predictions on examples like this due to its inflexibility in modeling knowledge relevance with respect to context. Figure 2b shows the learned weights for the event trigger word "head" in two sentences, one expresses a TRAVEL event while the other expresses a STARTPOSITION event. Again, we find that our model is capable of attending to relevant WordNet synsets and more accurately disambiguate event types.

5 Conclusion

In this paper, we introduce the KBLSTM network architecture as one approach to incorporating background KBs for improving recurrent neural networks for machine reading. This architecture employs an adaptive attention mechanism with a sentinel that allows for learning an appropriate tradeoff for blending knowledge from the KBs with information from the currently processed text, as well as selecting among relevant KB concepts for each word (e.g., to select the correct semantic categories for "clinton" as a town or person in figure 2a). Experimental results show that our model achieves state-of-the-art performance on standard benchmarks for both entity extraction and event extraction.

We see many additional opportunities to integrate background knowledge with training of neural network models for language processing. Though our model is evaluated on entity extraction and event extraction, it can be useful for other machine reading tasks. Our model can also be extended to integrate knowledge from a richer set of KBs in order to capture the diverse variety and depth of background knowledge required for accurate and deep language understanding.

Acknowledgments

This research was supported in part by DARPA under contract number FA8750-13-2-0005, and by NSF grants IIS-1065251 and IIS-1247489. We also gratefully acknowledge the support of the Microsoft Azure for Research program and the AWS Cloud Credits for Research program. In addition, we would like to thank anonymous reviewers for their helpful comments.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*. pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 167–176.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics* 4:357–370.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics* 2:477–490.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Charles J Fillmore. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences* 280(1):20–32.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*. Springer, pages 799–804.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*. pages 1693–1701.

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

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*. pages 57–60.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations (ICLR)* .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 529–539.

LDC. 2005. The ace 2005 evaluation plan. In *NIST*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 402–412.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1846–1851.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 73–82.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Marvin Minsky. 1988. *Society of mind*. Simon and Schuster.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)* .

Ndapandula Nakashole and Tom M Mitchell. 2015. A knowledge-intensive model for prepositional phrase attachment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 365–375.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*. pages 300–309.

Altaf Rahman and Vincent Ng. 2011. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 814–824.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)*. pages 147–155.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of HLT-NAACL*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems (NIPS)*. pages 926–934.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Association for Computational Linguistics (ACL)*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *Proceedings of the International Conference on Learning Representations (ICLR)* .

Derry Tanti Wijaya. 2016. *VerbKB: A Knowledge Base of Verbs for Natural Language Understanding*. Ph.D. thesis, Carnegie Mellon University.

WordNet. 2010. About wordnet. http://wordnet.princeton.edu.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference for Machine Learning (ICML)*.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*. pages 289–299.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *International Conference on Learning Representations (ICLR)* .

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Prerequisite Relation Learning for Concepts in MOOCs

Liangming Pan, Chengjiang Li, Juanzi Li* and Jie Tang
Knowledge Engineering Laboratory
Department of Computer Science and Technology
Tsinghua University, Beijing 100084, China (* corresponding author)
{panlm14@mails,licj17@mails,lijuanzi,tangjie}tsinghua.edu.cn

Abstract

What prerequisite knowledge should students achieve a level of mastery before moving forward to learn subsequent coursewares? We study the extent to which the prerequisite relation between knowledge concepts in Massive Open Online Courses (MOOCs) can be inferred automatically. In particular, what kinds of information can be leveraged to uncover the potential prerequisite relation between knowledge concepts. We first propose a representation learning-based method for learning latent representations of course concepts, and then investigate how different features capture the prerequisite relations between concepts. Our experiments on three datasets form Coursera show that the proposed method achieves significant improvements (+5.9-48.0% by F1-score) comparing with existing methods.

1 Introduction

Mastery learning was first formally proposed by Benjamin Bloom in 1968 (Bloom, 1981), suggesting that students must achieve a level of mastery (e.g., 90% on a knowledge test) in prerequisite knowledge before moving forward to learn subsequent knowledge concepts. From then on, prerequisite relations between knowledge concepts become a cornerstone for designing curriculum in schools and universities. Prerequisite relations essentially can be considered as the dependency among knowledge concepts. It is crucial for people to learn, organize, apply, and generate knowledge (Laurence and Margolis, 1999). Figure 1 shows a real example from Coursera. The student wants to learn "Conditional Random Field" (in video18 of CS229). The prerequisite knowledge might be "Hidden Markov Model" (in video25 of CS224), whose prerequisite knowledge is "Maximum Likelihood" (in video12 of Math112).

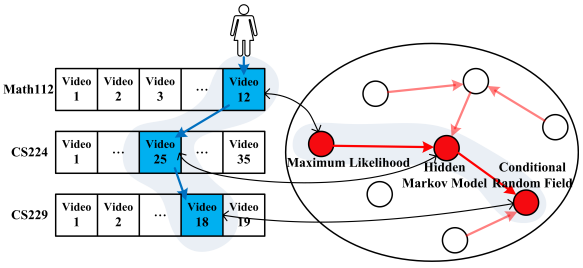


Figure 1: An example of prerequisite relations in MOOCs

Organizing the knowledge structure with prerequisite relations in education improves tasks such as curriculum planning (Yang et al., 2015), automatic reading list generation (Jardine, 2014), and improving education quality (Rouly et al., 2015). For example, as shown in Figure 1, with explicit prerequisite relations among concepts (in red), a coherent and reasonable learning sequence can be recommended to the student (in blue). Before, the prerequisite relationships were provided by teachers or teaching assistants (Novak, 1990); however in the era of MOOCs, it is becoming *infeasible* as the teachers would find that they are facing with hundreds of thousands of students with various background. Meanwhile, the rapid growth of Massive Open Online Courses has offered thousands of courses, and students are free to choose any course from the thousands of candidates. Therefore, there is a clear need for methods to automatically dig out the prerequisite relationships among knowledge concepts from the large course space, so that the students from different background can easily explore the knowledge space and better design their personalized learning schedule.

There are a few efforts aiming to automatically detect prerequisite relations for knowledge base. For example, Talukdar and Cohen (2012) proposed a method for inferring prerequisite relationships between entities in Wikipedia and Liang et al. (2015) presented a more general approach

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1447–1456
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1133

to predict prerequisite relationships. A few other works intend to extract prerequisite relationships from textbooks (Yosef et al., 2011; Wang et al., 2016). However, it is far from sufficient to directly apply these methods to the MOOC environments due to the following reasons. First, the focus of most previous attempts has been on prerequisite inference of Wikipedia concepts (either Wikipedia articles or Wikipedia concepts in textbooks). Many course concepts are not included in Wikipedia (Schweitzer, 2008; Okoli et al., 2014). We can leverage Wikipedia, in particular the existing entity relationships in Wikipedia, but cannot only rely on Wikipedia for detecting prerequisite relations in MOOCs. Second, with the thousands of courses from different universities and also very different disciplinaries, the MOOC scenario is much more complicated — there are not only inter-course concept relationships, but also intra-course and even intra-disciplinary relationships. Moreover, user interactions with the MOOC system might be also helpful to identify the prerequisite relations. How to fully leverage the different information to obtain a better performance for inferring prerequisite relations in MOOCs is a challenging issue.

In this paper, we attempt to figure out what kinds of information in MOOCs can be used to uncover the prerequisite relations among concepts. Specifically, we consider it from three aspects, including course concept semantics, course video context and course structure. First, semantic relatedness plays an important role in prerequisite relations between concepts. If two concepts have very different semantic meanings (e.g., "matrix" and "anthropology"), it is unlikely that they have prerequisite relations. However, statistical features in MOOCs do not provide sufficient information for capturing the concept semantics because of the short length of course videos in MOOCs, we propose an embedding-based method to incorporate external knowledge from Wikipedia to learn semantic representations of concepts in MOOCs. Based on it, we propose one **semantic feature** to calculate the semantic relatedness between concepts. Second, motivated by the *reference distance* (RefD) (Liang et al., 2015), we propose three new **contextual features**, i.e., Video Reference Distance, Sentence Reference Distance and Wikipedia Reference Distance, to infer prerequisite relations in MOOCs based on context information from different aspects, which are more general and informative than RefD and overcome its sparsity problem. Third, we examine different distributional patterns for concepts in MOOCs, including appearing position, distributional asymmetry, video coverage and survival time. We further propose three **structural features** to utilize these patterns to help prerequisite inference in MOOCs.

To evaluate the proposed method, we construct three datasets, each of which consists of multiple real courses in a specific domain from Coursera [1], the largest MOOC platform in the world. We also compare our method with the representative works of prerequisite learning and make a deep analysis of the feature contribution proposed in the paper. The experimental results show that our method achieves the state-of-the-art results in the prerequisite relation discovery in MOOCs. In summary, our contributions include: a) the first attempt, to the best of our knowledge, to detect prerequisite relations among concepts in MOOCs; b) proposal of a set of novel features that utilize contextual, structural and semantic information in MOOCs to identify prerequisite relations; c) design of three useful datasets based on real courses of Coursera to evaluate our method.

2 Problem Formulation

In this section, we first give some necessary definitions and then formulate the problem of prerequisite relation learning in MOOCs.

A **MOOC corpus** is composed by n courses in the same subject area, denoted as $\mathcal{D} = \{\mathcal{C}_1, \cdots, \mathcal{C}_i, \cdots, \mathcal{C}_n\}$, where $\mathcal{C}_i$ is one course. Each course $\mathcal{C}$ can be further represented as a video sequence $\mathcal{C} = (\mathcal{V}_1, \cdots, \mathcal{V}_i, \cdots, \mathcal{V}_{|\mathcal{C}|})$, where $\mathcal{V}_i$ denotes the i-th teaching video of course $\mathcal{C}$. Finally, we view each video $\mathcal{V}$ as a document of its video texts (video subtitles or speech script), i.e., $\mathcal{V} = (s_1 \cdots s_i \cdots s_{|\mathcal{V}|})$, where s_i is the i-th sentence of the video texts.

Course concepts are subjects taught in the course, i.e., the concepts not only mentioned but also discussed and taught in the course. Let us denote the course concept set of $\mathcal{D}$ as $\mathcal{K} = \mathcal{K}_1 \cup \cdots \cup \mathcal{K}_n$, where $\mathcal{K}_i$ is the set of course concepts in $\mathcal{C}_i$.

Prerequisite relation learning in MOOCs is formally defined as follows. Given a MOOC corpus $\mathcal{D}$ and its corresponding course concepts

[1] https://www.coursera.org/

$\mathcal{K}$, the objective is to learn a function $\mathcal{P} : \mathcal{K}^2 \rightarrow \{0, 1\}$ that maps a concept pair $\langle a, b\rangle$, where $a, b \in \mathcal{K}$, to a binary class that predicts whether a is a prerequisite concept of b.

In order to learn this mapping, we need to answer two crucial questions. How could we represent a course concept? What information regarding a concept pair is helpful to capture their prerequisite relation? We first propose an embedding-based method to learn appropriate semantic representations for each course concept in $\mathcal{K}$. Based on the learned representations, we propose 7 novel features to capture whether a concept pair has prerequisite relation. These features utilize different aspects of information and can be classified into 1 semantic feature, 3 contextual features and 3 structural features. In the following section, we first describe the semantic representations in detail, and then formally introduce our proposed features.

3 Method

3.1 Concept Representation & Semantic Relatedness

We first learn appropriate representations for course concepts. Given the course concepts $\mathcal{K}$ as input, we utilize a *Wikipedia corpus* to learn semantic representations for concepts in $\mathcal{K}$. A Wikipedia corpus $\mathcal{W}$ is a set of Wikipedia articles and can be represented as a sequence of words $\mathcal{W} = \langle w_1 \cdots w_i \cdots w_m \rangle$, where w_i denotes a word and m is the length of the word sequence. Our method consists of two steps: (1) entity annotation, and (2) representation learning.

Entity Annotation. We first automatically annotate the entities in $\mathcal{W}$ to obtain an **entity set** $\mathcal{E}$ and an **entity-annotated Wikipedia corpus** $\mathcal{W}' = \langle x_1 \cdots x_i \cdots x_{m'} \rangle$, where x_i corresponds to a word $w \in \mathcal{W}$ or an entity $e \in \mathcal{E}$. Note that $m' < m$ because multiple adjacent words could be labeled as one entity. Many entity linking tools are available for entity annotation, e.g. TAGME (Ferragina and Scaiella, 2010), AIDA (Yosef et al., 2011) and TremenRank (Cao et al., 2015). However, the rich hyperlinks created by Wiki editors provide a more natural way. In our experiments, we simply use the hyperlinks in Wikipedia articles as annotated entities.

Representation Learning. We then learn word embeddings (Mikolov et al., 2013b,a) on $\mathcal{W}'$ to obtain low-dimensional, real-valued vector representations for entities in $\mathcal{E}$ and words in $\mathcal{W}$. Let us denote v_e and v_w as the vector of $e \in \mathcal{E}$ and $w \in \mathcal{W}$, respectively. For a course concept $a \in \mathcal{K}$, suppose a is a N-gram term $\langle g_1 \cdots g_N \rangle$ and $g_1, \cdots, g_N \in \mathcal{W}$, we obtain its semantic representations v_a as follows.

$$v_a = \begin{cases} v_e, & \text{if } a \equiv e \text{ and } e \in \mathcal{E} \\ v_{g_1} + \cdots + v_{g_N}, & \text{otherwise} \end{cases} \quad (1)$$

It means that if a is a Wikipedia entity, we can directly obtain its semantic representations; otherwise, we obtain its vector via the vector addition of its individual word vectors. In this way, a has no corresponding vector only if any of its constituent word is absence in the whole Wikipedia corpus. This case is unusual because a large online encyclopedia corpus can easily cover almost all individual words of the vocabulary. Our experimental results verify that over 98% of the course concepts have vector representations.

Feature 1: *Semantic Relatedness*

For a given concept pair $\langle a, b\rangle$, the **semantic relatedness** between a and b, denoted as $\omega(a, b)$, is our first feature (the only semantic feature). With learned semantic representations, semantic relatedness of two concepts can be easily reflected by their distance in the vector space. We define $\omega(a, b) \in [0, 1]$ as the normalized cosine distance between v_a and v_b, as follows.

$$\omega(a, b) = \frac{1}{2}(1 + \frac{v_a \cdot v_b}{\|v_a\| \cdot \|v_b\|}) \quad (2)$$

3.2 Contextual Features

Context information in course videos provides important clues to infer prerequisite relations. In videos where concept A is taught, if the teacher also mentions concept B for a lot but not vice versa, then B is more likely to be a prerequisite of A than A of B. For example, "gradient descent" is a prerequisite concept of "back propagation". In teaching videos of "back propagation", the concept "gradient descent" is frequently mentioned when illustrating the optimization detail of back propagation. On the contrary, however, "back propagation" is unlikely to be mentioned when teaching "gradient descent". A similar observation also exists in Wikipedia, based on which Liang et al. (2015) proposed an indicator, namely *reference distance* (RefD), to infer prerequisite relations among Wikipedia articles. However, RefD is computed based on the link structure of Wikipedia, thus is only feasible for Wikipedia

concepts and is not applicable in plain text. We overcome the above shortcomings of RefD to propose three novel features, which utilize different aspects of context information—course videos, video sentences and Wikipedia articles—to infer prerequisite relations in MOOCs.

Feature 2: ***Video Reference Distance***

Given a concept pair $\langle a, b\rangle$ where $a, b \in \mathcal{K}$, we propose the **video reference weight** (Vrw) to quantify how b is referred by videos of a, defined as follows.

$$Vrw\left(a,b\right)=\frac{\sum\limits_{\mathcal{C}\in\mathcal{D}}\sum\limits_{\mathcal{V}\in\mathcal{C}}f\left(a,\mathcal{V}\right)\cdot r\left(\mathcal{V},b\right)}{\sum\limits_{\mathcal{C}\in\mathcal{D}}\sum\limits_{\mathcal{V}\in\mathcal{C}}f\left(a,\mathcal{V}\right)} \tag{3}$$

where $f\left(a,\mathcal{V}\right)$ indicates the term frequency of concept a in video $\mathcal{V}$, which reflects how important is concept a to this video. $r\left(\mathcal{V},b\right) \in \{0,1\}$ denotes whether concept b appears in video $\mathcal{V}$. Intuitively, if b appears in more important videos of a, $Vrw\left(a,b\right)$ tends to be larger, and the range of $Vrw\left(a,b\right)$ is between 0 and 1. Then, the **video reference distance** (Vrd) is defined as the difference of Vrw between two concepts, as follows.

$$Vrd\left(a,b\right)=Vrw\left(b,a\right)-Vrw\left(a,b\right) \tag{4}$$

In practice, this feature may be too sparse if the MOOC corpus is small. For an arbitrary concept pair, they may have no co-occurrence in all course videos. We expend the video reference distance to a more general version by considering the semantic relatedness among concepts. Besides the conditions in which A refers to B, we also consider the cases in which A-related concepts refer to B. We first define the **generalized video reference weight** ($GVrw$) as follows.

$$GVrw\left(a,b\right)=\frac{\sum_{i=1}^{M}Vrw\left(a_i,b\right)\cdot\omega\left(a_i,b\right)}{\sum_{i=1}^{M}\omega\left(a_i,b\right)} \tag{5}$$

where $a_1,\cdots,a_M \in \mathcal{K}$ are the top-M most similar concepts of a, measured by the semantic relatedness function $\omega(\cdot,\cdot)$ in feature 1. $GVrw$ is the weighted average of $Vrw\left(a_i,b\right)$, indicating how b is referred by a-related concepts in their corresponding videos. Note that $a_1 = a$, thus $GVrw\left(a,b\right) \equiv Vrw\left(a,b\right)$ when $M = 1$. Similarly, we define the **generalized video reference distance** ($GVrd$) as follows.

$$GVrd\left(a,b\right)=GVrw\left(b,a\right)-GVrw\left(a,b\right) \tag{6}$$

Intuitively, if most of b-related concepts refer to a but not vice versa, then a is likely to be a prerequisite of b. For example, it is plausible for the related concepts of "gradient descent", e.g., "steepest descent" and "Newton's method", to mention "matrix" but clearly not vice versa.

Feature 3: ***Sentence Reference Distance***

Sentence reference distance is similar to feature 2, but stands on the sentence level. Following the same design pattern of feature 2, we define the **sentence reference weight** (Srw) and **sentence reference distance** (Srd) as follows.

$$Srw\left(a,b\right)=\frac{\sum\limits_{\mathcal{C}\in\mathcal{D}}\sum\limits_{\mathcal{V}\in\mathcal{C}}\sum\limits_{s\in\mathcal{V}}r(s,a)\cdot r(s,b)}{\sum\limits_{\mathcal{C}\in\mathcal{D}}\sum\limits_{\mathcal{V}\in\mathcal{C}}\sum\limits_{s\in\mathcal{V}}r(s,a)} \tag{7}$$

$$Srd\left(a,b\right)=Srw\left(b,a\right)-Srw\left(a,b\right) \tag{8}$$

where $r\left(s,a\right) \in \{0,1\}$ is an indicator of whether concept a appears in sentence s. $Srw(a,b)$ calculates the ratio of B appearing in the sentences of a. We also define **generalized sentence reference weight** ($GSrw$) and **generalized sentence reference distance** ($GSrd$) as follows.

$$GSrw\left(a,b\right)=\frac{\sum_{i=1}^{M}Srw\left(a_i,b\right)\cdot\omega\left(a_i,b\right)}{\sum_{i=1}^{M}\omega\left(a_i,b\right)} \tag{9}$$

$$GSrd\left(a,b\right)=GSrw\left(b,a\right)-GSrw\left(a,b\right) \tag{10}$$

Feature 4: ***Wikipedia Reference Distance***

Contextual information of Wikipedia is also useful for detecting prerequisite relations. As mention before, RefD is not general enough to be applied in our settings, because it is limited to Wikipedia concepts. Therefore, we improve this indicator to a more general one, which is also suitable for non-wiki concepts.

Specifically, for a concept $a \in \mathcal{K}$, let us denote the top-M most related wiki entities of a as $\mathcal{R}_a = \langle e_1,\cdots,e_M\rangle$, where $e_1,\cdots,e_M \in \mathcal{E}$. Because concepts in $\mathcal{K}$ and entities in $\mathcal{E}$ are jointly embedded in the same vector space in Section 3.1, we can easily obtain $\mathcal{R}_a$ with the semantic relatedness metric $\omega(\cdot,\cdot)$ in Feature 1. We then define the **wikipedia reference weight** (Wrw) as follows.

$$Wrw\left(a,b\right)=\frac{\sum\limits_{e\in\mathcal{R}_a}Erw\left(e,b\right)\cdot\omega\left(e,a\right)}{\sum\limits_{e\in\mathcal{R}_a}\omega\left(e,a\right)} \tag{11}$$

where $Erw(e,a)$ is a binary indicator, in which $Erw(e,a) = 1$ if the Wikipedia article of e refers to any entity in $\mathcal{R}_a$, and $Erw(e,a) = 0$ otherwise. $Wrw\left(a,b\right)$ measures how frequently that a-related wiki entities refer to b-related wiki entities. Finally, **wikipedia reference distance** (Wrd) is

defined as the difference of Wrw between a and b, i.e., $Wrd(a,b) = Wrw(b,a) - Wrw(a,b)$.

3.3 Structural Features

Since course concepts are usually introduced based on their learning dependencies, the structure of MOOC courses also significantly contribute to prerequisite relation inference in MOOCs. However, structure-based features for prerequisite detection have not been well-studied in previous works. In this section, we investigate different structural information, including appearing positions of concepts, learning dependencies of videos and complexity levels of concepts, to propose three novel features to infer prerequisite relations in MOOCs. Before introducing these features, let us define two useful notations as follows. $\mathcal{C}(a)$ are the courses in which a is a course concept, i.e., $\mathcal{C}(a) = \{\mathcal{C}_i | \mathcal{C}_i \in \mathcal{D}, a \in \mathcal{K}_i\}$. $\mathcal{I}(\mathcal{C}, a)$ are the video indexes that contain concept a in course $\mathcal{C}$. For example, if a appears in the first and the 4-th video of $\mathcal{C}$, then $\mathcal{I}(\mathcal{C}, a) = \{1, 4\}$.

Feature 5: *Average Position Distance*

In a course, for a specific concept, its prerequisite concepts tend to be introduced before this concept and its subsequent concepts tend to be introduced after this concept. Based on this observation, for a concept pair $\langle a, b\rangle$, we calculate the distance of the average appearing position of a and b as one feature, namely **average position distance** (Apd). If $\mathcal{C}(a) \cap \mathcal{C}(b) \neq \emptyset$, $Apd(a,b)$ is formally defined as follows.

$$Apd(a,b) = \frac{\sum_{\mathcal{C} \in \mathcal{C}(a) \cap \mathcal{C}(b)} \left| \frac{\sum_{i \in \mathcal{I}(\mathcal{C},a)} i}{|\mathcal{I}(\mathcal{C},a)|} - \frac{\sum_{j \in \mathcal{I}(\mathcal{C},b)} j}{|\mathcal{I}(\mathcal{C},b)|} \right|}{|\mathcal{C}(a) \cap \mathcal{C}(b)|} \tag{12}$$

If $\mathcal{C}(a) \cap \mathcal{C}(b) = \emptyset$, we set $Apd(a,b) = 0$.

Feature 6: *Distributional Asymmetry Distance*

We also use the learning dependency of course videos to help infer learning dependency of course concepts. Based on our observation, the chance that a prerequisite concept is frequently mentioned in its subsequent videos is larger than that a subsequent concept is talked about in its prerequisite videos. Specifically, if video $\mathcal{V}_a$ is a precursor video of $\mathcal{V}_b$, and a is a prerequisite concept of b, then it is likely that $f(b, \mathcal{V}_a) < f(a, \mathcal{V}_b)$, where $f(a, \mathcal{V})$ denotes the term frequency of a in video $\mathcal{V}$. We thus define another feature, namely **distributional asymmetry distance** (Dad), to calculate the extent that a given concept pair satisfies this distributional asymmetry pattern. Formally, in course $\mathcal{C}$, for a given concept pair $\langle a, b\rangle$, we first define $\mathcal{S}(\mathcal{C}) = \{(i,j) | i \in \mathcal{I}(\mathcal{C}, a), j \in \mathcal{I}(\mathcal{C}, b), i < j\}$, i.e., all possible video pairs of $\langle a, b\rangle$ that have sequential relation. Then, the distributional asymmetry distance of $\langle a, b\rangle$ is formally defined as follows.

$$Dad(a,b) = \frac{\sum_{\mathcal{C} \in \mathcal{C}(a) \cap \mathcal{C}(b)} \frac{\sum_{(i,j) \in \mathcal{S}(\mathcal{C})} f(a, \mathcal{V}_i^{\mathcal{C}}) - f(b, \mathcal{V}_j^{\mathcal{C}})}{|\mathcal{S}(\mathcal{C})|}}{|\mathcal{C}(a) \cap \mathcal{C}(b)|} \tag{13}$$

where $\mathcal{V}_i^{\mathcal{C}}$ denotes the i-th video of course $\mathcal{C}$. If $\mathcal{C}(a) \cap \mathcal{C}(b) = \emptyset$, we set $Dad(a,b) = 0$.

Feature 7: *Complexity Level Distance*

Two related concepts with prerequisite relationship tend to have a difference in their complexity level, meaning that one concept is basic while another one is advanced. For example, "data set" and "training set" have learning dependencies and the latter concept is more advanced than the former one. However, "test set" and "training set" have no such relation when their complexity levels are similar. Complexity level of a course concept is implicit in its distribution in courses. Specifically, we observe that, for a concept in MOOCs, if it covers more videos in a course or it survives longer time in a course, then it is more likely to be a basic concept rather than an advanced one. We then formally define the **average video coverage** (avc) and the **average survival time** (ast) of a concept a as follows.

$$avc(a) = \frac{1}{|\mathcal{C}(a)|} \sum_{\mathcal{C} \in \mathcal{C}(a)} \frac{|\mathcal{I}(\mathcal{C}, a)|}{|\mathcal{C}|} \tag{14}$$

$$ast(a) = \frac{1}{|\mathcal{C}(a)|} \sum_{\mathcal{C} \in \mathcal{C}(a)} \frac{max(\mathcal{I}(\mathcal{C}, a)) - min(\mathcal{I}(\mathcal{C}, a)) + 1}{|\mathcal{C}|} \tag{15}$$

where $max/min(\mathcal{I}(\mathcal{C}, a))$ obtains the video index where a appears the last/first time in course $\mathcal{C}$. Based on the above equations, we define the **complexity level distance** (Cld) between concept a and b as follows.

$$Cld(a,b) = avc(a) \cdot ast(a) - avc(b) \cdot ast(b) \tag{16}$$

4 Experiments

4.1 Data Sets

In order to validate the efficiency of our features, we conducted experiments on three MOOC corpus with different domains: "Machine Learning" **(ML)**, "Data Structure and Algorithms" **(DSA)**, and "Calculus" **(CAL)**. To the best of our knowledge, there is no public data set for mining

Dataset	#courses	#videos	#concepts	#pairs		κ
				$-$	$+$	
ML	5	548	244	5,676	1,735	0.63
DSA	8	449	201	3,877	1,148	0.65
CAL	7	359	128	1,411	621	0.59

Table 1: Dataset Statistics

prerequisite relations in MOOCs. We created the experimental data sets through a three-stage process.

First, for each chosen domain, we select its relevant courses from Coursera, one of the leading MOOC platforms, and download all course materials using *coursera-dl* [2], a widely-used tool for automatically downloading Coursera.org videos. For example, for ML, we select 5 related courses [3] from 5 different universities and obtain a total of 548 course videos. Then, we manually label course concepts for each course: (1) Extract candidate concepts from documents of video subtitles following the method of Parameswaran et al. (2010). (2) Label the candidates as "course concept" or "not course concept" and obtain a set of course concepts for this course.

Finally, we manually annotate the prerequisite relations among the labeled course concepts. If the number of course concepts is n, the number of all possible pairs to be checked could reach $n \times (n-1)/2$, which requires arduous human labeling work. Therefore, for each dataset, we randomly select 25 percent of all possible pairs for evaluation. For each course concept pair $\langle a, b \rangle$, three human annotators majoring in the corresponding domain were asked to label them as "a is b's prerequisite", "b is a's prerequisite" or "no prerequisite relationship" using their own knowledge background and additional textbook resources. We take a majority vote of the annotators to create final labels and access the inter-annotator agreement using the average of pairwise κ statistics (Landis and Koch, 1981) between all pairs of the three annotators.

The statistics of the three datasets are listed in Table 1, where *#courses* and *#videos* are the total number of courses and videos in each dataset and *#concepts* is the number of labeled course concepts. The *#pairs* denotes the number of labeled concept pairs for evaluation, in which '$+$' denotes the number of positive instances, i.e. pairs who have prerequisite relations, and '$-$' denotes the number of negative instances.

[2]https://github.com/coursera-dl/coursera-dl

[3]These courses are: "Machine Learning (Stanford)", "Machine Learning (Washington)", "Practical Machine Learning (JHU)", "Machine Learning With Big Data (UCSD)" and "Neural Networks for Machine Learning (UofT)"

Classifier		ML		DSA		CAL	
	M	1	10	1	10	1	10
SVM	P	63.2	60.1	60.7	62.3	61.1	61.9
	R	68.5	72.4	**69.3**	67.5	**67.9**	68.3
	F_1	65.8	65.7	64.7	64.8	64.3	64.9
NB	P	58.0	58.2	62.9	62.6	60.1	60.6
	R	58.1	60.5	62.3	61.8	61.2	62.1
	F_1	58.1	59.4	62.6	62.2	60.6	61.3
LR	P	66.8	67.6	63.1	62.0	62.7	63.3
	R	60.8	61.0	64.8	66.8	63.6	64.1
	F_1	63.7	64.2	63.9	64.3	61.6	62.9
RF	P	**68.1**	**71.4**	**69.1**	**72.7**	**67.3**	**70.3**
	R	**70.0**	**73.8**	68.4	**72.3**	67.8	**71.9**
	F_1	**69.1**	**72.6**	**68.7**	**72.5**	**67.5**	**71.1**

Table 2: Classification results of the proposed method(%).

4.2 Evaluation Results

For each dataset, we apply 5-fold cross validation to evaluate the performance of the proposed method, i.e., testing our method on one fold while training the classifier using the other 4 folds. Usually, there are much fewer positive instances than negative instances, so we balance the training set by oversampling the positive instances (Yosef et al., 2011; Talukdar and Cohen, 2012). In our experiments, we employ 4 different binary classifiers, including NaïveBayes (*NB*), Logistic Regression (*LR*), SVM with linear kernel (*SVM*) and Random Forest (*RF*). We use precision (P), recall (R), and F1-score (F_1) to evaluate the prerequisite classification results. The experimental results are presented in Table 2.

Contextual features are shaped by the parameter M, i.e., the number of related concepts being considered. In our experiments, we tried different settings of M and report the results when M=1 and M=10 in Table 2. As for the semantic representation, we use the latest publicly available Wikipedia dump [4] and apply the skip-gram model (Mikolov et al., 2013b) to train word embeddings using the Python library gensim [5] with default parameters.

As shown in Table 2, the evaluation results varies by different classifiers. It turns out that NaïveBayes performs the worst. This seems to be caused by the fact that the independence assumption is not satisfied for our features; for

[4]https://dumps.wikimedia.org/enwiki/20170120/

[5]http://radimrehurek.com/gensim/

example, Feature 2 and Feature 3 both utilize the local context information, only with different granularity, thus are quite co-related. Random Forest beats others, with best F_1 across all three datasets. Its average F_1 outperforms SVM, NB and LR by 7.0%, 11.1% and 8.3%, respectively (M=10). The reason is as follows. Instead of a simple descriptive feature, each of our proposed feature determines whether a concept pair has prerequisite relation from a specific aspect; its function is similar to an independent weak classifier. Therefore, rather than using a linear combination of features for classification (e.g., SVM and LR), a boosting model (e.g., Random Forest) is more suitable for this task. The performance is slightly better when M=10 for all classifiers, with +0.20% for SVM, +0.53% for NB, +0.73% for LR and +3.63% for RF, with respect to the average F_1. The results verify the effectiveness of considering related concepts in contextual features. We use RF and set M=10 in the following experiments.

4.3 Comparison with Baselines

We further compare our approach with three representative methods for prerequisite inference.

4.3.1 Baseline Approaches

Hyponym Pattern Method (HPM). Prerequisite relationships often exists between hyponym-hypernym concept pairs (e.g., "Machine Learning" and "Supervised Learning"). As a baseline, we adopt the 10 lexico-syntactic patterns used by Wang et al. (2016) to extract hyponym relationships between concepts. If a concept pair matches at least one of these patterns in the MOOC corpus, we judge them to have prerequisite relations.

Reference Distance (RD) We also employ the RefD proposed by Liang et al. (2015) as one of our baselines. However, this method is only appliable to Wikipedia concepts. To make it comparable with our method, for each of our datasets, we construct a subset of it by picking out the concept pairs $\langle a, b\rangle$ in which a and b are both Wikipedia concepts. For example, we find 49% of course concepts in ML have their corresponding Wikipedia articles and 28% percent of concept pairs in ML meet the above condition. We use the new datasets constructed from ML, DSA and CAL, namely **W-ML**, **W-DSA**, and **W-CAL**, to compare our method with RefD.

Supervised Relationship Identification (SRI) Wang et al. (2016) has employed several features to infer prerequisite relations of Wikipedia concepts in textbooks, including 3 Textbook features and 6 Wikipedia features. Based on these features, they performed a binary classification using SVM to identify prerequisite relationships and has achieved state-of-the-art results. Because the Wikipedia features can only be applied to Wikipedia concepts, in order to make a comparison, we create two versions of their method: (1) **T-SRI**: only textbook features are used to train the classifier and (2) **F-SRI**: the original version, all features are used. We compare the performance of our method with T-SRI on ML, DSA and CAL datasets; we also compare our method with F-SRI on W-ML, W-DSA and W-CAL datasets.

Method		ML	DSA	CAL	W-ML	W-DSA	W-CAL
HPM	P	67.3	71.4	69.5	**79.9**	72.3	73.5
	R	18.4	14.8	16.5	25.5	27.3	23.3
	F_1	29.0	24.5	26.7	38.6	39.6	35.4
RD	P	–	–	–	73.4	**77.8**	**74.4**
	R	–	–	–	42.8	44.8	43.1
	F_1	–	–	–	54.1	56.8	54.6
T-SRI	P	61.4	62.3	62.5	58.1	60.1	62.7
	R	62.9	64.6	65.5	67.6	65.3	67.9
	F_1	62.1	63.4	64.0	62.5	62.6	65.2
F-SRI	P	–	–	–	64.3	64.3	64.8
	R	–	–	–	62.1	65.6	65.2
	F_1	–	–	–	63.2	64.9	65.0
MOOC	P	**71.4**	**72.7**	**70.3**	72.8	68.4	71.4
	R	**73.8**	**72.3**	**71.9**	**71.3**	**72.0**	**70.8**
	F_1	**72.6**	**72.5**	**71.1**	**72.0**	**70.2**	**71.1**

Table 3: Comparison with baselines(%).

4.3.2 Performance Comparison

In Table 3 we summarize the comparing results of different methods across different datasets ("MOOC" refers to our method). We find that our method outperforms baseline methods across all six datasets [6]. For example, the F_1 of our method on ML outperforms T-SRI and HPM by 10.5% and 43.6%, respectively. Specifically, we have the following observations. First, HPM achieves relatively high precision but low recall. This is because when A "is a" B, a prerequisite relation often exists from B to A, but clearly not vise versa. Second, T-SRI has certain effectiveness for learning prerequisite relations, with F_1 ranging from 62.1 to 65.2%. However, T-SRI only considers relatively simple features, such as the sequential and co-occurrence among concepts. With more

[6]The improvements are all statistically significant tested with bootstrap re-sampling with 95% confidence.

comprehensive feature engineering, the F_1 of our method significantly outperforms T-SRI (+10.5% on ML, +9.1% on DSA and +7.1% on CAL). Third, incorporating Wikipedia-based features (F-SRI) achieves certain promotion in performance (+0.93% comparing with T-SRI in average F_1).

4.4 Feature Contribution Analysis

In order to get an insight into the importance of each feature in our method, we perform a contribution analysis with different features. Here, we run our approach 10 times on the ML dataset. In each of the first 7 times, one feature is removed; in each of the rest 3 times, one group of features are removed, e.g., removing contextual features means removing $Gvrd$, $Gsrd$ and Wrd at the same time. We record the decrease of F1-score for each setting. Table 4 lists the evaluation results after ignoring different features.

According to the decrement of F1-scores, we find that all the proposed features are useful in predicting prerequisite relations. Especially, we observe that Cld (Feature 7), decreasing our best F1-score by 7.4%, plays the most important role. This suggests that most concepts do exist difference in complexity level. For two concepts, the difference of their coverage and survival times in courses are important for prerequisite relation detection. On the contrary, with 1.9% decrease, Sr (Feature 1) is relatively less important. We may easily find two concepts which have related semantic meanings (e.g., "test set" and "training set") but have no prerequisite relationship. However, semantic relatedness is critical for the contextual features because it overcomes the problem of the sparsity of context in calculation. We experience a decrease of 5.4% when we further do not consider related concepts in contextual features, i.e., set M=1. As for the feature group contribution, we observe that Structural Features, with a decrease of 9.2%, has a greater impact than the other two groups. This is as expected because it includes Cld. Among the three structural features, Apd makes relatively less contribution. The reason is that sometimes the professor may frequently mention a prerequisite concept after introducing a subsequent concept orally, for helping students better understand the concept.

5 Related Works

To the best of our knowledge, there has been no previous work on mining prerequisite relations among concepts in MOOCs. Some researchers have been engaged in detecting other type of prerequisite relations. For example, Yang et al. (2015) proposed to induce prerequisite relations among courses to support curriculum planning. Liu et al. (2011) studied learning-dependency between knowledge units, a special text fragment containing concepts, using a classification-based method. In the area of education, researchers have tried to find general prerequisite structures from students' test performance (Vuong et al., 2011; Scheines et al., 2014; Huang et al., 2015). Different from them, we focus on more fine-grained prerequisite relations, i.e., the prerequisite relations among course concepts.

Among the few related works of mining prerequisite relations among concepts, Liang et al. (2015) and Talukdar and Cohen (Talukdar and Cohen, 2012) studied prerequisite relationships between Wikipedia articles. They assumed that hyperlinks between Wikipedia pages indicate a prerequisite relationship and design several useful features. Based on these Wikipedia features plus some textbook features, Wang et al. (Wang et al., 2016) proposed a method to construct a concept map from textbooks, which jointly learns the key concepts and their prerequisite relations. However, the investigation of only Wikipedia concepts is also the bottleneck of their studies. In our work, we propose more general features to infer prerequisite relations among concepts, regardless of whether the concept is in Wikipedia or not. Liang et al. (2017) propose an optimization based framework to discover concept prerequisite relations from course dependencies. Gordon et al. (2016) utilize cross-entropy to learn concept dependencies in scientific corpus. Besides local statistical information, our method also utilize external knowledge to enrich concept semantics, which is more informativeness.

	Ignored Feature(s)	P	R	F_1
Single	Sr	69.6	72.9	71.2(**-1.4**)
	GVrd	68.8	71.4	70.1(-2.5)
	GSrd	67.9	71.4	69.6(-3.0)
	Wrd	70.1	72.1	71.1(-1.5)
	Apd	69.7	70.8	70.2(-2.4)
	Dad	69.2	69.5	69.4(-3.2)
	Cld	64.9	65.6	65.2(**-7.4**)
Group	Semantic	69.6	72.9	71.2(**-1.4**)
	Contextual	66.4	68.9	67.6(-5.0)
	Structural	63.7	64.2	63.4(**-9.2**)

Table 4: Contribution analysis of different features(%).

Our work is also related to the study of automatic relation extraction. Different research lines have been proposed around this topic, including hypernym-hyponym relation extraction (Ritter et al., 2009; Wei et al., 2012), entity relation extraction (Zhou et al., 2006; Fan et al., 2014; Lin et al., 2015) and open relation extraction (Fader et al., 2011). However, previous works mainly focus on factual relations, the extraction of cognitive relations (e.g. prerequisite relations) has not been well studied yet.

6 Conclusions and Future Work

We conducted a new investigation on automatically inferring prerequisite relations among concepts in MOOCs. We precisely define the problem and propose several useful features from different aspects, i.e., contextual, structural and semantic features. Moreover, we apply an embedding-based method that jointly learns the semantic representations of Wikipedia concepts and MOOC concepts to help implement the features. Experimental results on online courses with different domains validate the effectiveness of the proposed method. Promising future directions would be to investigate how to utilize user interaction in MOOCs for better prerequisite learning, as well as how deep learning models can be used to automatically learn useful features to help infer prerequisite relations.

Acknowledgments

This work is supported by 973 Program (No. 2014CB340504), NSFC Key Program (No. 61533018), Fund of Online Education Research Center, Ministry of Education (No. 2016ZD102), Key Technologies Research and Development Program of China (No. 2014BAK04B03) and NSFC-NRF (No. 61661146007).

References

Benjamin Samuel Bloom. 1981. *All our children learning: A primer for parents, teachers, and other educators*. McGraw-Hill Companies.

Yixin Cao, Juanzi Li, Xiaofei Guo, Shuanhu Bai, Heng Ji, and Jie Tang. 2015. Name list only? target entity disambiguation in short texts. In *Proceedings of EMNLP*. pages 654–664.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of EMNLP*. pages 1535–1545.

Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng, and Edward Y. Chang. 2014. Distant supervision for relation extraction with matrix completion. In *Proceedings of ACL*. pages 839–849.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of CIKM*. pages 1625–1628.

Jonathan Gordon, Linhong Zhu, Aram Galstyan, Prem Natarajan, and Gully Burns. 2016. Modeling concept dependencies in a scientific corpus. In *Proceedings of ACL*.

Xiaopeng Huang, Kyeong Yang, and Victor B. Lawrence. 2015. An efficient data mining approach to concept map generation for adaptive learning. In *Proceedings of ICDM*. pages 247–260.

James Gregory Jardine. 2014. *Automatically generating reading lists*. Ph.D. thesis, University of Cambridge, UK.

RJ Landis and GG Koch. 1981. The measurement of interrater agreement. *Statistics methods for rates and proportions* 2:212–236.

Stephen Laurence and Eric Margolis. 1999. Concepts and cognitive science. *Concepts: core readings* pages 3–81.

Chen Liang, Zhaohui Wu, Wenyi Huang, and C. Lee Giles. 2015. Measuring prerequisite relations among concepts. In *Proceedings of EMNLP*. pages 1668–1674.

Chen Liang, Jianbo Ye, Zhaohui Wu, Bart Pursel, and C. Lee Giles. 2017. Recovering concept prerequisite relations from university course dependencies. In *Proceedings of AAAI*. pages 4786–4791.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*. pages 2181–2187.

Jun Liu, Lu Jiang, Zhaohui Wu, Qinghua Zheng, and Ya-nan Qian. 2011. Mining learning-dependency between knowledge units from text. *The VLDB Journal* 20(3):335–345.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *International Journal of CoRR* abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*. pages 3111–3119.

Joseph D. Novak. 1990. Concept mapping: A useful tool for science education. *International Journal of Research in Science Teaching* 27(10):937C949.

Chitu Okoli, Mohamad Mehdi, Mostafa Mesgari, Finn Årup Nielsen, and Arto Lanamäki. 2014. Wikipedia in the eyes of its beholders: A systematic review of scholarly research on wikipedia readers and readership. *International Journal of the American Society for Information Science and Technology (JASIST)* 65(12):2381–2403.

Aditya G. Parameswaran, Hector Garcia-Molina, and Anand Rajaraman. 2010. Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment (PVLDB)* 3(1):566–577.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Proceedings of AAAI*. pages 88–93.

Jean Michel Rouly, Huzefa Rangwala, and Aditya Johri. 2015. What are we teaching?: Automated evaluation of CS curricula content using topic modeling. In *Proceedings of ICER*. pages 189–197.

Richard Scheines, Elizabeth Silver, and Ilya M. Goldin. 2014. Discovering prerequisite relationships among knowledge components. In *Proceedings of EDM*. pages 355–356.

Nick J Schweitzer. 2008. Wikipedia and psychology: Coverage of concepts and its use by undergraduate students. *International Journal of Teaching of Psychology* 35(2):81–85.

Partha Pratim Talukdar and William W Cohen. 2012. Crowdsourced comprehension: predicting prerequisite structure in wikipedia. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. pages 307–315.

Annalies Vuong, Tristan Nixon, and Brendon Towle. 2011. A method for finding prerequisites within a curriculum. In *Proceedings of EDM*. pages 211–216.

Shuting Wang, Alexander Ororbia, Zhaohui Wu, Kyle Williams, Chen Liang, Bart Pursel, and C Lee Giles. 2016. Using prerequisites to extract concept maps fromtextbooks. In *Proceedings of CIKM*. pages 317–326.

Bifan Wei, Jun Liu, Jian Ma, Qinghua Zheng, Wei Zhang, and Boqin Feng. 2012. MOTIF-RE: motif-based hypernym/hyponym relation extraction from wikipedia links. In *Proceedings of ICONIP*. pages 610–619.

Yiming Yang, Hanxiao Liu, Jaime G. Carbonell, and Wanli Ma. 2015. Concept graph learning from educational data. In *Proceedings of WSDM*. pages 159–168.

Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. 2011. AIDA: an online tool for accurate disambiguation of named entities in text and tables. *Proceedings of the VLDB Endowment (PVLDB)* 4(12):1450–1453.

Guodong Zhou, Jian Su, and Min Zhang. 2006. Modeling commonality among related classes in relation extraction. In *Proceedings of ACL*.

Unsupervised Text Segmentation Based on Native Language Characteristics

Shervin Malmasi[1,2] **Mark Dras**[2] **Mark Johnson**[2] **Lan Du**[3] **Magdalena Wolska**[4]
[1]Harvard Medical School, Harvard University
smalmasi@bwh.harvard.edu
[2]Department of Computing, Macquarie University
{ shervin.malmasi, mark.dras, mark.johnson }@mq.edu.au
[3]Faculty of IT, Monash University
lan.du@monash.edu
[4]LEAD Graduate School, Universität Tübingen
magdalena.wolska@uni-tuebingen.de

Abstract

Most work on segmenting text does so on the basis of topic changes, but it can be of interest to segment by other, stylistically expressed characteristics such as change of authorship or native language. We propose a Bayesian unsupervised text segmentation approach to the latter. While baseline models achieve essentially random segmentation on our task, indicating its difficulty, a Bayesian model that incorporates appropriately compact language models and alternating asymmetric priors can achieve scores on the standard metrics around halfway to perfect segmentation.

1 Introduction

Most work on automatically segmenting text has been on the basis of topic: segment boundaries correspond to topic changes (Hearst, 1997). There are various contexts, however, in which it is of interest to identify changes in other characteristics; for example, there has been work on identifying changes in authorship (Koppel et al., 2011) and poetic voice (Brooke et al., 2012). In this paper we investigate text segmentation on the basis of change in the native language of the writer.

Two illustrative contexts where this task might be of interest are patchwriting detection and literary analysis. Patchwriting is the heavy use of text from a different source with some modification and insertion of additional words and sentences to form a new text. Pecorari (2003) notes that this is a kind of textual plagiarism, but is a strategy for learning to write in an appropriate language and style, rather than for deception. Keck (2006), Gilmore et al. (2010) and Vieyra et al. (2013) found that non-native speakers, not surprisingly in situations of imperfect mastery of a language, are strongly over-represented in this kind of textual plagiarism. In these cases the boundaries between the writer's original text and (near-)copied native text are often quite apparent to the reader, as in this short example from Li and Casanave (2012) (copied text italicised): "Nevertheless, doubtfulness can be cleared reasonably by the experiments conducted upon the 'split-brain patients', *in whom intra-hemispheric communication is no longer possible*. To illustrate, *one experiment has the patient sit at a table with a* non-transparent *screen blocking the objects behind*, who is then asked to *reach* the objects with different hand respectively." Because patchwriting can indicate imperfect comprehension of the source (Jamieson and Howard, 2013), identifying it and supporting novice writers to improve it has become a focus of programmes like the Citation Project.[1]

For the second, perhaps more speculative context of literary analysis, consider Joseph Conrad, known for having written a number of famous English-language novels, such as *Heart of Darkness*; he was born in Poland and moved to England at the age of 21. His writings have been the subject of much manual analysis, with one particular direction of such research being the identification of likely influences on his English writing, including his native Polish language and the French he learnt before English. Morzinski (1994), for instance, notes aspects of his writing that exhibit Polish-like syntax, verb inflection, or other linguistic characteristics (*e.g.* "Several had still their staves in their hands" where the awkwardly placed adverb *still* is typical of Polish). These appear both in isolated sentences and in larger chunks of text, and part of

[1]http://citationproject.net/

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1457–1469
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1134

an analysis can involve identifying these chunks.

This raises the question: Can NLP and computational models identify the points in a text where native language changes? Treating this as an unsupervised text segmentation problem, we present the first Bayesian model of text segmentation based on authorial characteristics, applied to native language.

2 Related Work

Topic Segmentation The most widely-researched text segmentation task has as its goal to divide a text into topically coherent segments. *Lexical cohesion* (Halliday and Hasan, 1976) is an important concept here: the principle that text is not formed by a random set of words and sentences but rather logically ordered sets of related words that together form a topic. In addition to the semantic relation between words, other methods such as back-references and conjunctions also help achieve cohesion. Based on this, Morris and Hirst (1991) proposed the use of *lexical chains*, sequences of related words (defined via thesaurus), to break up a text into topical segments: breaks in lexical chains indicate breaks in topic. The *TextTiling* algorithm (Hearst, 1994, 1997) took a related approach, defining a function over lexical frequency and distribution information to determine topic boundaries, and assuming that each topic has its own vocabulary and that large shifts in this vocabulary usage correspond to topic shifts.

There have been many approaches since that time. A key one, which is the basis for our own work, is the unsupervised Bayesian technique BAYESSEG of Eisenstein and Barzilay (2008), based on a generative model that assumes that each segment has its own language model. Under this assumption the task can be framed as predicting boundaries at points which maximize the probability of a text being generated by a given language model. Their method is based on lexical cohesion — expressed in this context as topic segments having compact and consistent lexical distributions — and implements this within a probabilistic framework by modelling words within each segment as draws from a multinomial language model associated with that segment.

Much other subsequent work either uses this as a baseline, or extends it in some way: Jeong and Titov (2010), for example, who propose a model for joint discourse segmentation and alignment for documents with parallel structures, such as a text with commentaries or presenting alternative views on the same topic; or Du et al. (2013), who use hierarchical topic structure to improve the linear segmentation.

Bible Authorship Koppel et al. (2011) consider the task of decomposing a document into its authorial components based on their stylistic properties and propose an unsupervised method for doing so. The authors use as their data two biblical books, Jeremiah and Ezekiel, that are generally believed to be single-authored: their task was to segment a single artificial text constructed by interleaving chapters of the two books. Their most successful method used work in biblical scholarship on lexical choice: they give as an example the case that in Hebrew there are seven synonyms for the word *fear*, and that different authors may choose consistently from among them. Then, having constructed their own synsets using available biblical resources and annotations, they represent texts by vectors of synonyms and apply a modified cosine similarity measure to compare and cluster these vectors. While the general task is relevant to this paper, the particular notion of synonymy here means the approach is specific to this problem, although their approach is extended to other kinds of text in Akiva and Koppel (2013). Aldebei et al. (2015) proposed a new approach motivated by this work, similarly clustering sentences, then using a Naive Bayes classifier with modified prior probabilities to classify sentences.

Poetry Voice Detection Brooke et al. (2012) perform stylistic segmentation of a well-known poem, *The Waste Land* by T.S. Eliot. This poem is renowned for the great number of voices that appear throughout the text and has been the subject of much literary analysis (Bedient and Eliot, 1986; Cooper, 1987). These distinct voices, conceived of as representing different characters, have differing tones, lexis and grammatical styles (*e.g.* reflecting the level of formality). The transitions between the voices are not explicitly marked in the poem and the task here is to predict the breaks where these voice changes occur. The authors argue that the use of generative models is not feasible for this task, noting: "Generative models, which use a bag-of-words assumption, have a very different problem: in their standard form, they can capture only lexical cohesion, which is not the (primary) focus of stylistic analysis."

They instead present a method based on a curve that captures stylistic change, similar to the TextTiling approach but generalised to use a range of features. The local maxima in this change curve represent potential breaks in the text. The features are both internal to the poem (*e.g.* word length, syllable count, POS tag) as well as external (*e.g.* average unigram counts in the 1T Corpus or sentiment polarity from a lexicon). Results on an artificially constructed mixed-style poem achieve a P_k of 0.25. Brooke et al. (2013) extend this by considering clustering following an initial segmentation.

Native Language Identification (NLI) NLI casts the detecting of native language (L1) influence in writing in a non-native (L2) language as a classification task: the framing of the task in this way comes from Koppel et al. (2005). There has been much activity on it in the last few years, with Tetreault et al. (2012) providing a comprehensive analysis of features that had been used up until that point, and a shared task in 2013 (Tetreault et al., 2013) that attracted 29 entrants. The shared task introduced a new, now-standard dataset, TOEFL11, and work has continued on improving classification results, *e.g.* by Bykh and Meurers (2014) and Ionescu et al. (2014).

In addition to work on the classification task itself, there have also been investigations of the features used, and how they might be employed elsewhere. Malmasi and Cahill (2015) examine the effectiveness of individual feature types used in the shared task and the diversity of those features. Of relevance to the present paper, simple part-of-speech n-grams alone are fairly effective, with classification accuracies of between about 40% and 65%; higher-order n-grams are more effective than lower, and the more fine-grained CLAWS2 tagset more effective than the Penn Treebank tagset. An area for application of these features is in Second Language Acquisition (SLA), as a data-driven approach to finding L1-related characteristics that might be a result of cross-linguistic influence and consequently a possible starting for an SLA hypothesis (Ellis, 2008); Swanson and Charniak (2013) and Malmasi and Dras (2014) propose methods for this.

Tying It Together Contra Brooke et al. (2012), we show that it is possible to develop effective generative models for segmentation on stylistic factors, of the sort used for topic segmentation. To apply it specifically to segmentation based on a writer's L1, we draw on work in NLI.

3 Experimental Setup

We investigate the task of L1-based segmentation in three stages:

1. Can we define any models that do better than random, in a best case scenario? For this best case scenario, we determine results over a devset with the best prior found by a grid search, for a single language pair likely to be relatively easily distinguishable. Note that as this is *unsupervised* segmentation, it is a devset in the sense that it is used to find the best prior, and also in a sense that some models as described in §4 use information from a related NLI task on the underlying data.

2. If the above is true, do the results also hold for test data, using priors derived from the devset?

3. Further, do the results also hold for all language pairs available in our dataset, not just a single easily distinguishable pair?

We first describe the evaluation data — artificial texts generated from learner essays, similar to the artificially constructed texts of previously described work on Bible authorship and poetry segmentation — and evaluation metrics, followed in §4 by the definitions of our Bayesian models.

3.1 Source Data

We use as the source of data the TOEFL11 dataset used for the NLI shared task (Blanchard et al., 2013) noted in §2. The data consists of 12100 essays by writers with 11 different L1s, taken from TOEFL tests where the test-taker is given a *prompt*[2] as the topic for the essay. The corpus is balanced across L1s and prompts (which allows us to verify that segmentation isn't occurring on the basis of topic), and is split into standard training, dev and test sets.

3.2 Document Generation

As the main task is to segment texts by the author's L1, we want to ensure that we are not segmenting by topic and thus use texts written by authors from different L1 backgrounds on the same topic (prompt). We will also create one dataset to verify that segmentation by topic works in this domain; for this we use texts written by authors from the same L1 background on different topics.

For our L1-varying datasets, we construct composite documents to be segmented as alternat-

[2]For example, prompt P7 is: "Do you agree or disagree with the following statement? It is more important for students to understand ideas and concepts than it is for them to learn facts. Use reasons and examples to support your answer."

ing segments drawn from TOEFL11 from two different L1s holding the topic (prompt) constant, broadly following a standard approach (Brooke et al., 2012, for example) (see Appendix A.1 for details). We follow the same process for our topic-varying datasets, but hold the L1 constant while alternating the topic (prompt). For our single pair of L1s, we choose German and Italian. German is the class with the highest NLI accuracy in the TOEFL11 corpus across the shared task results and Italian also performs very well. Additionally, there is very little confusion between the two; a binary NLI classifier we trained on the language pair achieved 97% accuracy. For our all-pairs results, given the 11 languages in the TOEFL11 corpus, we have 55 sets of documents of alternating L1s (one of which is German–Italian).

We generate four distinct types of datasets for our experiments using the above methodology. The documents in these datasets, as described below, differ in the parameters used to select the essays for each segment and what type of tokens are used. Tokens (words) can be represented in their original form and used for performing segmentation. Alternatively, using an insight from Wong et al. (2012), we can represent the documents at a level other than lexical: the text could consist of the POS tags corresponding to all of the tokens, or n-grams over those POS tags. The POS representation is motivated by the usefulness of POS-based features for capturing L1-based stylistic differences as noted in §2. Our method for encoding n-grams is described in Appendix A.2.

TOPICSEG-TOKENS This data is generated by keeping the L1 class constant and alternating segments between two topics. We chose Italian for the L1 class and essays from the prompts "P7" and "P8" are used. The dataset, constructed from TOEFL11-TRAIN and TOEFL11-DEV, contains a total of 53 artificial documents, and will be used to verify that topic segmentation as discussed in Eisenstein and Barzilay (2008) functions as expected for data from this domain: that is, that topic change is detectable.

TOPICSEG-PTB Here the tokens in each text are replaced with their POS tags or n-grams over those tags, and the segmentation is performed over this data. In this dataset the tags are obtained via the Stanford Tagger and use the Penn Treebank (PTB) tagset. The same source data (TOEFL11-TRAIN and TOEFL11-DEV), L1 and topics as TOPICSEG-TOKENS are used for a total of 53 documents. This dataset will be used to investigate, *inter alia*, whether segmentation over these stylistically related features could take advantage of topic cues. We would expect not.

L1SEG-PTB This dataset is used for segmentation based on native language, also using (n-grams over) the PTB POS tags. We choose a specific topic and then retrieve all essays from the corpus that match this; here we chose prompt "P7", since it had the largest number of essays for our chosen single L1 pair, German–Italian. For the dataset constructed from TOEFL11-TRAIN and TOEFL11-DEV (which we will refer to as L1SEG-PTB-GI-DEV), this resulted in 57 documents. Documents that are composites of two L1s are then generated as described above. For investigating questions 2 and 3 above, we similarly have datasets constructed from the the smaller TOEFL11-TEST data (L1SEG-PTB-GI-TEST), which consist of 5 documents of 5 segments each for the single L1 pair, and from all language pairs (L1SEG-PTB-ALL-DEV, L1SEG-PTB-ALL-TEST). We would expect that these datasets should not be segmentable by topic, as all the segments are on the same topic; the segments should however, differ in stylistic characteristics related to the L1.

L1SEG-CLAWS2 This dataset is generated using the same methodology as L1SEG-PTB, with the exception that the essays are tagged using the RASP tagger which uses the more fine-grained CLAWS2 tagset, noting that the CLAWS2 tagset performed better in the NLI classification task (Malmasi and Cahill, 2015).

3.3 Evaluation

We use the standard P_k (Beeferman et al., 1999) and WindowDiff (WD) (Pevzner and Hearst, 2002) metrics, which (broadly speaking) select sentences using a moving window of size k and determines whether these sentences correctly or incorrectly fall into the same or different reference segmentations. P_k and WD scores range between 0 and 1, with a lower score indicating better performance, and 0 a perfect segmentation. It has been noted that some "degenerate" algorithms — such as placing boundaries randomly or at every possible position — can score 0.5 (Pevzner and Hearst, 2002). WD scores are typically similar to P_k, correcting for differential penalties between false positive boundaries and false negatives implicit in P_k. P_k and WD scores reported in §5 are

averages across all documents in a dataset. Formal definitions are given in Appendix A.3.

4 Segmentation Models

For all of our segmentation we use as a starting point the unsupervised Bayesian method of Eisenstein and Barzilay (2008); see §2.[3] We recap the important technical definitions here.

In Equation 1 of their work they define the observation likelihood as,

$$p(\mathbf{X} \mid \mathbf{z}, \Theta) = \prod_{t}^{T} p(\mathbf{x}_t \mid \theta_{z_t}), \quad (1)$$

where $\mathbf{X}$ is the set of all T sentences, $\mathbf{z}$ is the vector of segment assignments for each sentence, $\mathbf{x}_t$ is the bag of words drawn from the language model and Θ is the set of all K language models $\Theta_1 \ldots \Theta_K$. As is standard in segmentation work, K is assumed to be fixed and known (Malioutov and Barzilay, 2006); it is set to the actual number of segments. The authors also impose an additional constraint, that z_t must be equal to either z_{t-1} (the previous sentence's segment) or $z_{t-1}+1$ (the next segment), in order to ensure a linear segmentation.

This segmentation model has two parameters: the set of language models Θ and the segment assignment indexes $\mathbf{z}$. The authors note that since this task is only concerned with the segment assignments, searching in the space of language models is not desirable. They offer two alternatives to overcome this: (1) taking point estimates of the language models, which is considered to be theoretically unsatisfying and (2) marginalizing them out, which yields better performance. Equation 7 of Eisenstein and Barzilay (2008), reproduced here, shows how they marginalize over the language models, supposing that each language model is drawn from a symmetric Dirichlet prior (*i.e.* $\theta_j \sim \mathrm{Dir}(\theta_0)$):

$$p(\mathbf{X} \mid \mathbf{z}, \theta_0) = \prod_{j}^{K} p_{dcm}(\{\mathbf{x}_t : z_t = j\} \mid \theta_0) \quad (2)$$

The Dirichlet compound multinomial distribution p_{dcm} expresses the expectation over all the multinomial language models, when conditioned on the symmetric Dirichlet prior θ_0:

$$p_{dcm}(\{\mathbf{x}_t : z_t = j\} \mid \theta_0) = \frac{\Gamma(W\theta_0)}{\Gamma(N_j + W\theta_o)} \prod_{i}^{W} \frac{\Gamma(n_{j,i} + \theta_0)}{\Gamma(\theta_0)} \quad (3)$$

where W is the number of words in the vocabulary and $N_j = \sum_i^W n_{j,i}$, the total number of words in the segment j. They then observe that the optimal segmentation maximizes the joint probability

$$p(\mathbf{X}, \mathbf{z} \mid \theta_0) = p(\mathbf{X} \mid \mathbf{z}, \theta_0) p(\mathbf{z})$$

and assume a uniform $p(\mathbf{z})$ over valid segmentations with no probability mass assigned to invalid segmentations. The hyperparameter θ_0 can be chosen, or can be learned via an Expectation-Maximization process.

Inference Eisenstein and Barzilay (2008) defined two methods of inference, a dynamic programming (DP) one and one using Metropolis-Hastings (MH). Only MH is applicable where shifting a boundary will affect the probability of every segment, not just adjacent segments, as in their model incorporating cue phrases. Where this is not the case, they use DP inference. Their DP inference algorithm is suitable for all of our models, so we also use that.

Priors For our priors, we carry out a grid search on the devsets (that is, the datasets derived from TOEFL11-TRAIN and TOEFL11-DEV) in the interval $[0.1, 3.0]$, partitioned into 30 evenly spaced values; this includes both weak and strong priors.[4]

4.1 TOPICSEG

Our first model is exactly the one proposed by Eisenstein and Barzilay (2008) described above. The aim here is to look at how we perform at segmenting learner essays by topic in order to confirm that topic segmentation works for this domain and these types of topics. We apply this model to the TOPICSEG-TOKENS and TOPICSEG-PTB datasets where the texts have the same L1 and boundaries are placed between essays of differing topics (prompts).

4.2 L1SEG

Our second model modifies that of Eisenstein and Barzilay (2008) by revising the generative story.

[3] An open-source implementation of the method, called BAYESSEG, is made available by the authors at `http://groups.csail.mit.edu/rbg/code/bayesseg/`

[4] The Eisenstein and Barzilay (2008) code does implement an EM method for finding priors in the symmetric case, but we found that perhaps surprisingly the grid search almost always found better ones.

Where they assume a standard generative model over words with constraints on topic change between sentences, we make minor modifications to adapt the model for our task. The standard generative story (Blei, 2012) — an account of how a model generates the observed data — usually generates words in a two-stage process: (1) For each document, randomly choose a distribution of topics. (2) For each word in the document: (a) Assign a topic from those chosen in step 1. (b) Randomly choose a word from that topic's vocabulary.

Here we modify this story to be over part-of-speech data instead of lexical items. By using this representation (which as noted in §2 is useful for NLI classification) we aim to segment our texts based on the L1 of the author for each segment. For this model we only make use of the L1SEG-PTB-GI-DEV dataset.[5]

4.3 L1SEG-COMP

It is not obvious that the same properties that produce compact distributions in standard lexical chains would also be the case for POS data, particularly if extended to POS n-grams which can result in a very large number of potential tokens. In this regard Eisenstein and Barzilay (2008) note: "To obtain a high likelihood, the language models associated with each segment should concentrate their probability mass on a compact subset of words. Language models that spread their probability mass over a broad set of words will induce a lower likelihood. This is consistent with the principle of lexical cohesion."

Eisenstein and Barzilay (2008) discuss this within the context of topic segmentation.[6] However, it is unclear if this would also would happen for POS tags; there is no syntactic analogue for the sort of lexical chains important in topic segmentation. It may then turn out that using all POS tags or n-grams over them as in the previous model would not achieve a strong performance. We thus use knowledge from the NLI classification task to help.

Discarding Non-Discriminative Features One approach that could possibly overcome these limitations is the removal of features from the input space that have been found to be non-discriminative in NLI classification. This would allow us to encode POS sequence information via n-grams while also keeping the model's vocabulary sufficiently small. Doing this requires the use of extrinsic information for filtering the n-grams. The use of such extrinsic information has proven to be useful for other similar tasks such as the poetry style change segmentation work of Brooke et al. (2012), as noted in §2.

We perform this filtering using the discriminative feature lists derived from the NLI classification task using the system and method described in Malmasi and Dras (2014), also noted in §2. We extract the top 300 most discriminative POS n-gram features for each L1 from TOEFL11-TRAIN and TOEFL11-DEV, resulting in two lists of 600 POS bigrams and trigrams; these are thus independent of our test datasets. (We illustrate a text with respect to these discriminative features in Appendix A.4.) Note that discriminative n-grams can overlap with each other within the same class and also between two classes. We resolve such conflicts by using the weights of the features from the classification task as described in Malmasi and Dras (2014) and choosing the feature with the higher weight.

4.4 L1SEG-ASYMP

Looking at the distribution of discriminative features in our documents, one idea is that incorporating knowledge about which features are associated with which L1 could potentially help improve the results. One approach to do this is the use of asymmetric priors. We note that features associated with an L1 often dominate in a segment. Accordingly, priors can represent evidence external to the data that some some aspect should be weighted more strongly: for us, this is evidence from the NLI classification task. The segmentation models discussed so far only make use of a symmetric prior but later work mentions that it would be possible to modify this to use an asymmetric prior (Eisenstein, 2009).

Given that priors are effective for incorporating external information, recent work has highlighted the importance of optimizing over such priors, and in particular, the use of asymmetric priors. Key work on this is by Wallach et al. (2009) on LDA, who report that "an asymmetric Dirichlet prior over the document-topic distributions has substantial advantages over a symmetric prior",

[5] We also looked at including words. The results of these models were always worse, and we do not discuss them in this paper.

[6] For example, a topic segment related to the previously mentioned essay prompt P7 might concentrate its probability mass on the following set of words: {`education`, `learning`, `understanding`, `fact`, `theory`, `idea`, `concept`, `knowledge`}.

with prior values being determined through hyperparameter optimization. Such methods have since been applied in other tasks such as sentiment analysis (Lin and He, 2009; Lin et al., 2012) to achieve substantial improvements. For sentiment analysis, Lin and He (2009) incorporate external information from a subjectivity lexicon. In applying LDA, instead of using a uniform Dirichlet prior for the document–sentiment distribution, they use asymmetric priors for positive and negative sentiment, determined empirically.

For our task, we assign a prior to each of two languages in a document, one corresponding to $\mathrm{L1}_a$ and the other to $\mathrm{L1}_b$. Given this, we can assume that segments will alternate between $\mathrm{L1}_a$ and $\mathrm{L1}_b$. And instead of a single θ_0, we have two asymmetric priors that we call θ_a, θ_b corresponding to $\mathrm{L1}_a$ and $\mathrm{L1}_b$ respectively. This will require reworking the definition of p_{dcm} in Equation 3. First adapting Equation 2,

$$p(\mathbf{X} \mid \mathbf{z}, \theta_a, \theta_b) = \prod_{\{j_o\}} p_{dcm}(\{\mathbf{x}_t : z_t = j_o\} \mid \theta_a) \cdot \prod_{\{j_e\}} p_{dcm}(\{\mathbf{x}_t : z_t = j_e\} \mid \theta_b), \quad (4)$$

with $\{j_o\} = \{j \mid j \mod 2 = 1, 1 \leq j \leq K\}$ the set of indices over odd segments and $\{j_e\} = \{j \mid j \mod 2 = 0, 1 \leq j \leq K\}$ the set over evens. K is the (usual) total number of segments. Then

$$p_{dcm}(\{\mathbf{x}_t : z_t = j_o\} \mid \theta_a) = \frac{\Gamma(\sum_k^W \theta_a[k])}{\Gamma(N_{j_o} + \sum_k^W \theta_a[k])} \prod_i^W \frac{\Gamma(n_{j,i} + \theta_a[i])}{\Gamma(\theta_a[i])} \quad (5)$$

W is now more generally the number of items in our vocabulary (whether words or POS n-grams). A notational addition here is $\theta_a[k]$ which refers to the $\mathrm{L1}_a$ prior for the kth word or POS n-gram. There is an analogous p_{dcm} for θ_b.

The next issue is how to construct the θ_a and θ_b. The simplest scenario would require a single constant value for all elements in one L1 and another for all elements in the other L1. Specifically, using $\mathrm{discrim}(\mathrm{L1}_x)$ to denote "the ranked list of discriminative n-grams for $\mathrm{L1}_x$", we define

$$\theta_a[i] = \begin{cases} c_1 & \text{if } \theta_a[i] \in \mathrm{discrim}(\mathrm{L1}_a) \\ c_2 & \text{if } \theta_a[i] \in \mathrm{discrim}(\mathrm{L1}_b) \end{cases}$$

and analogously for $\theta_b[i]$. We would expect that $c_1 > c_2$ (*i.e.* the prior is stronger for elements that come from the appropriate ranked list of discriminative features), but these values will be learned.

In principle we would calculate versions of $p(\mathbf{X} \mid \mathbf{z}, \theta_a, \theta_b)$ twice: once where we assign θ_a to segment 1, and the second time where we assign θ_b. We'd then compare the two $p(\mathbf{X} \mid \mathbf{z}, \theta_a, \theta_b)$, and see which one fits better. In this work, however, we will fix the initial L1: segment 1 corresponds to $\mathrm{L1}_a$ and consequently has prior θ_a.[7]

5 Results

5.1 Segmenting by Topic

We begin by testing the TOPICSEG model to ensure that the Bayesian segmentation methodology can achieve reasonable results for segmenting learner essays by topic. The results on the TOPICSEG-TOKENS dataset (Table 1) show that content words are very effective at segmenting the writings by topic, achieving $\mathrm{P_k}$ values in the range 0.19–0.21. These values are similar to those reported for segmenting *Wall Street Journal* text (Beeferman et al., 1999). On the other hand, using the PTB POS tag version of the data in the TOPICSEG-PTB dataset results in very poor segmentation results, with $\mathrm{P_k}$ values around 0.45. This is essentially the same as the performance of degenerate algorithms (noted in §3.3) of 0.5. This demonstrates that, as expected, POS unigrams do not provide enough information for topic segmentation; it is not possible to construct even an approximation to lexical chains using them.

5.2 L1-based Segmentation

Having verified that the Bayesian segmentation approach is effective for topic segmentation on this data, we now turn to the L1SEG model for segmenting by the native language.

From the results in Table 1 we see very poor performance with a $\mathrm{P_k}$ value of 0.466 for segmenting the texts in L1SEG-PTB-GI-DEV using the unigrams as is. This was a somewhat unexpected result given than we know POS unigram distributions are able to capture differences between L1-groups (Malmasi and Cahill, 2015), albeit with limited accuracy. Moreover, neither bigram nor trigram encodings, which perform better at NLI, resulted in any improvement in our results.

[7] This requires an extension of the BAYESSEG software to support asymmetric priors. We will make this extended version of the code available under the same conditions as BAYESSEG. Please contact the first or second author for this.

Model	Dataset	Prior(s)	P_k	WD
TopicSeg	TopicSeg-Tokens	0.1	0.203	0.205
TopicSeg	TopicSeg-PTB	0.8	0.444	0.480
L1Seg	L1Seg-PTB-GI-Dev unigrams	0.1	0.466	0.489
L1Seg	L1Seg-PTB-GI-Dev bigrams	0.8	0.466	0.487
L1Seg	L1Seg-PTB-GI-Dev trigrams	0.8	0.480	0.489
L1Seg-Comp	L1Seg-PTB-GI-Dev bigrams	0.1	0.476	0.490
L1Seg-Comp	L1Seg-PTB-GI-Dev trigrams	0.4	0.393	0.398
L1Seg-Comp	L1Seg-CLAWS2-GI-Dev bigrams	0.4	0.387	0.400
L1Seg-Comp	L1Seg-CLAWS2-GI-Dev trigrams	0.4	0.370	0.373
L1Seg-AsymP	L1Seg-CLAWS2-GI-Dev trigrams	(0.6,0.3)	**0.316**	**0.318**

Table 1: Results on devsets for single L1 pair (German–Italian).

Model	P_k	WD
L1Seg-Comp	0.358	0.360
L1Seg-AsymP	**0.266**	**0.271**

Table 2: Results on testset L1Seg-CLAWS2-GI-Test for single L1 pair (German–Italian). Priors are the ones from the corresponding devsets in Table 1.

Model	P_k	WD
L1Seg-Comp	0.365 (0.014)	0.369 (0.019)
L1Seg-AsymP	**0.299** (0.022)	**0.312** (0.027)
L1Seg-Comp	0.376 (0.032)	0.381 (0.033)
L1Seg-AsymP	**0.314** (0.043)	**0.319** (0.045)

Table 3: Results on dev and test datasets (upper: L1Seg-CLAWS2-All-Dev, lower: L1Seg-CLAWS2-All-Test): means and standard deviations (in parentheses) across datasets for all 55 L1 pairs.

5.3 Incorporating Discriminative Features

Filtering the bigrams results in some minor improvements over the best results from the L1Seg model. However, there are substantial improvements when using the filtered POS trigrams, with a P_k value of 0.393. We did not test unigrams as they were the weakest NLI feature of the three.

This improvement is, we believe, because the Bayesian modelling of lexical cohesion over the input tokens requires that each segment concentrates its probability mass on a compact subset of words. In the context of the n-gram tokenization method tested in the previous section, the L1Seg model with n-grams would most likely exacerbate the issue by substantially increasing the number of tokens in the language model: while the unigrams do not capture enough information to distinguish non-lexical shifts, the n-grams provide too many features.

We also see that using the CLAWS2 tagset outperforms the PTB tagset. The results achieved for bigrams are much higher, while the trigram results are also better, with $P_k = 0.370$. NLI experiments using different POS tagsets have established that more fine-grained tagsets (*i.e.* those with more tag categories) provide greater classification accuracy when used as n-gram features for classification.[8] Results here comport with the previous findings.

As one of the two best models, we run it on the held-out test data, using the best priors found from the grid search on the devset data (Table 2); we find the P_k and WD values are comparable (and in fact slightly better), so the model still works if the filtering uses discriminative NLI features from the devset. Looking at results across all 55 L1 pairs (Table 3), we also see similar mean P_k and WD values with only a small standard deviation, indicating the approach works just as well across all language pairs. Priors here are all also weak, in the range $[0.1, 0.9]$.

In sum, the results here demonstrate the importance of inducing a compact distribution, which we did here by reducing the vocabulary size by stripping non-informative features.

5.4 Applying Two Asymmetric Priors

Our final model, L1Seg-AsymP, assesses whether setting different priors for each L1 can improve performance. Our grid search over two priors gives 900 possible prior combinations. These combinations also include cases where θ_a and θ_b are symmetric, which is equivalent to the L1Seg-Comp model. We observe (Table 1) that

[8] In §2 we noted the comparison of PTB and CLAWS2 tagsets in Malmasi and Cahill (2015); also, Gyawali et al. (2013) compared Penn Treebank and Universal POS tagsets and found that the more fine-grained PTB ones did better.

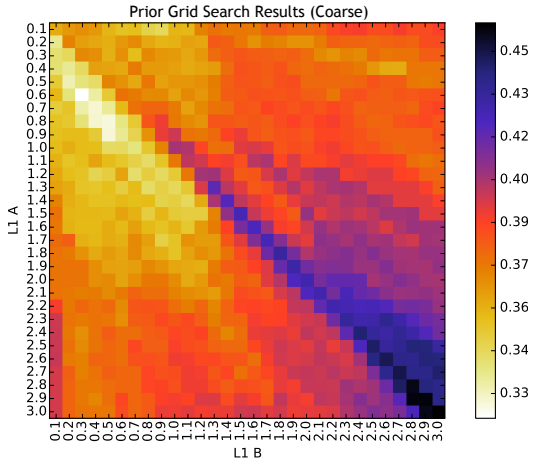


Figure 1: Heatmap over asymmetric priors on L1SEG-CLAWS2-ALL-DEV

the prior pair of $(0.6, 0.3)$ achieves a P_k value of 0.321, a substantial improvement over the previous best result of 0.370. Inspecting priors (see Figure 1 for a heatmap over priors) shows that the best results are in the region of weak priors for both values, which is consistent with the emphasis on compactness since weak priors result in more compact models (noted by *e.g.* Wang and Blei (2009)). Moreover, they are away from the diagonal, *i.e.* the L1SEG-COMP model will not produce the best results. A more fine-grained grid search, focusing on the range that provided the best results in the coarse search, can improve the results further still: over the interval $[0.3, 0.9]$, partitioned into 60 evenly spaced values, finds a prior pair of $(0.64, 0.32)$ that provides a slight improvement of the P_k value to 0.316.

As with L1SEG-COMP, we also evaluate this on the same held-out test set (Table 2). Applying the best asymmetric prior from the devset grid search, this improves to 0.266. Again, results across all 55 L1 pairs (Table 3) show the same pattern, and much as for L1SEG-COMP, priors are all weak or neutral (range $[0.1, 1.0]$). These results thus demonstrate that setting an asymmetric prior gives the best performance on this task.

6 Conclusion and Future Work

Applying the approach to our two illustrative applications of §1, patchwriting and literary analysis, would require development of relevant corpora. In both cases the distinction would be between native writing and writing that shows characteristics of a non-native speaker, rather than between two non-native L1s. There isn't yet a topic-balanced corpus like TOEFL11 which includes native speaker writing for evaluation, although we expect (given recent results on distinguishing native from non-native text in Malmasi and Dras (2015)) that the techniques should carry over. For the literary analysis, as well, to bridge the gap between work like Morzinski (1994) and a computational application, it remains to be seen how precise an annotation is possible for this task. Additionally, the granularity of segmentation may need to be finer than sentence-level, as suggested by the examples in §1; this level of granularity hasn't previously been tackled in unsupervised segmentation.

In terms of possible developments for the models presented for the task here, previous NLI work has shown that other, syntactic features can be useful for capturing L1-based differences. The incorporation of these features for this segmentation task could be a potentially fruitful avenue for future work. We have taken a fairly straightforward approach which modifies the generative story. A more sophisticated approach would be to incorporate features into the unsupervised model. One such example is the work of Berg-Kirkpatrick et al. (2010) which demonstrates that each component multinomial of a generative model can be turned into a miniature logistic regression model with the use of a modified EM algorithm. Their results showed that the feature-enhanced unsupervised models which incorporate linguistically-motivated features achieve substantial improvements for tasks such as POS induction and word segmentation. We note also that the models are potentially applicable to other stylistic segmentation tasks beyond L1 influence.

As far as this initial work is concerned we have shown that, framed as a segmentation task, it is possible to identify units of text that differ stylistically in their L1 influence. We demonstrated that it is possible to define a generative story and associated Bayesian models for stylistic segmentation, and further that segmentation results improve substantially by compacting the n-gram distributions, achieved by incorporating knowledge about discriminative features extracted from NLI models. Our best results come from a model that uses alternating asymmetric priors for each L1, with the priors selected using a grid search and then evaluated on a held-out test set.

Acknowledgements

The authors thank John Pate for very helpful discussions in the early stages of the paper, and the three anonymous referees for useful suggestions.

A Details on Dataset Generation and Evaluation

A.1 Document Generation

For our L1-varying datasets, we construct composite documents to be segmented as alternating segments drawn from TOEFL11 from two different L1s. Broadly following a standard approach (Brooke et al., 2012, for example), to generate such a document, we randomly draw TOEFL11 essays — each of which constitutes a segment — from the appropriate L1s and concatenate them, alternating the L1 class after each segment. This is repeated until the maximum number of segments per document, s, is reached. We generate multiple composite documents until all TOEFL11 have been used. In this work we use datasets generated with $s = 5$.[9] We follow the same process for our topic-varying datasets, but hold the L1 constant while alternating the topic (prompt).

A.2 Encoding n-gram information

Lau et al. (2013) investigated the importance of n-grams within topic models over lexical items. They note that in topic modelling each token receives a topic label and that the words in a collocation — *e.g. stock market*, *White House* or *health care* — may receive different topic assignments despite forming a single semantic unit. They found that identifying collocations (via a t-test) and preprocessing the text to turn these into single tokens provides a notable improvement over a unigram bag of words.

We implement a similar preprocessing step that converts each sentence within each document to a set of bigrams or trigrams using a sliding window, where each n-gram is represented by a single token. So, for example, the trigram `DT JJ NN` becomes a single token: `DT-JJ-NN`.

A.3 Evaluation: Metric Definitions

Given two segmentations r (reference) and h (hypothesis) for a corpus of N sentences,

[9] This is the average number of segments per chapter in the written text used by Eisenstein and Barzilay (2008). However, we have also successfully replicated our results using $s = 7, 9$.

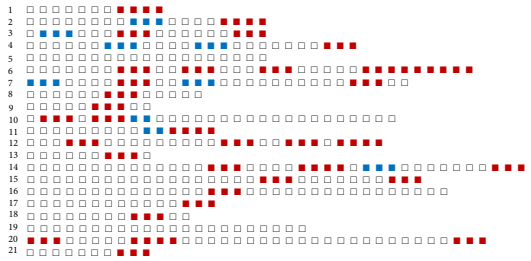


Figure 2: A visualization of sentences from a single segment. Each row represents a sentence and each token is represented by a square. Token trigrams considered discriminative for either of our two L1 classes are shown in blue or red, with the rest being considered non-discriminative.

$$P_D(r,h) = \sum_{1 \le i \le j \le N} D(i,j)(\delta_r(i,j) \bar{\oplus} \delta_h(i,j)) \quad (6)$$

where $\delta_r(i,j)$ is an indicator function specifying whether i and j lie in the same reference segment, $\delta_h(i,j)$ similarly for a hypothesised segment, $\bar{\oplus}$ is the XNOR function, and D is a distance probability distribution over the set of possible distances between sentences. For P_k, this D is defined by a fixed window of size k which contains all the probability mass, and k is set to be half the average reference segment length. The WD definition is:

$$WD(r,h) = \frac{1}{N-k} \sum_{i=1}^{N-k} (|b(r_i, r_{i+k}) - b(h_i, h_{i+k})| > 0) \quad (7)$$

where $b(r_i, r_j)$ represents the number of boundaries between positions i and j in the reference text (similarly, the hypothesis text).

A.4 Visualisation of Discriminative Features

Figure 2 shows a visualization of the discriminative features of a single segment where each row represents a sentence and each token is represented by a square. Tokens that are part of a trigram which is considered discriminative for either of our two L1 classes are shown in blue or red. Note that discriminative n-grams can overlap with each other within the same class (*e.g.* on lines 1 and 2 where two overlapping trigrams form a group of four consecutive tokens) and also between two classes (*e.g.* on lines 10 and 11).

References

Navot Akiva and Moshe Koppel. 2013. A Generic Unsupervised Method for Decomposing Multi-Author Documents. *Journal of the American Society for Information Science and Technology (JASIST)* 64(11):2256–2264.

Khaled Aldebei, Xiangjian He, and Jie Yang. 2015. Unsupervised decomposition of a multi-author document based on naive-bayesian model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 501–505. http://www.aclweb.org/anthology/P15-2082.

Calvin Bedient and Thomas Stearns Eliot. 1986. *He Do the Police in Different Voices: The Waste Land and its protagonist*. University of Chicago Press.

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical Models for Text Segmentation. *Machine Learning* 34(1-3):177–210. https://doi.org/10.1023/A:1007506220214.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 582–590. http://www.aclweb.org/anthology/N10-1083.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A Corpus of Non-Native English. Technical report, Educational Testing Service.

David M. Blei. 2012. Probabilistic topic models. *Communications of the ACM* 55(4):77–84.

Julian Brooke, Adam Hammond, and Graeme Hirst. 2012. Unsupervised Stylistic Segmentation of Poetry with Change Curves and Extrinsic Features. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*. Association for Computational Linguistics, Montréal, Canada, pages 26–35. http://www.aclweb.org/anthology/W12-2504.

Julian Brooke, Graeme Hirst, and Adam Hammond. 2013. Clustering voices in the waste land. In *Proceedings of the Workshop on Computational Linguistics for Literature*. Association for Computational Linguistics, Atlanta, Georgia, pages 41–46. http://www.aclweb.org/anthology/W13-1406.

Serhiy Bykh and Detmar Meurers. 2014. Exploring Syntactic Features for Native Language Identification: A Variationist Perspective on Feature Encoding and Ensemble Optimization. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* pages 1962–1973.

John Xiros Cooper. 1987. *TS Eliot and the politics of voice: The argument of The Waste Land*. 79. UMI Research Press.

Lan Du, Wray Buntine, and Mark Johnson. 2013. Topic segmentation with a structured topic model. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 190–200. http://www.aclweb.org/anthology/N13-1019.

Jacob Eisenstein. 2009. Hierarchical text segmentation from multi-scale lexical cohesion. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*. Boulder, CO, pages 353–361. www.aclweb.org/anthology/N09-1040.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Honolulu, Hawaii, pages 334–343. http://www.aclweb.org/anthology/D08-1035.

Rod Ellis. 2008. *The Study of Second Language Acquisition, 2nd edition*. Oxford University Press, Oxford, UK.

Joanna Gilmore, Denise Strickland, Briana Timmerman, Michelle Maher, and David Feldon. 2010. Weeds in the flower garden: An exploration of plagiarism in graduate students research proposals and its connection to enculturation, ESL, and contextual factors. *International Journal for Educational Integrity* 6(1):13–28.

Binod Gyawali, Gabriela Ramirez, and Thamar Solorio. 2013. Native Language Identification: a Simple n-gram Based Approach. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, Atlanta, Georgia, pages 224–231. http://www.aclweb.org/anthology/W13-1729.

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman Publishing Group.

Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Las Cruces, New Mexico, USA, pages 9–16. https://doi.org/10.3115/981732.981734.

Marti A. Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Lingustics* 23(1):33–64. http://www.aclweb.org/anthology/J97-1003.

Radu Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? A language-independent approach to native language identification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1363–1373. http://www.aclweb.org/anthology/D14-1142.

Sandra Jamieson and Rebecca Moore Howard. 2013. Sentence-Mining: Uncovering the Amount Of Reading and Reading Comprehension In College Writers' Researched Writing. In Randall McClure and James P. Purdy, editors, *The New Digital Scholar: Exploring and Enriching the Research and Writing Practices of NextGen Students*, American Society for Information Science and Technology, Medford, NJ, pages 111–133.

Minwoo Jeong and Ivan Titov. 2010. Unsupervised discourse segmentation of documents with inherently parallel structure. In *Proceedings of the ACL 2010 Conference Short Papers*. Association for Computational Linguistics, Uppsala, Sweden, pages 151–155. http://www.aclweb.org/anthology/P10-2028.

Casey Keck. 2006. The use of paraphrase in summary writing: A comparison of L1 and L2 writers. *Journal of Second Language Writing* 15(4):261–278.

Moshe Koppel, Navot Akiva, Idan Dershowitz, and Nachum Dershowitz. 2011. Unsupervised decomposition of a document into authorial components. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 1356–1364. http://www.aclweb.org/anthology/P11-1136.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically determining an anonymous author's native language. In *Intelligence and Security Informatics*. Springer-Verlag, volume 3495 of *LNCS*, pages 209–217.

Jey Han Lau, Timothy Baldwin, and David Newman. 2013. On Collocations and Topic Models. *ACM Transactions on Speech and Language Processing (TSLP)* 10(3).

Yongyan Li and Christine Pearson Casanave. 2012. Two first-year students strategies for writing from sources: Patchwriting or plagiarism? *Journal of Second Language Writing* 21:165–180.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. ACM, New York, NY, USA, pages 375–384. http://doi.acm.org/10.1145/1645953.1646003.

Chenghua Lin, Yulan He, Richard Everson, and Stefan Rüger. 2012. Weakly supervised joint sentiment-topic detection from text. *Knowledge and Data Engineering, IEEE Transactions on* 24(6):1134–1145.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, pages 25–32. https://doi.org/10.3115/1220175.1220179.

Shervin Malmasi and Aoife Cahill. 2015. Measuring feature diversity in native language identification. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, Denver, Colorado, pages 49–55. http://www.aclweb.org/anthology/W15-0606.

Shervin Malmasi and Mark Dras. 2014. Language Transfer Hypotheses with Linear SVM Weights. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1385–1390. http://aclweb.org/anthology/D14-1144.

Shervin Malmasi and Mark Dras. 2015. Multilingual Native Language Identification. *Natural Language Engineering* https://doi.org/10.1017/S1351324915000406.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational linguistics* 17(1):21–48.

Mary Morzinski. 1994. *The Linguistic influence of Polish on Joseph Conrad's style*. Columbia University Press, New York, NY.

Diane Pecorari. 2003. Good and original: Plagiarism and patchwriting in academic second-language writing. *Journal of Second Language Writing* 12(4):317–345.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics* 28(1):19–36. https://doi.org/10.1162/089120102317341756.

Ben Swanson and Eugene Charniak. 2013. Extracting the Native Language Signal for Second Language Acquisition. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 85–94. http://www.aclweb.org/anthology/N13-1009.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth*

Workshop on Innovative Use of NLP for Building Educational Applications. Association for Computational Linguistics, Atlanta, Georgia, pages 48–57. http://www.aclweb.org/anthology/W13-1706.

Joel Tetreault, Daniel Blanchard, Aoife Cahill, and Martin Chodorow. 2012. Native tongues, lost and found: Resources and empirical evaluations in native language identification. In *Proceedings of COLING 2012*. The COLING 2012 Organizing Committee, Mumbai, India, pages 2585–2602. http://www.aclweb.org/anthology/C12-1158.

Michelle Vieyra, Denise Strickland, and Brianna Timmerman. 2013. Patterns in plagiarism and patchwriting in science and engineering graduate students' research proposals. *International Journal for Educational Integrity* 9(1):35–49.

Hanna M. Wallach, David M. Mimno, and Andrew McCallum. 2009. Rethinking lda: Why priors matter. In Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, Curran Associates, Inc., pages 1973–1981.

Chong Wang and David M Blei. 2009. Decoupling sparsity and smoothness in the discrete hierarchical dirichlet process. In *Advances in Neural Information Processing Systems*. pages 1982–1989.

Sze-Meng Jojo Wong, Mark Dras, and Mark Johnson. 2012. Exploring Adaptor Grammars for Native Language Identification. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, pages 699–709. http://www.aclweb.org/anthology/D12-1064.

Weakly Supervised Cross-Lingual Named Entity Recognition via Effective Annotation and Representation Projection

Jian Ni and **Georgiana Dinu** and **Radu Florian**
IBM T. J. Watson Research Center
1101 Kitchawan Road, Yorktown Heights, NY 10598, USA
{nij, gdinu, raduf}@us.ibm.com

Abstract

The state-of-the-art named entity recognition (NER) systems are supervised machine learning models that require large amounts of manually annotated data to achieve high accuracy. However, annotating NER data by human is expensive and time-consuming, and can be quite difficult for a new language. In this paper, we present two weakly supervised approaches for cross-lingual NER with no human annotation in a target language. The first approach is to create automatically labeled NER data for a target language via annotation projection on comparable corpora, where we develop a heuristic scheme that effectively selects good-quality projection-labeled data from noisy data. The second approach is to project distributed representations of words (word embeddings) from a target language to a source language, so that the source-language NER system can be applied to the target language without re-training. We also design two co-decoding schemes that effectively combine the outputs of the two projection-based approaches. We evaluate the performance of the proposed approaches on both in-house and open NER data for several target languages. The results show that the combined systems outperform three other weakly supervised approaches on the CoNLL data.

1 Introduction

Named entity recognition (NER) is a fundamental information extraction task that automatically detects named entities in text and classifies them into pre-defined entity types such as PERSON, ORGANIZATION, GPE (GeoPolitical Entities), EVENT, LOCATION, TIME, DATE, etc. NER provides essential inputs for many information extraction applications, including relation extraction, entity linking, question answering and text mining. Building fast and accurate NER systems is a crucial step towards enabling large-scale automated information extraction and knowledge discovery on the huge volumes of electronic documents existing today.

The state-of-the-art NER systems are supervised machine learning models (Nadeau and Sekine, 2007), including maximum entropy Markov models (MEMMs) (McCallum et al., 2000), conditional random fields (CRFs) (Lafferty et al., 2001) and neural networks (Collobert et al., 2011; Lample et al., 2016). To achieve high accuracy, a NER system needs to be trained with a large amount of manually annotated data, and is often supplied with language-specific resources (e.g., gazetteers, word clusters, etc.). Annotating NER data by human is rather expensive and time-consuming, and can be quite difficult for a new language. This creates a big challenge in building NER systems of multiple languages for supporting multilingual information extraction applications.

The difficulty of acquiring supervised annotation raises the following question: given a well-trained NER system in a source language (e.g., English), how can one go about extending it to a new language with decent performance and no human annotation in the target language? There are mainly two types of approaches for building weakly supervised cross-lingual NER systems.

The first type of approaches create weakly labeled NER training data in a target language. One way to create weakly labeled data is through annotation projection on aligned parallel corpora or translations between a source language and a target language, e.g., (Yarowsky et al., 2001; Zitouni and Florian, 2008; Ehrmann et al., 2011). Another way is to utilize the text and structure of

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1470–1480
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1135

Wikipedia to generate weakly labeled multilingual training annotations, e.g., (Richman and Schone, 2008; Nothman et al., 2013; Al-Rfou et al., 2015).

The second type of approaches are based on direct model transfer, e.g., (Täckström et al., 2012; Tsai et al., 2016). The basic idea is to train a single NER system in the source language with language independent features, so the system can be applied to other languages using those universal features.

In this paper, we make the following contributions to weakly supervised cross-lingual NER with no human annotation in the target languages. First, for the *annotation projection* approach, we develop a heuristic, language-independent data selection scheme that seeks to select good-quality projection-labeled NER data from comparable corpora. Experimental results show that the data selection scheme can significantly improve the accuracy of the target-language NER system when the alignment quality is low and the projection-labeled data are noisy.

Second, we propose a new approach for direct NER model transfer based on *representation projection*. It projects word representations in vector space (word embeddings) from a target language to a source language, to create a universal representation of the words in different languages. Under this approach, the NER system trained for the source language can be directly applied to the target language without the need for re-training.

Finally, we design two *co-decoding* schemes that combine the outputs (views) of the two projection-based systems to produce an output that is more accurate than the outputs of individual systems. We evaluate the performance of the proposed approaches on both in-house and open NER data sets for a number of target languages. The results show that the combined systems outperform the state-of-the-art cross-lingual NER approaches proposed in Täckström et al. (2012), Nothman et al. (2013) and Tsai et al. (2016) on the CoNLL NER test data (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003).

We organize the paper as follows. In Section 2 we introduce three NER models that are used in the paper. In Section 3 we present an annotation projection approach with effective data selection. In Section 4 we propose a representation projection approach for direct NER model transfer. In Section 5 we describe two co-decoding schemes that effectively combine the outputs of two projection-based approaches. In Section 6 we evaluate the performance of the proposed approaches. We describe related work in Section 7 and conclude the paper in Section 8.

2 NER Models

The NER task can be formulated as a sequence labeling problem: given a sequence of words $x_1, ..., x_n$, we want to infer the NER tag l_i for each word x_i, $1 \leq i \leq n$. In this section we introduce three NER models that are used in the paper.

2.1 CRFs and MEMMs

Conditional random fields (CRFs) are a class of discriminative probabilistic graphical models that provide powerful tools for labeling sequential data (Lafferty et al., 2001). CRFs learn a conditional probability model $p_\lambda(\mathbf{l}|\mathbf{x})$ from a set of labeled training data, where $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ is a random sequence of input words, $\mathbf{l} = (\mathbf{l}_1, ..., \mathbf{l}_n)$ is the sequence of label variables (NER tags) for $\mathbf{x}$, and $\mathbf{l}$ has certain Markov properties conditioned on $\mathbf{x}$. Specifically, a general-order CRF with order o assumes that label variable $\mathbf{l}_i$ is dependent on a fixed number o of previous label variables $\mathbf{l}_{i-1}, ..., \mathbf{l}_{i-o}$, with the following conditional distribution:

$$p_\lambda(\mathbf{l}|\mathbf{x}) = \frac{e^{\sum_{i=1}^{n}\sum_{k=1}^{K}\lambda_k f_k(\mathbf{l}_i, \mathbf{l}_{i-1}, ..., \mathbf{l}_{i-o}, \mathbf{x})}}{Z_\lambda(\mathbf{x})} \quad (1)$$

where f_k's are feature functions, λ_k's are weights of the feature functions (parameters to learn), and $Z_\lambda(\mathbf{x})$ is a normalization constant. When $o = 1$, we have a first-order CRF which is also known as a linear-chain CRF.

Given a set of labeled training data $\mathcal{D} = (\mathbf{x}^{(j)}, \mathbf{l}^{(j)})_{j=1,...,N}$, we seek to find an optimal set of parameters λ^* that maximize the conditional log-likelihood of the data:

$$\lambda^* = \arg\max_\lambda \sum_{j=1}^{N} \log p_\lambda(\mathbf{l}^{(j)}|\mathbf{x}^{(j)}) \quad (2)$$

Once we obtain λ^*, we can use the trained model $p_{\lambda^*}(\mathbf{l}|\mathbf{x})$ to decode the most likely label sequence $\mathbf{l}^*$ for any new input sequence of words $\mathbf{x}$ (via the Viterbi algorithm for example):

$$\mathbf{l}^* = \arg\max_{\mathbf{l}} p_{\lambda^*}(\mathbf{l}|\mathbf{x}) \quad (3)$$

A related conditional probability model, called *maximum entropy Markov model* (MEMM) (McCallum et al., 2000), assumes that $\mathbf{l}$ is a Markov

chain conditioned on $\mathbf{x}$:

$$p_\lambda(\mathbf{l}|\mathbf{x}) = \prod_{i=1}^{n} p_\lambda(\mathbf{l}_i|\mathbf{l}_{i-1},...,\mathbf{l}_{i-o},\mathbf{x}) = \prod_{i=1}^{n} \frac{e^{\sum_{k=1}^{K}\lambda_k f_k(\mathbf{l}_i,\mathbf{l}_{i-1},...,\mathbf{l}_{i-o},\mathbf{x})}}{Z_\lambda(\mathbf{l}_{i-1},...,\mathbf{l}_{i-o},\mathbf{x})} \quad (4)$$

The main difference between CRFs and MEMMs is that CRFs normalize the conditional distribution over the whole sequence as in (1), while MEMMs normalize the conditional distribution per token as in (4). As a result, CRFs can better handle the label bias problem (Lafferty et al., 2001). This benefit, however, comes at a price. The training time of order-o CRFs grows exponentially ($O(M^{o+1})$) with the number of output labels M, which is typically slow even for moderate-size training data if M is large. In contrast, the training time of order-o MEMMs is linear ($O(M)$) with respect to M independent of o, so it can handle larger training data with higher order of dependency. We have implemented both a linear-chain CRF model and a general-order MEMM model.

2.2 Neural Networks

With the increasing popularity of distributed (vector) representations of words, neural network models have recently been applied to tackle many NLP tasks including NER (Collobert et al., 2011; Lample et al., 2016).

We have implemented a feedforward neural network model which maximizes the log-likelihood of the training data similar to that of (Collobert et al., 2011). We adopt a locally normalized model (the conditional distribution is normalized per token as in MEMMs) and introduce context dependency by conditioning on the previously assigned tags. We use a target word and its surrounding context as features. We do not use other common features such as gazetteers or character-level representations as such features might not be readily available or might not transfer to other languages.

We have deployed two neural network architectures. The first one (called NN1) uses the word embedding of a word as the input. The second one (called NN2) adds a smoothing prototype layer that computes the cosine similarity between a word embedding and a fixed set of prototype vectors (learned during training) and returns a weighted average of these prototype vectors as the input. In our experiments we find that with the smoothing layer, NN2 tends to have a more balanced precision and recall than NN1. Both networks have one hidden layer, with sigmoid and softmax activation functions on the hidden and output layers respectively. The two neural network models are depicted in Figure 1.

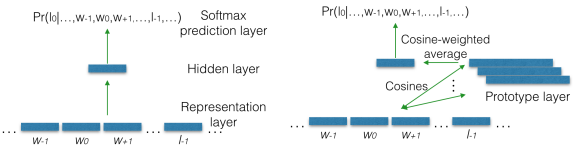


Figure 1: Architecture of the two neural network models: left-NN1, right-NN2.

3 Annotation Projection Approach

The existing annotation projection approaches require parallel corpora or translations between a source language and a target language with alignment information. In this paper, we develop a heuristic, language-independent data selection scheme that seeks to select good-quality projection-labeled data from noisy comparable corpora. We use English as the source language.

Suppose we have comparable[1] sentence pairs $(\mathbf{X},\mathbf{Y})$ between English and a target language, where $\mathbf{X}$ includes N English sentences $\mathbf{x}^{(1)},...,\mathbf{x}^{(N)}$, $\mathbf{Y}$ includes N target-language sentences $\mathbf{y}^{(1)},...,\mathbf{y}^{(N)}$, and $\mathbf{y}^{(j)}$ is aligned to $\mathbf{x}^{(j)}$ via an alignment model, $1 \le j \le N$. We use a sentence pair $(\mathbf{x},\mathbf{y})$ as an example to illustrate how the annotation projection procedure works, where $\mathbf{x} = (x_1, x_2, ..., x_s)$ is an English sentence, and $\mathbf{y} = (y_1, y_2, ..., y_t)$ is a target-language sentence that is aligned to $\mathbf{x}$.

Annotation Projection Procedure

1. Apply the English NER system on the English sentence $\mathbf{x}$ to generate the NER tags $\mathbf{l} = (l_1, l_2, ..., l_s)$ for $\mathbf{x}$.

2. Project the NER tags to the target-language sentence $\mathbf{y}$ using the alignment information. Specifically, if a sequence of English words $(x_i, ..., x_{i+p})$ is aligned to a sequence of target-language words $(y_j, ..., y_{j+q})$, and $(x_i, ..., x_{i+p})$ is recognized (by the English NER system) as an entity with NER tag l,

[1] Ideally, the sentences would be translations of each other, but we only require possibly parallel sentences.

then $(y_j, ..., y_{j+q})$ is labeled with l[2].
Let $\mathbf{l}' = (l'_1, l'_2, ..., l'_t)$ be the projected NER tags for the target-language sentence $\mathbf{y}$.

We can apply the annotation projection procedure on all the sentence pairs $(\mathbf{X}, \mathbf{Y})$, to generate projected NER tags $\mathbf{L}'$ for the target-language sentences $\mathbf{Y}$. ($\mathbf{Y}$, $\mathbf{L}'$) are automatically labeled NER data with no human annotation in the target language. One can use those projection-labeled data to train an NER system in the target language. The quality of such weakly labeled NER data, and consequently the accuracy of the target-language NER system, depend on both 1) the accuracy of the English NER system, and 2) the alignment accuracy of the sentence pairs.

Since we don't require actual translations, but only comparable data, the downside is that if some of the data are not actually parallel and if we use all for weakly supervised learning, the accuracy of the target-language NER system might be adversely affected. We are therefore motivated to design effective data selection schemes that can select good-quality projection-labeled data from noisy data, to improve the accuracy of the annotation projection approach for cross-lingual NER.

3.1 Data Selection Scheme

We first design a metric to measure the annotation quality of a projection-labeled sentence in the target language. We construct a frequency table T which includes all the entities in the projection-labeled target-language sentences. For each entity e, T also includes the projected NER tags for e and the relative frequency (empirical probability) $\hat{P}(l|e)$ that entity e is labeled with tag l. Table 1 shows a snapshot of the frequency table where the target language is Portuguese.

We use $\hat{P}(l|e)$ to measure the reliability of labeling entity e with tag l in the target language. The intuition is that if an entity e is labeled by a tag l with higher frequency than other tags in the projection-labeled data, it is more likely that the annotation is correct. For example, if the joint accuracy of the source NER system and alignment system is greater than 0.5, then the correct tag of a random entity will have a higher relative frequency than other tags in a large enough sample.

Based on the frequency scores, we calculate the quality score of a projection-labeled target-language sentence $\mathbf{y}$ by averaging the frequency scores of the projected entities in the sentence:

$$q(\mathbf{y}) = \frac{\Sigma_{e \in \mathbf{y}} \hat{P}(l'(e)|e)}{n(\mathbf{y})} \quad (5)$$

where $l'(e)$ is the projected NER tag for e, and $n(\mathbf{y})$ is the total number of entities in sentence $\mathbf{y}$.

We use $q(\mathbf{y})$ to measure the annotation quality of sentence $\mathbf{y}$, and $n(\mathbf{y})$ to measure the amount of annotation information contained in sentence $\mathbf{y}$. We design a *heuristic data selection scheme* which selects projection-labeled sentences in the target language that satisfy the following condition:

$$q(\mathbf{y}) \geq q;\ n(\mathbf{y}) \geq n \quad (6)$$

where q is a quality score threshold and n is an entity number threshold. We can tune the two parameters to make tradeoffs among the annotation quality of the selected sentences, the annotation information contained in the selected sentences, and the total number of sentence selected.

One way to select the threshold parameters q and n is via a development set - either a small set of human-annotated data or a sample of the projection-labeled data. We select the threshold parameters via *coordinate search* using the development set: we first fix $n = 3$ and search the best $\hat{q}$ in $[0, 0.9]$ with a step size of 0.1; we then fix $q = \hat{q}$ and select the best $\hat{n}$ in $[1, 5]$ with a step size of 1.

Entity Name	NER Tag	Frequency
Estados Unidos	GPE	0.853
Estados Unidos	ORGANIZATION	0.143
Estados Unidos	PEOPLE	0.001
Estados Unidos	PRODUCT	0.001
Estados Unidos	TITLEWORK	0.001
Estados Unidos	EVENT	0.001

Table 1: A snapshot of the frequency table where the target language is Portuguese. *Estados Unidos* means *United States*. The correct NER tag for Estados Unidos is GPE which has the highest relative frequency in the weakly labeled data.

3.2 Accuracy Improvements

We evaluate the effectiveness of the data selection scheme via experiments on 4 target languages: Japanese, Korean, German and Portuguese. We use comparable corpora between English and each target language (ranging from 2M to 6M tokens) with alignment information. For each target language, we also have a set of manually annotated NER data (ranging from 30K to 45K tokens)

[2]If the IOB (Inside, Outside, Beginning) tagging format is used, then $(y_j, y_{j+1}, ..., y_{j+q})$ is labeled with (B-l, I-l,...,I-l).

Language	(q, n)	Training Size	F_1 Score
Japanese	(0, 0)	4.9M	41.2
	(0.7, 4)	1.3M	53.4
Korean	(0, 0)	4.5M	25.0
	(0.4, 2)	1.5M	38.7
German	(0, 0)	5.2M	67.2
	(0.4, 4)	2.6M	67.5
Portuguese	(0, 0)	2.1M	61.5
	(0.1, 4)	1.5M	62.7

Table 2: Performance comparison of weakly supervised NER systems trained without data selection ($(q, n) = (0, 0)$) and with data selection ($(\hat{q}, \hat{n})$ determined by coordinate search).

which are served as the test data for evaluating the target-language NER system.

The source (English) NER system is a linear-chain CRF model which achieves an accuracy of 88.9 F_1 score on an independent NER test set. The alignment systems between English and the target languages are maximum entropy models (Ittycheriah and Roukos, 2005), with an accuracy of 69.4/62.0/76.1/88.0 F_1 score on independent Japanese/Korean/German/Portuguese alignment test sets.

For each target language, we randomly select 5% of the projection-labeled data as the development set and the remaining 95% as the training set. We compare an NER system trained with all the projection-labeled training data with no data selection (i.e., $(q, n) = (0, 0)$) and an NER system trained with projection-labeled data selected by the data selection scheme where the development set is used to select the threshold parameters q and n via coordinate search. Both NER systems are 2nd-order MEMM models[3] which use the same template of features.

The results are shown in Table 2. For different target languages, we use the same source (English) NER system for annotation projection, so the differences in the accuracy improvements are mainly due to the alignment quality of the comparable corpora between English and different target languages. When the alignment quality is low (e.g., as for Japanese and Korean) and hence the projection-labeled NER data are quite noisy, the proposed data selection scheme is very effective in selecting good-quality projection-labeled data and the improvement is big: +12.2 F_1 score for Japanese and +13.7 F_1 score for Korean. Using a stratified shuffling test (Noreen, 1989), for a significance level of 0.05, data-selection is statistically significantly better than no-selection for Japanese, Korean and Portuguese.

[3]In our experiments, CRFs cannot handle training data with a few million words, since our NER system has over 50 entity types, and the training time of CRFs grows at least quadratically in the number of entity types.

4 Representation Projection Approach

In this paper, we propose a new approach for direct NER model transfer based on representation projection. Under this approach, we train a single English NER system that uses only word embeddings as input representations. We create mapping functions which can map words in any language into English and we simply use the English NER system to decode. In particular, by mapping all languages into English, we are using one universal NER system and we do not need to re-train the system when a new language is added.

4.1 Monolingual Word Embeddings

We first build vector representations of words (word embeddings) for a language using monolingual data. We use a variant of the Continuous Bag-of-Words (CBOW) word2vec model (Mikolov et al., 2013a), which concatenates the context words surrounding a target word instead of adding them (similarly to (Ling et al., 2015)). Additionally, we employ weights $w = \frac{1}{dist(x,x_c)}$ that decay with the distance of a context word x_c to a target word x. Tests on word similarity benchmarks show this variant leads to small improvements over the standard CBOW model.

We train 300-dimensional word embeddings for English. Following (Mikolov et al., 2013b), we use larger dimensional embeddings for the target languages, namely 800. We train word2vec for 1 epoch for English/Spanish and 5 epochs for the rest of the languages for which we have less data.

4.2 Cross-Lingual Representation Projection

We learn cross-lingual word embedding mappings, similarly to (Mikolov et al., 2013b). For a target language f, we first extract a small training dictionary from a phrase table that includes word-to-word alignments between English and the target language f. The dictionary contains English and target-language word pairs with weights: $(x_i, y_i, w_i)_{i=1,...,n}$, where x_i is an English word, y_i is a target-language word, and the weight $w_i = \hat{P}(x_i|y_i)$ is the relative frequency of x_i given y_i as extracted from the phrase table.

Suppose we have monolingual word embeddings for English and the target language f. Let $\mathbf{u_i} \in \mathcal{R}^{d_1}$ be the vector representation for English word x_i, $\mathbf{v_i} \in \mathcal{R}^{d_2}$ be the vector representation for target-language word y_i. We find a linear mapping $\mathbf{M}_{f \to e}$ by solving the following weighted least squares problem where the dictionary is used as the training data:

$$\mathbf{M}_{f \to e} = \arg\min_{\mathbf{M}} \sum_{i=1}^{n} w_i ||\mathbf{u_i} - \mathbf{M}\mathbf{v_i}||^2 \quad (7)$$

In (7) we generalize the formulation in (Mikolov et al., 2013b) by adding frequency weights to the word pairs, so that more frequent pairs are of higher importance. Using $\mathbf{M}_{f \to e}$, for any new word in f with vector representation $\mathbf{v}$, we can project it into the English vector space as the vector $\mathbf{M}_{f \to e}\mathbf{v}$.

The training dictionary plays a key role in finding an effective cross-lingual embedding mapping. To control the size of the dictionary, we only include word pairs with a minimum frequency threshold. We set the threshold to obtain approximately 5K to 6K unique word pairs for a target language, as our experiments show that larger-size dictionaries might harm the performance of representation projection for direct NER model transfer.

4.3 Direct NER Model Transfer

The source (English) NER system is a neural network model (with architecture NN1 or NN2) that uses only word embedding features (embeddings of a word and its surrounding context) in the English vector space. Model transfer is achieved simply by projecting the target language word embeddings into the English vector space and decoding these using the English NER system.

More specifically, given the word embeddings of a sequence of words in a target language f, $(\mathbf{v}_1, ..., \mathbf{v}_t)$, we project them into the English vector space by applying the linear mapping $\mathbf{M}_{f \to e}$: $(\mathbf{M}_{f \to e}\mathbf{v}_1, ..., \mathbf{M}_{f \to e}\mathbf{v}_t)$. The English NER system is then applied on the projected input to produce NER tags. Words not in the target-language vocabulary are projected into their English embeddings if they are found in the English vocabulary, or into an NER-trained UNK vector otherwise.

5 Co-Decoding

Given two weakly supervised NER systems which are trained with different data using different models (MEMM model for annotation projection and neural network model for representation projection), we would like to design a *co-decoding* scheme that can combine the outputs (views) of the two systems to produce an output that is more accurate than the outputs of individual systems.

Since both systems are statistical models and can produce confidence scores (probabilities), a natural co-decoding scheme is to compare the confidence scores of the NER tags generated by the two systems and select the tags with higher confidences scores. However, confidence scores of two weakly supervised systems may not be directly comparable, especially when comparing O tags with non-O tags (i.e., entity tags). We consider an *exclude-O confidence-based co-decoding scheme* which we find to be more effective empirically. It is similar to the pure confidence-based scheme, with the only difference that it always prefers a non-O tag of one system to an O tag of the other system, regardless of their confidence scores.

In our experiments we find that the annotation projection system tends to have a high precision and low recall, i.e., it detects fewer entities, but for the detected entities the accuracy is high. The representation projection system tends to have a more balanced precision and recall. Based on this observation, we develop the following *rank-based co-decoding scheme* that gives higher priority to the high-precision annotation projection system:

1. The combined output includes all the entities detected by the annotation projection system.

2. It then adds all the entities detected by the representation projection system that do not conflict[4] with entities detected by the annotation projection system (to improve recall).

Note that an entity X detected by the representation projection system does not conflict with the annotation projection system if the annotation projection system produces O tags for the entire span of X. For example, suppose the output tag sequence of annotation projection is (B-PER,O,O,O,O), of representation projection is (B-ORG,I-ORG,O,B-LOC,I-LOC), then the combined output under the rank-based scheme will be (B-PER,O,O,B-LOC,I-LOC).

[4]Two entities detected by two different systems conflict with each other if either 1) the two entities have different spans but overlap with each other; or 2) the two entities have the same span but with different NER tags.

Japanese	P	R	F_1
Annotation-Projection (AP)	69.9	43.2	53.4
Representation-Projection (NN1)	71.5	36.6	48.4
Representation-Projection (NN2)	59.9	42.4	49.7
Co-Decoding (Conf): AP+NN1	65.7	49.5	56.5
Co-Decoding (Rank): AP+NN1	68.3	51.6	**58.8**
Co-Decoding (Conf): AP+NN2	59.5	53.3	56.2
Co-Decoding (Rank): AP+NN2	61.6	54.5	57.8
Supervised (272K)	*84.5*	*80.9*	*82.7*
Korean	**P**	**R**	**F_1**
Annotation-Projection (AP)	69.5	26.8	38.7
Representation-Projection (NN1)	66.1	23.2	34.4
Representation-Projection (NN2)	68.5	43.4	53.1
Co-Decoding (Conf): AP+NN1	68.2	41.0	51.2
Co-Decoding (Rank): AP+NN1	71.3	42.8	53.5
Co-Decoding (Conf): AP+NN2	68.9	53.4	60.2
Co-Decoding (Rank): AP+NN2	70.0	53.3	**60.5**
Supervised (97K)	*88.2*	*74.0*	*80.4*
German	**P**	**R**	**F_1**
Annotation-Projection (AP)	76.5	60.5	67.5
Representation-Projection (NN1)	69.0	48.8	57.2
Representation-Projection (NN2)	63.7	66.1	64.9
Co-Decoding (Conf): AP+NN1	68.5	61.7	64.9
Co-Decoding (Rank): AP+NN1	72.7	65.0	68.6
Co-Decoding (Conf): AP+NN2	64.7	71.3	67.9
Co-Decoding (Rank): AP+NN2	67.1	72.6	**69.7**
Supervised (125K)	*77.8*	*68.1*	*72.6*
Portuguese	**P**	**R**	**F_1**
Annotation-Projection (AP)	84.0	50.1	62.7
Representation-Projection (NN1)	70.5	47.6	56.8
Representation-Projection (NN2)	66.0	63.4	64.7
Co-Decoding (Conf): AP+NN1	72.0	55.8	62.9
Co-Decoding (Rank): AP+NN1	77.5	59.7	67.4
Co-Decoding (Conf): AP+NN2	68.1	67.1	67.6
Co-Decoding (Rank): AP+NN2	70.9	68.3	**69.6**
Supervised (173K)	*79.8*	*71.9*	*75.6*

Table 3: In-house NER data: Precision, Recall and F_1 score on exact phrasal matches. The highest F_1 score among all the weakly supervised approaches is shown in bold. Same for Tables 4 and 5.

6 Experiments

In this section, we evaluate the performance of the proposed approaches for cross-lingual NER, including the 2 projection-based approaches and the 2 co-decoding schemes for combining them:
(1) The annotation projection (AP) approach with heuristic data selection;
(2) The representation projection approach (with two neural network architectures NN1 and NN2);
(3) The exclude-O confidence-based co-decoding scheme;
(4) The rank-based co-decoding scheme.

6.1 NER Data Sets

We have used various NER data sets for evaluation. The first group includes in-house human-annotated newswire NER data for four languages: Japanese, Korean, German and Portuguese, annotated with over 50 entity types. The main motivation of deploying such a fine-grained entity type set is to build cognitive question answering applications on top of the NER systems. The entity type set has been engineered to cover many of the frequent entity types that are targeted by naturally-phrased questions. The sizes of the test data sets are ranging from 30K to 45K tokens.

The second group includes open human-annotated newswire NER data for Spanish, Dutch and German from the CoNLL NER data sets (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). The CoNLL data have 4 entity types: PER (persons), ORG (organizations), LOC (locations) and MISC (miscellaneous entities). The sizes of the development/test data sets are ranging from 35K to 70K tokens. The development data are used for tuning the parameters of learning methods.

6.2 Evaluation for In-House NER Data

In Table 3, we show the results of different approaches for the in-house NER data. For annotation projection, the source (English) NER system is a linear-chain CRF model trained with 328K tokens of human-annotated English newswire data. The target-language NER systems are 2nd-order MEMM models trained with 1.3M, 1.5M, 2.6M and 1.5M tokens of projection-labeled data for Japanese, Korean, German and Portuguese, respectively. The projection-labeled data are selected using the heuristic data selection scheme (see Table 2). For representation projection, the source (English) NER systems are neural network models with architectures NN1 and NN2 (see Figure 1), both trained with 328K tokens of human-annotated English newswire data.

The results show that the annotation projection (AP) approach has a relatively high precision and low recall. For representation projection, neural network model NN2 (with a smoothing layer) is better than NN1, and NN2 tends to have a more balanced precision and recall. The rank-based co-decoding scheme is more effective for combining the two projection-based approaches. In particular, the rank-based scheme that combines AP and NN2 achieves the highest F_1 score among all the weakly supervised approaches for Korean, German and Portuguese (second highest F_1 score for Japanese), and it improves over the best of the two

Spanish	P	R	F_1
Annotation-Projection (AP)	65.5	59.1	62.1
Representation-Projection (NN1)	63.9	52.2	57.4
Representation-Projection (NN2)	55.3	51.8	53.5
Co-Decoding (Conf): AP+NN1	64.3	66.8	**65.5**
Co-Decoding (Rank): AP+NN1	63.7	65.3	64.5
Co-Decoding (Conf): AP+NN2	58.0	63.9	60.8
Co-Decoding (Rank): AP+NN2	60.8	64.5	62.6
Supervised (264K)	*81.3*	*79.8*	*80.6*
Dutch	**P**	**R**	**F_1**
Annotation-Projection (AP)	73.3	63.0	67.8
Representation-Projection (NN1)	82.6	47.4	60.3
Representation-Projection (NN2)	66.3	43.5	52.5
Co-Decoding (Conf): AP+NN1	72.3	66.5	**69.3**
Co-Decoding (Rank): AP+NN1	72.8	65.3	68.8
Co-Decoding (Conf): AP+NN2	65.3	64.7	65.0
Co-Decoding (Rank): AP+NN2	69.7	66.0	67.8
Supervised (199K)	*82.9*	*81.7*	*82.3*
German	**P**	**R**	**F_1**
Annotation-Projection (AP)	71.8	54.7	62.1
Representation-Projection (NN1)	79.4	41.4	54.4
Representation-Projection (NN2)	64.6	42.7	51.4
Co-Decoding (Conf): AP+NN1	70.1	59.5	64.4
Co-Decoding (Rank): AP+NN1	71.0	59.4	**64.7**
Co-Decoding (Conf): AP+NN2	64.2	59.9	62.0
Co-Decoding (Rank): AP+NN2	66.8	60.6	63.6
Supervised (206K)	*81.2*	*64.3*	*71.8*

Table 4: CoNLL NER development data.

projection-based systems by 2.2 to 7.4 F_1 score.

We also provide the performance of *supervised learning* where the NER system is trained with human-annotated data in the target language (with size shown in the bracket). While the performance of the weakly supervised systems is not as good as supervised learning, it is important to build weakly supervised systems with decent performance when supervised annotation is unavailable. Even if supervised annotation is feasible, the weakly supervised systems can be used to pre-annotate the data, and we observed that pre-annotation can improve the annotation speed by 40%-60%, which greatly reduces the annotation cost.

6.3 Evaluation for CoNLL NER Data

For the CoNLL data, the source (English) NER system for annotation projection is a linear-chain CRF model trained with the CoNLL English training data (203K tokens), and the target-language NER systems are 2nd-order MEMM models trained with 1.3M, 7.0M and 1.2M tokens of projection-labeled data for Spanish, Dutch and German, respectively. The projection-labeled data are selected using the heuristic data selection scheme, where the threshold parameters q and n are determined via coordinate search based on the CoNLL development sets. Compared with no data selection, the data selection scheme improves the annotation projection approach by 2.7/2.0/2.7 F_1 score on the Spanish/Dutch/German development data. In addition to standard NER features such as n-gram word features, word type features, prefix and suffix features, the target-language NER systems also use the multilingual Wikipedia entity type mappings developed in (Ni and Florian, 2016) to generate dictionary features and as decoding constraints, which improve the annotation projection approach by 3.0/5.4/7.9 F_1 score on the Spanish/Dutch/German development data.

For representation projection, the source (English) NER systems are neural network models (NN1 and NN2) trained with the CoNLL English training data. Compared with the standard CBOW word2vec model, the concatenated variant improves the representation projection approach (NN1) by 8.9/11.4/6.8 F_1 score on the Spanish/Dutch/German development data, as well as by 2.0 F_1 score on English. In addition, the frequency-weighted cross-lingual word embedding projection (7) improves the representation projection approach (NN1) by 2.2/6.3/3.7 F_1 score on the Spanish/Dutch/German development data, compared with using uniform weights on the same data. We do observe, however, that using uniform weights when keeping only the most frequent translation of a word instead of all word pairs above a threshold in the training dictionary, leads to performance similar to that of the frequency-weighted projection.

In Table 4 we show the results for the CoNLL development data. For representation projection, NN1 is better than NN2. Both the annotation projection approach and NN1 tend to have a high precision. In this case, the exclude-O confidence-based co-decoding scheme that combines AP and NN1 achieves the highest F_1 score for Spanish and Dutch (second highest F_1 score for German), and improves over the best of the two projection-based systems by 1.5 to 3.4 F_1 score.

In Table 5 we compare our top systems (confidence or rank-based co-decoding of AP and NN1, determined by the development data) with the best results of the cross-lingual NER approaches proposed in Täckström et al. (2012), Nothman et al. (2013) and Tsai et al. (2016) on the CoNLL test data. Our systems outperform the previous state-of-the-art approaches, closing more of the gap to

Spanish	P	R	F_1
Täckström et al. (2012)	x	x	59.3
Nothman et al. (2013)	x	x	61.0
Tsai et al. (2016)	x	x	60.6
Co-Decoding (Conf): AP+NN1	64.9	65.2	**65.1**
Co-Decoding (Rank): AP+NN1	64.6	63.9	64.3
Supervised (264K)	*82.5*	*82.3*	*82.4*
Dutch	**P**	**R**	**F_1**
Täckström et al. (2012)	x	x	58.4
Nothman et al. (2013)	x	x	64.0
Tsai et al. (2016)	x	x	61.6
Co-Decoding (Conf): AP+NN1	69.1	62.0	**65.4**
Co-Decoding (Rank): AP+NN1	69.3	61.0	64.8
Supervised (199K)	*85.1*	*83.9*	*84.5*
German	**P**	**R**	**F_1**
Täckström et al. (2012)	x	x	40.4
Nothman et al. (2013)	x	x	55.8
Tsai et al. (2016)	x	x	48.1
Co-Decoding (Conf): AP+NN1	68.5	51.0	**58.5**
Co-Decoding (Rank): AP+NN1	68.3	50.4	58.0
Supervised (206K)	*79.6*	*65.3*	*71.8*

Table 5: CoNLL NER test data.

supervised learning.

7 Related Work

The traditional annotation projection approaches (Yarowsky et al., 2001; Zitouni and Florian, 2008; Ehrmann et al., 2011) project NER tags across language pairs using parallel corpora or translations. Wang and Manning (2014) proposed a variant of annotation projection which projects expectations of tags and uses them as constraints to train a model based on generalized expectation criteria. Annotation projection has also been applied to several other cross-lingual NLP tasks, including word sense disambiguation (Diab and Resnik, 2002), part-of-speech (POS) tagging (Yarowsky et al., 2001) and dependency parsing (Rasooli and Collins, 2015).

Wikipedia has been exploited to generate weakly labeled multilingual NER training data. The basic idea is to first categorize Wikipedia pages into entity types, either based on manually constructed rules that utilize the category information of Wikipedia (Richman and Schone, 2008) or Freebase attributes (Al-Rfou et al., 2015), or via a classifier trained with manually labeled Wikipedia pages (Nothman et al., 2013). Heuristic rules are then developed in these works to automatically label the Wikipedia text with NER tags. Ni and Florian (2016) built high-accuracy, high-coverage multilingual Wikipedia entity type mappings using weakly labeled data and applied those mappings as decoding constrains or dictionary features to improve multilingual NER systems.

For direct NER model transfer, Täckström et al. (2012) built cross-lingual word clusters using monolingual data in source/target languages and aligned parallel data between source and target languages. The cross-lingual word clusters were then used to generate universal features. Tsai et al. (2016) applied the cross-lingual wikifier developed in (Tsai and Roth, 2016) and multilingual Wikipedia dump to generate language-independent labels (FreeBase types and Wikipedia categories) for n-grams in a document, and those labels were used as universal features.

Different ways of obtaining cross-lingual embeddings have been proposed in the literature. One approach builds monolingual representations separately and then brings them to the same space typically using a seed dictionary (Mikolov et al., 2013b; Faruqui and Dyer, 2014). Another line of work builds inter-lingual representations simultaneously, often by generating mixed language corpora using the supervision at hand (aligned sentences, documents, etc.) (Vulić and Moens, 2015; Gouws et al., 2015). We opt for the first solution in this paper because of its flexibility: we can map all languages to English rather than requiring separate embeddings for each language pair. Additionally we are able to easily add a new language without any constraints on the type of data needed. Note that although we do not specifically create inter-lingual representations, by training mappings to the common language, English, we are able to map words in different languages to a common space. Similar approaches for cross-lingual model transfer have been applied to other NLP tasks such as document classification (Klementiev et al., 2012), dependency parsing (Guo et al., 2015) and POS tagging (Gouws and Søgaard, 2015).

8 Conclusion

In this paper, we developed two weakly supervised approaches for cross-lingual NER based on effective annotation and representation projection. We also designed two co-decoding schemes that combine the two projection-based systems in an intelligent way. Experimental results show that the combined systems outperform three state-of-the-art cross-lingual NER approaches, providing a strong baseline for building cross-lingual NER systems with no human annotation in target languages.

References

Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2015. Polyglot-ner: Massive multilingual named entity recognition. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, Vancouver, British Columbia, Canada. https://doi.org/10.1137/1.9781611974010.66.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537. http://dl.acm.org/citation.cfm?id=1953048.2078186.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, ACL'02, pages 255–262. https://doi.org/10.3115/1073083.1073126.

Maud Ehrmann, Marco Turchi, and Ralf Steinberger. 2011. Building a multilingual named entity-annotated corpus using annotation projection. In *Proceedings of Recent Advances in Natural Language Processing*. Association for Computational Linguistics, pages 118–124. http://aclweb.org/anthology/R11-1017.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, pages 462–471. http://www.aclweb.org/anthology/E14-1049.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning*. JMLR Workshop and Conference Proceedings, pages 748–756. http://jmlr.org/proceedings/papers/v37/gouws15.pdf.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1386–1390. http://www.aclweb.org/anthology/N15-1157.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Beijing, China, pages 1234–1244. http://www.aclweb.org/anthology/P15-1119.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. pages 89–96. http://aclweb.org/anthology/H05-1012.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*. The COLING 2012 Organizing Committee, Mumbai, India, pages 1459–1474. http://www.aclweb.org/anthology/C12-1089.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML'01, pages 282–289. http://dl.acm.org/citation.cfm?id=645530.655813.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 260–270. https://doi.org/10.18653/v1/N16-1030.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1299–1304. http://www.aclweb.org/anthology/N15-1142.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML'00, pages 591–598. http://dl.acm.org/citation.cfm?id=645529.658277.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *CoRR* abs/1309.4168. http://arxiv.org/abs/1309.4168.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes* 30(1):3–26. Publisher: John Benjamins Publishing Company. https://doi.org/10.1075/li.30.1.03nad.

Jian Ni and Radu Florian. 2016. Improving multilingual named entity recognition with wikipedia entity type mapping. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1275–1284. https://doi.org/10.18653/v1/D16-1135.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. John Wiley & Sons, Inc., New York, NY, USA.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Journal of Artificial Intelligence* 194:151–175. https://doi.org/10.1016/j.artint.2012.03.006.

Sadegh Mohammad Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 328–338. https://doi.org/10.18653/v1/D15-1039.

E. Alexander Richman and Patrick Schone. 2008. Mining wiki resources for multilingual named entity recognition. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, pages 1–9. http://aclweb.org/anthology/P08-1001.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 477–487. http://aclweb.org/anthology/N12-1052.

Erik F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the Sixth Conference on Natural Language Learning - Volume 20*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL'02, pages 1–4. https://doi.org/10.3115/1118853.1118877.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, Stroudsburg, PA, USA, CONLL'03, pages 142–147. https://doi.org/10.3115/1119176.1119195.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 219–228. https://doi.org/10.18653/v1/K16-1022.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual wikification using multilingual embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 589–598. https://doi.org/10.18653/v1/N16-1072.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Beijing, China, pages 719–725. http://www.aclweb.org/anthology/P15-2118.

Mengqiu Wang and D. Christopher Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association of Computational Linguistics* 2:55–66. http://aclweb.org/anthology/Q14-1005.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT'01, pages 1–8. https://doi.org/10.3115/1072133.1072187.

Imed Zitouni and Radu Florian. 2008. Mention detection crossing the language barrier. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 600–609. http://aclweb.org/anthology/D08-1063.

Context Sensitive Lemmatization Using Two Successive Bidirectional Gated Recurrent Networks

Abhisek Chakrabarty **Onkar Arun Pandit** **Utpal Garain**
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
203 B.T. Road, Kolkata-700108, India
abhisek0842@gmail.com, oapandit@gmail.com, utpal@isical.ac.in

Abstract

We introduce a composite deep neural network architecture for supervised and language independent context sensitive lemmatization. The proposed method considers the task as to identify the correct edit tree representing the transformation between a word-lemma pair. To find the lemma of a surface word, we exploit two successive bidirectional gated recurrent structures - the first one is used to extract the character level dependencies and the next one captures the contextual information of the given word. The key advantages of our model compared to the state-of-the-art lemmatizers such as Lemming and Morfette are - (i) it is independent of human decided features (ii) except the gold lemma, no other expensive morphological attribute is required for joint learning. We evaluate the lemmatizer on nine languages - Bengali, Catalan, Dutch, Hindi, Hungarian, Italian, Latin, Romanian and Spanish. It is found that except Bengali, the proposed method outperforms Lemming and Morfette on the other languages. To train the model on Bengali, we develop a gold lemma annotated dataset[1] (having $1,702$ sentences with a total of $20,257$ word tokens), which is an additional contribution of this work.

1 Introduction

Lemmatization is the process to determine the root/dictionary form of a surface word. Morphologically rich languages suffer due to the existence of various inflectional and derivational variations of a root depending on several linguistic properties such as honorificity, parts of speech (POS), person, tense etc. Lemmas map the related word forms to lexical resources thus identifying them as the members of the same group and providing their semantic and syntactic information. Stemming is a way similar to lemmatization producing the common portion of variants but it has several limitations - (i) there is no guarantee of a stem to be a legitimate word form (ii) words are considered in isolation. Hence, for context sensitive languages i.e. where same inflected word form may come from different sources and can only be disambiguated by considering its neighbouring information, there lemmatization defines the foremost task to handle diverse text processing problems (e.g. sense disambiguation, parsing, translation).

The key contributions of this work are as follows. We address context sensitive lemmatization introducing a two-stage bidirectional gated recurrent neural network (BGRNN) architecture. Our model is a supervised one that needs lemma tagged continuous text to learn. Its two most important advantages compared to the state-of-the-art supervised models (Chrupala et al., 2008; Toutanova and Cherry, 2009; Gesmundo and Samardzic, 2012; Müller et al., 2015) are - (i) we do not need to define hand-crafted features such as the word form, presence of special characters, character alignments, surrounding words etc. (ii) parts of speech and other morphological attributes of the surface words are not required for joint learning. Additionally, unknown word forms are also taken care of as the transformation between word-lemma pair is learnt, not the lemma itself. We exploit two steps learning in our method. At first, characters in the words are passed sequentially through a BGRNN to get a syntactic embedding of each word and then the outputs are

[1]The dataset and the code of model architecture are released with the paper. They are also available in http://www.isical.ac.in/~utpal/resources.php

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1481–1491
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1136

combined with the corresponding semantic embeddings. Finally, mapping between the combined embeddings to word-lemma transformations are learnt using another BGRNN.

For the present work, we assess our model on nine languages having diverse morphological variations. Out of them, two (Bengali and Hindi) belong to the Indic languages family and the rests (Catalan, Dutch, Hungarian, Italian, Latin, Romanian and Spanish) are taken from the European languages. To evaluate the proposed model on Bengali, a lemma annotated continuous text has been developed. As so far there is no such standard large dataset for supervised lemmatization in Bengali, the prepared one would surely contribute to the respective NLP research community. For the remaining languages, standard datasets are used for experimentation. Experimental results reveal that our method outperforms Lemming (Müller et al., 2015) and Morfette (Chrupala et al., 2008) on all the languages except Bengali.

1.1 Related Works

Efforts on developing lemmatizers can be divided into two principle categories (i) rule/heuristics based approaches (Koskenniemi, 1984; Plisson et al., 2004) which are usually not portable to different languages and (ii) learning based methods (Chrupala et al., 2008; Toutanova and Cherry, 2009; Gesmundo and Samardzic, 2012; Müller et al., 2015; Nicolai and Kondrak, 2016) requiring prior training dataset to learn the morphological patterns. Again, the later methods can be further classified depending on whether context of the current word is considered or not. Lemmatization without context (Cotterell et al., 2016; Nicolai and Kondrak, 2016) is closer to stemming and not the focus of the present work. It is noteworthy here that the supervised lemmatization methods do not try to classify the lemma of a given word form as it is infeasible due to having a large number of lemmas in a language. Rather, learning the transformation between word-lemma pair is more generalized and it can handle the unknown word forms too. Several representations of word-lemma transformation have been introduced so far such as shortest edit script (SES), label set, edit tree by Chrupala et al. (2008), Gesmundo and Samardzic (2012) and Müller et al. (2015) respectively. Following Müller et al. (2015), we consider lemmatization as the edit tree classification problem. Toutanova and Cherry (2009); Müller et al. (2015) also showed that joint learning of lemmas with other morphological attributes is mutually beneficial but obtaining the gold annotated datasets is very expensive. In contrast, our model needs only lemma annotated continuous text (not POS and other tags) to learn the word morphology.

Since our experiments include the Indic languages also, it would not be an overstatement to say that there have been little efforts on lemmatization so far (Faridee et al., 2009; Loponen and Järvelin, 2010; Paul et al., 2013; Bhattacharyya et al., 2014). The works by Faridee et al. (2009); Paul et al. (2013) are language specific rule based for Bengali and Hindi respectively. (Loponen and Järvelin, 2010)'s primary objective was to improve the retrieval performance. Bhattacharyya et al. (2014) proposed a heuristics based lemmatizer using WordNet but they did not consider context of the target word which is an important basis to lemmatize Indic languages. Chakrabarty and Garain (2016) developed an unsupervised language independent lemmatizer and evaluated it on Bengali. They consider the contextual information but the major disadvantage of their method is dependency on dictionary as well as POS information. Very recently, a supervised neural lemmatization model has been introduced by Chakrabarty et al. (2016). They treat the problem as lemma transduction rather than classification. The particular root in the dictionary is chosen as the lemma with which the transduced vector possesses maximum cosine similarity. Hence, their approach fails when the correct lemma of a word is not present in the dictionary. Besides, the lemmatization accuracy obtained by the respective method is not very significant. Apart from the mentioned works, there is no such commendable effort so far.

Rest of this paper is organized as follows. In section 2, we describe the proposed lemmatization method. Experimental setup and the results are presented in section 3. Finally, in section 4 we conclude the paper.

2 The Proposed Method

As stated earlier in section 1.1, we represent the mapping between a word to its lemma using edit tree (Chrupała, 2008; Müller et al., 2015). An edit tree embeds all the necessary edit operations within it i.e. insertions, deletions and substitutions of strings required throughout the transformation

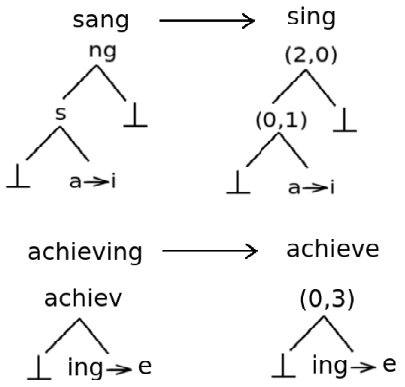


Figure 1: Edit trees for the word-lemma pairs 'sang-sing' and 'achieving-achieve'.

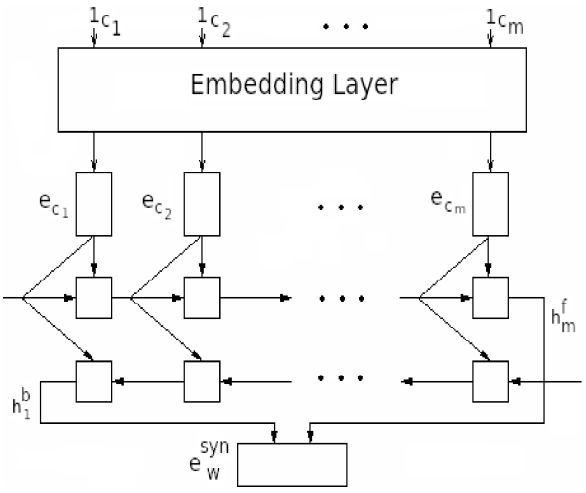


Figure 2: Syntactic vector composition for a word.

process. Figure 1 depicts two edit trees that map the inflected English words *'sang'* and *'achieving'* to their respective lemmas *'sing'* and *'achieve'*. For generalization, edit trees encode only the substitutions and the length of prefixes and suffixes of the longest common substrings. Initially, all unique edit trees are extracted from the associated surface word-lemma pairs present in the training set. The extracted trees refer to the class labels in our model. So, for a test word, the goal is to classify the correct edit tree which, applied on the word, returns the lemma.

Next, we will describe the architecture of the proposed neural lemmatization model. It is evident that for morphologically rich languages, both syntactic and semantic knowledge help in lemmatizing a surface word. Now a days, it is a common practice to embed the functional properties of words into vector representations. Despite the word vectors prove very effectual in semantic processing tasks, they are modelled using the distributional similarity obtained from a raw corpus. Morphological regularities, local and non-local dependencies in character sequences that play deciding roles to find the lemmas, are not taken into account where each word has its own vector interpretation. We address this issue by incorporating two different embeddings into our model. Semantic embedding is achieved using word2vec (Mikolov et al., 2013a,b), which has been empirically found highly successful. To devise the syntactic embedding of a word, we follow the work of Ling et al. (2015) that uses compositional character to word model using bidirectional long-short term memory (BLSTM) network. In our experiments, different gated recurrent cells such as LSTM (Graves, 2013) and GRU (Cho et al., 2014), are explored. The next subsection describes the module to construct the syntactic vectors by feeding the character sequences into BGRNN architecture.

2.1 Forming Syntactic Embeddings

Our goal is to build syntactic embeddings of words that capture the similarities in morphological level. Given an input word w, the target is to obtain a d dimensional vector representing the syntactic structure of w. The procedure is illustrated in Figure 2. At first, an alphabet of characters is defined as C. We represent w as a sequence of characters $c_1, \ldots, c_m$ where m is the word length and each character c_i is defined as a one hot encoded vector $\mathbf{1}_{c_i}$, having one at the index of c_i in the alphabet C. An embedding layer is defined as $\mathbf{E}_c \in \mathbb{R}^{d_c \times |C|}$, that projects each one hot encoded character vector to a d_c dimensional embedded vector. For a character c_i, its projected vector $\mathbf{e}_{c_i}$ is obtained from the embedding layer $\mathbf{E}_c$, using this relation $\mathbf{e}_{c_i} = \mathbf{E}_c \cdot \mathbf{1}_{c_i}$ where '$\cdot$' is the matrix multiplication operation.

Given a sequence of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ as input, a LSTM cell computes the state sequence $\mathbf{h}_1, \ldots, \mathbf{h}_m$ using the following equations:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{V}_f \mathbf{c}_{t-1} + \mathbf{b}_f)$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{V}_i \mathbf{c}_{t-1} + \mathbf{b}_i)$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{V}_o \mathbf{c}_t + \mathbf{b}_o)$$
$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t),$$

Whereas, the updation rules for GRU are as follows

$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z)$$
$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r)$$
$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot tanh(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h),$$

σ denotes the sigmoid function and $\odot$ stands for the element-wise (Hadamard) product. Unlike the simple recurrent unit, LSTM uses an extra memory cell $\mathbf{c}_t$ that is controlled by three gates - input ($\mathbf{i}_t$), forget ($\mathbf{f}_t$) and output ($\mathbf{o}_t$). $\mathbf{i}_t$ controls the amount of new memory content added to the memory cell, $\mathbf{f}_t$ regulates the degree to which the existing memory is forgotten and $\mathbf{o}_t$ finally adjusts the memory content exposure. **W**, **U**, **V** (weight matrices), **b** (bias) are the parameters.

Without having a memory cell like LSTM, a GRU uses two gates namely update ($\mathbf{z}_t$) and reset ($\mathbf{r}_t$). The gate, $\mathbf{z}_t$ decides the amount of update needed for activation and $\mathbf{r}_t$ is used to ignore the previous hidden states (when close to 0, it forgets the earlier computation). So, for a sequence of projected characters $\mathbf{e}_{c_1}, \ldots, \mathbf{e}_{c_m}$, the forward and the backward networks produce the state sequences $\mathbf{h}_1^f, \ldots, \mathbf{h}_m^f$ and $\mathbf{h}_m^b, \ldots, \mathbf{h}_1^b$ respectively. Finally, we obtain the syntactic embedding of w, denoted as $\mathbf{e}_w^{syn}$, by concatenating the final states of these two sequences.

$$\mathbf{e}_w^{syn} = [\mathbf{h}_1^b, \mathbf{h}_m^f]$$

2.2 Model

We present the sketch of the final integrated model in Figure 3. For a word w, let $\mathbf{e}_w^{sem}$ denotes its semantic embedding obtained using word2vec. Both the vectors, $\mathbf{e}_w^{syn}$ and $\mathbf{e}_w^{sem}$ are concatenated together to shape the composite representation $\mathbf{e}_w^{com}$ which carries the morphological and distributional information within it. Firstly, for all the words present in the training set, their composite vectors are generated. Next, they are fed sentence-wise into the next level of BGRNN to train the model for the edit tree classification task. This second level bidirectional network accounts the local context in both forward and backward directions, which is essential for lemmatization in context sensitive languages. Let, $\mathbf{e}_{w_1}^{com}, \ldots, \mathbf{e}_{w_n}^{com}$ be the input sequence of composite vectors to the BGRNN model, representing a sentence having n words $w_1, \ldots, w_n$. For the i^{th} vector $\mathbf{e}_{w_i}^{com}$, $\mathbf{h}_i^f$ and

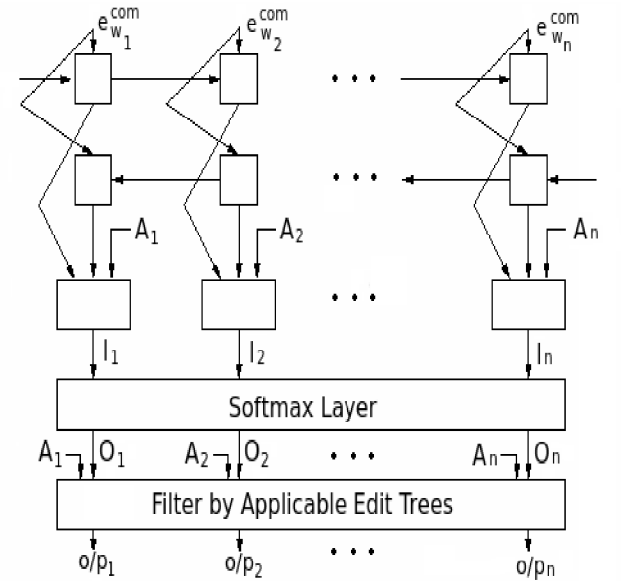


Figure 3: Second level BGRNN model for edit tree classification.

$\mathbf{h}_i^b$ denote the forward and backward states respectively carrying the informations of $w_1, \ldots, w_i$ and $w_i, \ldots, w_n$.

2.2.1 Incorporating Applicable Edit Trees Information

One aspect that we did not look into so far, is that for a word all unique edit trees extracted from the training set are not applicable as this would lead to incompatible substitutions. For example, the edit tree for the word-lemma pair 'sang-sing' depicted in Figure 1, cannot be applied on the word 'achieving'. This information is prior before training the model i.e. for any arbitrary word, we can sort out the subset of unique edit trees from the training samples in advance, which are applicable on it. In general, if all the unique edit trees in the training data are set as the class labels, the model will learn to distribute the probability mass over all the classes which is a clear-cut bottleneck. In order to alleviate this problem, we take a novel strategy so that for individual words in the input sequence, the model will learn, to which classes, the output probability should be apportioned.

Let $T = \{t_1, \ldots, t_k\}$ be the set of distinct edit trees found in the training set. For the word w_i in the input sequence $w_1, \ldots, w_n$, we define its applicable edit trees vector as $\mathbf{A}_i = (a_i^1, \ldots, a_i^k)$ where $\forall j \in \{1, \ldots, k\}$, $a_i^j = 1$ if t_j is applicable for w_i, otherwise 0. Hence, $\mathbf{A}_i$ holds the information regarding the set of edit trees to concentrate

upon, while processing the word w_i. We combine $\mathbf{A}_i$ together with $\mathbf{h}_i^f$ and $\mathbf{h}_i^b$ for the final classification task as following,

$$\mathbf{l}_i = softplus(\mathbf{L}^f \mathbf{h}_i^f + \mathbf{L}^b \mathbf{h}_i^b + \mathbf{L}^a \mathbf{A}_i + \mathbf{b}_l),$$

where 'softplus' denotes the activation function $f(x) = ln(1 + e^x)$ and $\mathbf{L}^f, \mathbf{L}^b, \mathbf{L}^a$ and $\mathbf{b}_l$ are the parameters trained by the network. At the end, $\mathbf{l}_i$ is passed through the softmax layer to get the output labels for w_i.

To pick the correct edit tree from the output of the softmax layer, we exploit the prior information $\mathbf{A}_i$. Instead of choosing the class that gets the maximum probability, we select the maximum over the classes corresponding to the applicable edit trees. The idea is expressed as follows. Let $\mathbf{O}_i = (o_i^1, \ldots, o_i^k)$ be the output of the softmax layer. Instead of opting for the maximum over $o_i^1, \ldots, o_i^k$ as the class label, the highest probable class out of those corresponding to the applicable edit trees, is picked up. That is, the particular edit tree $t_j \in T$ is considered as the right candidate for w_i, where

$$j = \text{argmax}_{j' \in \{1,\ldots,k\} \wedge a_i^{j'}=1} \, o_i^{j'}$$

In this way, we cancel out the non-applicable classes and focus only on the plausible candidates.

3 Experimentation

Out of the nine reference languages, initially we choose four of them (Bengali, Hindi, Latin and Spanish) for in-depth analysis. We conduct an exhaustive set of experiments - such as determining the direct lemmatization accuracy, accuracy obtained without using applicable edit trees in training, measuring the model's performance on the unseen words etc. on these four languages. Later we consider five more languages (Catalan, Dutch, Hungarian, Italian and Romanian) mostly for testing the generalization ability of the proposed method. For these additional languages, we present only the lemmatization accuracy in section 3.2.

Datasets: As Bengali is a low-resourced language, a relatively large lemma annotated dataset is prepared for the present work using Tagore's short stories collection[2] and randomly selected news articles from miscellaneous domains. One linguist took around 2 months to complete the annotation which was checked by another person and differences were sorted out. Out of the 91 short stories of Tagore, we calculate the value of (# tokens / # distinct tokens) for each story. Based on this value (lower is better), top 11 stories are selected. The news articles[3] are crafted from the following domains: animal, archaeology, business, country, education, food, health, politics, psychology, science and travelogue. In Hindi, we combine the COLING'12 shared task data for dependency parsing and Hindi WSD health and tourism corpora[4] (Khapra et al., 2010) together[5]. For Latin, the data is taken from the PROIEL treebank (Haug and Jøhndal, 2008) and for Spanish, we merge the training and development datasets of CoNLL'09 (Hajič et al., 2009) shared task on syntactic and semantic dependencies. The dataset statistics are given in Table 1. We assess the lemmatization performance by measuring the direct accuracy which is the ratio of the number of correctly lemmatized words to the total number of input words. The experiments are performed using 4 fold cross validation technique i.e. the datasets are equi-partitioned into 4 parts at sentence level and then each part is tested exactly once using the model trained on the remaining 3 parts. Finally, we report the average accuracy over 4 fold.

	# Sentences	# Word Tokens
Bengali	1,702	20,257
Hindi	36,143	819,264
Latin	15,002	165,634
Spanish	15,984	477,810

Table 1: Dataset statistics of the 4 languages.

Induction of Edit Tree Set: Initially, distinct edit trees are induced from the word-lemma pairs present in the training set. Next, the words in the training data are annotated with their corresponding edit trees. Training is accomplished on this edit tree tagged text. Figure 4 plots the growth of the edit tree set against the number of word-lemma samples in the four languages. With the increase of samples, the size of edit tree set gradually converges revealing the fact that most of the frequent transformation patterns (both regular and irregular) are covered by the induction process. From

[2] www.rabindra-rachanabali.nltr.org

[3] http://www.anandabazar.com/

[4] http://www.cfilt.iitb.ac.in/wsd/annotated_corpus/

[5] We also release the Hindi dataset with this paper as it is a combination of two different datasets.

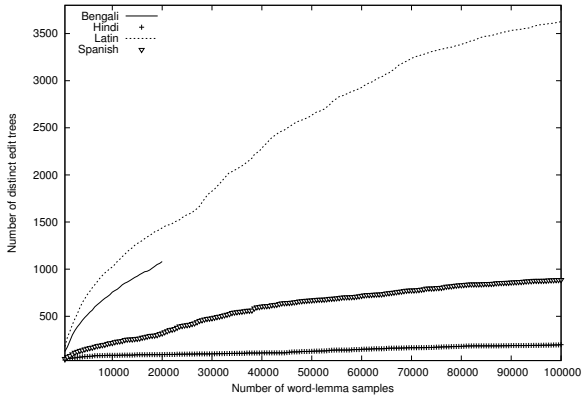


Figure 4: Increase of the edit tree set size with the number of word-lemma samples.

Figure 4, morphological richness can be compared across the languages. When convergence happens quickly i.e. at relatively less number of samples, it evidences that the language is less complex. Among the four reference languages, Latin stands out as the most intricate, followed by Bengali, Spanish and Hindi.

Semantic Embeddings: We obtain the distributional word vectors for Bengali and Hindi by training the word2vec model on FIRE Bengali and Hindi news corpora[6]. Following the work by Mikolov et al. (2013a), continuous-bag-of-words architecture with negative sampling is used to get 200 dimensional word vectors. For Latin and Spanish, we use the embeddings released by Bamman and Smith (2012)[7] and Cardellino (2016)[8] respectively.

Syntactic Representation: We acquire the statistics of word length versus frequency from the datasets and find out that irrespective of the languages, longer words (have more than 20-25 characters) are few in numbers. Based on this finding, each word is limited to a sequence of 25 characters. Smaller words are padded null characters at the end and for the longer words, excess characters are truncated out. So, each word is represented as a 25 length array of one hot encoded vectors which is given input to the embedding layer that works as a look up table producing an equal length array of embedded vectors. Initialization of the embedding layer is done randomly and the embedded vector dimension is set to 10. Eventually, the output of the embedding layer is passed to the first level BGRNN for learning the syntactic representation.

Hyper Parameters: There are several hyper parameters in our model such as the number of neurons in the hidden layer ($\mathbf{h}_t$) of both first and second level BGRNN, learning mode, number of epochs to train the models, optimization algorithm, dropout rate etc. We experiment with different settings of these parameters and report where optimum results are achieved. For both the bidirectional networks, number of hidden layer neurons is set to 64. Online learning is applied for updation of the weights. Number of epochs varies across languages to converge the training. It is maximum for Bengali (around 80 epochs), followed by Latin, Spanish and Hindi taking around 50, 35 and 15 respectively. Throughout the experiments, we set the dropout rate as 0.2 to prevent over-fitting. Different optimization algorithms like AdaDelta (Zeiler, 2012), Adam (Kingma and Ba, 2014), RMSProp (Dauphin et al., 2015) are explored. Out of them, Adam yields the best result. We use the categorical cross-entropy as the loss function in our model.

Baselines: We compare our method with Lemming[9] and Morfette[10]. Both the model jointly learns lemma and other morphological tags in context. Lemming uses a 2nd-order linear-chain CRF to predict the lemmas whereas, the current version of Morfette is based on structured perceptron learning. As POS information is a compulsory requirement of these two models, the Bengali data is manually POS annotated. For the other languages, the tags were already available. Although this comparison is partially biased as the proposed method does not need POS information, but the experimental results show the effectiveness of our model. There is an option in Lemming and Morfette to provide an exhaustive set of root words which is used to exploit the dictionary features i.e. to verify if a candidate lemma is a valid form or not. To make the comparisons consistent, we do not exploit any external dictionary in our experiments.

3.1 Results

The lemmatization results are presented in Table 2. We explore our proposed model with two types of gated recurrent cells - LSTM and GRU. As there

[6] http://fire.irsi.res.in/fire

[7] http://www.cs.cmu.edu/~dbamman/latin.html

[8] http://crscardellino.me/SBWCE/

[9] http://cistern.cis.lmu.de/lemming/

[10] https://github.com/gchrupala/morfette

	Bengali	Hindi	Latin	Spanish
BLSTM-BLSTM	90.84/91.14	94.89/**94.90**	89.35/89.52	97.85/97.91
BGRU-BGRU	90.63/90.84	94.44/94.50	89.40/**89.59**	98.07/**98.11**
Lemming	**91.69**	91.64	88.50	93.12
Morfette	90.69	90.57	87.10	92.90

Table 2: Lemmatization accuracy (in %) without/with restricting output classes.

	Bengali	Hindi	Latin	Spanish
BLSTM-BLSTM	86.46/89.52	94.34/94.52	85.70/87.35	97.39/97.62
BGRU-BGRU	86.39/88.90	93.84/94.04	85.49/86.87	97.51/97.73

Table 3: Lemmatization accuracy (in %) without using applicable edit trees in training.

are two successive bidirectional networks - the first one for building the syntactic embedding and the next one for the edit tree classification, so basically we deal with two different models BLSTM-BLSTM and BGRU-BGRU. Table 2 shows the comparison results of these models with Lemming and Morfette. In all cases, the average accuracy over 4 fold cross validation on the datasets is reported. For an entry 'x/y' in Table 2, x denotes the accuracy without output classes restriction, i.e. taking the maximum over all edit tree classes present in the training set, whereas y refers to the accuracy when output is restricted in only the applicable edit tree classes of the input word. Except for Bengali, the proposed models outperform the baselines for the other three languages. In Hindi, BLSTM-BLSTM gives the best result (94.90%). For Latin and Spanish, the highest accuracy is achieved by BGRU-BGRU (89.59% and 98.11% respectively). In the Bengali dataset, Lemming produces the optimum result (91.69%) beating its closest performer BLSTM-BLSTM by 0.55%. It is to note that the training set size in Bengali is smallest compared to the other languages (on average, 16,712 tokens in each of the 4 folds). Overall, BLSTM-BLSTM and BGRU-BGRU perform equally good. For Bengali and Hindi, the former model is better and for Latin and Spanish, the later yields more accuracy. Throughout the experiments, restricting the output over applicable classes improves the performance significantly. The maximum improvements we get are: 0.30% in Bengali using BLSTM-BLSTM (from 90.84% to 91.14%), 0.06% in Hindi using BGRU-BGRU (from 94.44% to 94.50%), 0.19% in Latin using BGRU-BGRU (from 89.40% to 89.59%) and 0.06% in Spanish using BLSTM-BLSTM (from 97.85% to 97.91%). To compare between the two baselines, Lemming consistently performs better than Morfette (the maximum difference between their accuracies is 1.40% in Latin).

Bengali	Hindi	Latin	Spanish
27.17	5.25	15.74	7.54

Table 4: Proportion of unknown word forms (in %) present in the test sets.

Effect of Training without Applicable Edit Trees: We also explore the impact of applicable edit trees in training. To see the effect, we train our model without giving the applicable edit trees information as input. In the model design, the equation for the final classification task is changed as follows,

$$\mathbf{l}_i = softplus(\mathbf{L}^f \mathbf{h}_i^f + \mathbf{L}^b \mathbf{h}_i^b + \mathbf{b}_l),$$

The results are presented in Table 3. Except for Spanish, BLSTM-BLSTM outperforms BGRU-BGRU in all the other languages. As compared with the results in Table 2, for every model, training without applicable edit trees degrades the lemmatization performance. In all cases, BGRU-BGRU model gets more affected than BLSTM-BLSTM. Language-wise, the drops in its accuracy are: 1.94% in Bengali (from 90.84% to 88.90%), 0.46% in Hindi (from 94.50% to 94.04%), 2.72% in Latin (from 89.59% to 86.87%) and 0.38% in Spanish (from 98.11% to 97.73%).

One important finding to note in Table 3 is that irrespective of any particular language and model used, the amount of increase in accuracy due to the output restriction on the applicable classes is much more than that observed in Table 2. For instance, in Table 2 the accuracy improvement for Bengali using BLSTM-BLSTM is 0.30% (from 90.84% to 91.14%), whereas in Table 3 the corresponding value is 3.06% (from 86.46% to 89.52%). These outcomes signify the fact that training with the ap-

	Bengali	Hindi	Latin	Spanish
BLSTM-BLSTM	71.06/72.10	87.80/88.18	60.85/61.63	88.06/88.79
BGRU-BGRU	70.44/71.22	88.34/88.40	60.65/61.52	91.48/92.25
Lemming	74.10	90.35	57.19	58.89
Morfette	70.27	88.59	47.41	57.61

Table 5: Lemmatization accuracy (in %) on unseen words.

	Bengali	Hindi	Latin	Spanish
BLSTM-BLSTM	56.16/66.26	87.42/88.41	49.80/56.05	86.22/87.97
BGRU-BGRU	59.45/66.84	87.19/88.26	50.24/55.35	86.74/88.49

Table 6: Lemmatization accuracy (in %) on unseen words without using applicable edit trees in training.

plicable edit trees already learns to dispense the output probability to the legitimate classes over which, output restriction cannot yield much enhancement.

Results for Unseen Word Forms: Next, we discuss about the lemmatization performance on those words which were absent in the training set. Table 4 shows the proportion of unseen forms averaged over 4 folds on the datasets. In Table 5, we present the accuracy obtained by our models and the baselines. For Bengali and Hindi, Lemming produces the best results (74.10% and 90.35%). For Latin and Spanish, BLSTM-BLSTM and BGRU-BGRU obtain the highest accuracy (61.63% and 92.25%) respectively. In Spanish, our model gets the maximum improvement over the baselines. BGRU-BGRU beats Lemming with 33.36% margin (on average, out of 9,011 unseen forms, 3,005 more tokens are correctly lemmatized). Similar to the results in Table 2, the results in Table 5 evidences that restricting the output in applicable classes enhances the lemmatization performance. The maximum accuracy improvements due to the output restriction are: 1.04% in Bengali (from 71.06% to 72.10%), 0.38% in Hindi (from 87.80% to 88.18%) using BLSTM-BLSTM and 0.87% in Latin (from 60.65% to 61.52%), 0.77% in Spanish (from 91.48% to 92.25%) using BGRU-BGRU.

Further, we investigate the performance of our models trained without the applicable edit trees information, on the unseen word forms. The results are given in Table 6. As expected, for every model, the accuracy drops compared to the results shown in Table 5. The only exception that we find out is in the entry for Hindi with BLSTM-BLSTM. Though without restricting the output, the accuracy in Table 5 (87.80%) is higher than the corresponding value in Table 6 (87.42%), but after output restriction, the performance changes (88.18% in Table 5, 88.41% in Table 6) which reveals that only selecting the maximum probable class over the applicable ones would be a better option for the unseen word forms in Hindi.

	Sem. Embedding	Syn. Embedding
Bengali	90.76/91.02	86.61/86.82
Hindi	94.86/94.86	91.24/91.25
Latin	88.90/89.09	85.31/85.49
Spanish	97.95/98	96.07/96.10

Table 7: Results (in %) obtained using semantic and syntactic embeddings separately.

	# Sentences	# Word Tokens
Catalan	14,832	474,069
Dutch	13,050	197,925
Hungarian	1,351	31,584
Italian	13,402	282,611
Romanian	8,795	202,187

Table 8: Dataset statistics of the 5 additional languages.

Effects of Semantic and Syntactic Embeddings in Isolation: To understand the impact of the combined word vectors on the model's performance, we measure the accuracy experimenting with each one of them separately. While using the semantic embedding, only distributional word vectors are used for edit tree classification. On the other hand, to test the effect of the syntactic embedding exclusively, output from the character level recurrent network is fed to the second level BGRNN. We present the results in Table 7. For Bengali and Hindi, experiments are carried out with the BLSTM-BLSTM model as it gives better results for these languages compared to BGRU-BGRU (given in Table 2). Similarly for Latin and Spanish, the results obtained from BGRU-BGRU are reported. From the outcome of these experiments, use of semantic vec-

	Catalan	Dutch	Hungarian	Italian	Romanian
BLSTM-BLSTM	97.93/**97.95**	93.20/**93.44**	91.03/**91.46**	96.06/**96.09**	94.25/**94.32**
Lemming	89.80	86.95	87.95	92.51	93.34
Morfette	89.46	86.62	86.52	92.02	94.13

Table 9: Lemmatization accuracy (in %) for the 5 languages.

tor proves to be more effective than the character level embedding. However, to capture the distributional properties of words efficiently, a huge corpus is needed which may not be available for low resourced languages. In that case, making use of syntactic embedding is a good alternative. Nonetheless, use of both types of embedding together improves the result.

3.2 Experimental Results for Another Five Languages

As mentioned earlier, five additional languages (Catalan, Dutch, Hungarian, Italian and Romanian) are considered to test the generalization ability of the method. The datasets are taken from the UD Treebanks[11] (Nivre et al., 2017). For each language, we merge the training and development data together and perform 4 fold cross validation on it to measure the average accuracy. The dataset statistics are shown in Table 8. For experimentation, we use the pre-trained semantic embeddings released by (Bojanowski et al., 2016). Only BLSTM-BLSTM model is explored and it is compared with Lemming and Morfette. The hyper parameters are kept same as described previously except for the number of epochs needed for training across the languages. We present the results in Table 9. For all the languages, BLSTM-BLSTM outperforms Lemming and Morfette. The maximum improvement over the baselines we get is for Catalan (beats Lemming and Morfette by 8.15% and 8.49% respectively). Similar to the results in Table 2, restricting the output over applicable classes yields consistent performance improvement.

4 Conclusion

This article presents a neural network based context sensitive lemmatization method which is language independent and supervised in nature. The proposed model learns the transformation patterns between word-lemma pairs and hence, can handle the unknown word forms too. Additionally, it does not rely on human defined features and various morphological tags except the gold lemma annotated continuous text. We explore different variations of the model architecture by changing the type of recurrent units. For evaluation, nine languages are taken as the references. Except Bengali, the proposed method outperforms the state-of-the-art models (Lemming and Morfette) on all the other languages. For Bengali, it produces the second best performance (91.14% using BLSTM-BLSTM). We measure the accuracy on the partial data (keeping the data size comparable to the Bengali dataset) for Hindi, Latin and Spanish to check the effect of the data amount on the performance. For Hindi, the change in accuracy is insignificant but for Latin and Spanish, accuracy drops by 3.50% and 6% respectively. The time requirement of the proposed method is also analyzed. Training time depends on several parameters such as size of the data, number of epochs required for convergence, configuration of the system used etc. In our work, we use the 'keras' software keeping 'theano' as backend. The codes were run on a single GPU (Nvidia GeForce GTX 960, 2GB memory). Once trained, the model takes negligible time to predict the appropriate edit trees for test words (e.g. 844 and 930 words/second for Bengali and Hindi respectively). We develop a Bengali lemmatization dataset which is definitely a notable contribution to the language resources. From the present study, one important finding comes out that for the unseen words, the lemmatization accuracy drops by a large margin in Bengali and Spanish, which may be the area of further research work. Apart from it, we intend to propose a neural architecture that accomplishes the joint learning of lemmas with other morphological attributes.

[11] http://universaldependencies.org/

References

David Bamman and David Smith. 2012. Extracting two thousand years of latin from a million book library. *J. Comput. Cult. Herit.* 5(1):2:1–2:13. https://doi.org/10.1145/2160165.2160167.

Pushpak Bhattacharyya, Ankit Bahuguna, Lavita Talukdar, and Bornali Phukan. 2014. Facilitating multi-lingual sense annotation: Human mediated

lemmatizer. In Heili Orav, Christiane Fellbaum, and Piek Vossen, editors, *Proceedings of the Seventh Global Wordnet Conference*. Tartu, Estonia, pages 224–231. http://www.aclweb.org/anthology/W14-0130.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* https://arxiv.org/abs/1607.04606.

Cristian Cardellino. 2016. Spanish Billion Words Corpus and Embeddings. http://crscardellino.me/SBWCE/.

Abhisek Chakrabarty, Akshay Chaturvedi, and Utpal Garain. 2016. A neural lemmatizer for bengali. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France, pages 2558–2561. http://www.lrec-conf.org/proceedings/lrec2016/pdf/955$_{P}aper.pdf$.

Abhisek Chakrabarty and Utpal Garain. 2016. Benlem (a bengali lemmatizer) and its role in wsd. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 15(3):12:1–12:18. https://doi.org/10.1145/2835494.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* https://arxiv.org/abs/1409.1259.

Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University. http://doras.dcu.ie/550/.

Grzegorz Chrupala, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with morfette. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA), Marrakech, Morocco. http://www.lrec-conf.org/proceedings/lrec2008/pdf/594$_{p}aper.pdf$.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The sigmorphon 2016 shared task—morphological reinflection. In *Proceedings of the 2016 Meeting of SIGMORPHON*. Association for Computational Linguistics, Berlin, Germany. http://aclweb.org/anthology/sigmorphon.html.

Yann N. Dauphin, Harm de Vries, and Yoshua Bengio. 2015. Equilibrated adaptive learning rates for non-convex optimization. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 1504–1512. http://dl.acm.org/citation.cfm?id=2969239.2969407.

Abu Zaher Md Faridee, Francis M Tyers, et al. 2009. Development of a morphological analyser for bengali. In *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation*. Universidad de Alicante. Departamento de Lenguajes y Sistemas Informáticos, pages 43–50. http://www.mt-archive.info/FreeRBMT-2009-Faridee.pdf.

Andrea Gesmundo and Tanja Samardzic. 2012. Lemmatisation as a tagging task. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Jeju Island, Korea, pages 368–372. http://www.aclweb.org/anthology/P12-2072.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* https://arxiv.org/abs/1308.0850.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*. Association for Computational Linguistics, Boulder, Colorado, pages 1–18. http://www.aclweb.org/anthology/W09-1201.

Dag TT Haug and Marius Jøhndal. 2008. Creating a parallel treebank of the old indo-european bible translations. In *Proceedings of the Second Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2008)*. pages 27–34.

Mitesh Khapra, Anup Kulkarni, Saurabh Sohoney, and Pushpak Bhattacharyya. 2010. All words domain adapted wsd: Finding a middle ground between supervision and unsupervision. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1532–1541. http://www.aclweb.org/anthology/P10-1155.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* https://arxiv.org/abs/1412.6980.

Kimmo Koskenniemi. 1984. A general computational model for word-form recognition and production. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stanford, California, USA, pages 178–181. https://doi.org/10.3115/980491.980529.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form:

Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1520–1530. http://aclweb.org/anthology/D15-1176.

Aki Loponen and Kalervo Järvelin. 2010. A dictionary and corpus independent statistical lemmatizer for information retrieval in low resource languages. In *Multilingual and Multimodal Information Access Evaluation*, Springer, pages 3–14. https://doi.org/10.1007/978-3-642-15998-5_3.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. Curran Associates Inc., USA, NIPS'13, pages 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 746–751. http://www.aclweb.org/anthology/N13-1090.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2268–2274. http://aclweb.org/anthology/D15-1272.

Garrett Nicolai and Grzegorz Kondrak. 2016. Leveraging inflection tables for stemming and lemmatization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1138–1147. http://www.aclweb.org/anthology/P16-1108.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çar Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà M, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phng Lê Hng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11234/1-1983.

Snigdha Paul, Nisheeth Joshi, and Iti Mathur. 2013. Development of a hindi lemmatizer. *International Journal of Computational Linguistics and Natural Language Processing* 2(5):380–384. https://arxiv.org/abs/1305.6211.

Joël Plisson, Nada Lavrac, Dunja Mladenic, et al. 2004. A rule based approach to word lemmatization. *Proceedings of IS-2004* pages 83–86.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, pages 486–494. http://aclweb.org/anthology/P/P09/P09-1055.pdf.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* https://arxiv.org/abs/1212.5701.

Learning to Create and Reuse Words in Open-Vocabulary Neural Language Modeling

Kazuya Kawakami[♠] **Chris Dyer**[♣] **Phil Blunsom**[♠♣]
[♠]Department of Computer Science, University of Oxford, Oxford, UK
[♣]DeepMind, London, UK
{kazuya.kawakami,phil.blunsom}@cs.ox.ac.uk,cdyer@google.com

Abstract

Fixed-vocabulary language models fail to account for one of the most characteristic statistical facts of natural language: the frequent creation and reuse of new word types. Although character-level language models offer a partial solution in that they can create word types not attested in the training corpus, they do not capture the "bursty" distribution of such words. In this paper, we augment a hierarchical LSTM language model that generates sequences of word tokens character by character with a caching mechanism that learns to reuse previously generated words. To validate our model we construct a new open-vocabulary language modeling corpus (the Multilingual Wikipedia Corpus; MWC) from comparable Wikipedia articles in 7 typologically diverse languages and demonstrate the effectiveness of our model across this range of languages.

1 Introduction

Language modeling is an important problem in natural language processing with many practical applications (translation, speech recognition, spelling autocorrection, etc.). Recent advances in neural networks provide strong representational power to language models with distributed representations and unbounded dependencies based on recurrent networks (RNNs). However, most language models operate by generating words by sampling from a closed vocabulary which is composed of the most frequent words in a corpus. Rare tokens are typically replaced by a special token, called the unknown word token, ⟨UNK⟩. Although fixed-vocabulary language models have some important practical applications and are appealing models for study, they fail to capture two empirical facts about the distribution of words in natural languages. First, vocabularies keep growing as the number of documents in a corpus grows: new words are constantly being created (Heaps, 1978). Second, rare and newly created words often occur in "bursts", i.e., once a new or rare word has been used once in a document, it is often repeated (Church and Gale, 1995; Church, 2000).

The open-vocabulary problem can be solved by dispensing with word-level models in favor of models that predict sentences as sequences of characters (Sutskever et al., 2011; Chung et al., 2017). Character-based models are quite successful at learning what (new) word forms look like (e.g., they learn a language's orthographic conventions that tell us that *sustinated* is a plausible English word and *bzoxqir* is not) and, when based on models that learn long-range dependencies such as RNNs, they can also be good models of how words fit together to form sentences.

However, existing character-sequence models have no explicit mechanism for modeling the fact that once a rare word is used, it is likely to be used again. In this paper, we propose an extension to character-level language models that enables them to reuse previously generated tokens (§2). Our starting point is a hierarchical LSTM that has been previously used for modeling sentences (word by word) in a conversation (Sordoni et al., 2015), except here we model words (character by character) in a sentence. To this model, we add a caching mechanism similar to recent proposals for caching that have been advocated for closed-vocabulary models (Merity et al., 2017; Grave et al., 2017). As word tokens are generated, they are placed in an LRU cache, and, at each time step the model decides whether to copy a previously generated word from the cache or to generate it from scratch, character by character. The decision of whether

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1492–1502
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1137

to use the cache or not is a latent variable that is marginalised during learning and inference. In summary, our model has three properties: it creates new words, it accounts for their burstiness using a cache, and, being based on LSTM s over word representations, it can model long range dependencies.

To evaluate our model, we perform ablation experiments with variants of our model without the cache or hierarchical structure. In addition to standard English data sets (PTB and WikiText-2), we introduce a new multilingual data set: the Multilingual Wikipedia Corpus (MWC), which is constructed from comparable articles from Wikipedia in 7 typologically diverse languages (§3) and show the effectiveness of our model in all languages (§4). By looking at the posterior probabilities of the generation mechanism (language model vs. cache) on held-out data, we find that the cache is used to generate "bursty" word types such as proper names, while numbers and generic content words are generated preferentially from the language model (§5).

2 Model

In this section, we describe our hierarchical character language model with a word cache. As is typical for RNN language models, our model uses the chain rule to decompose the problem into incremental predictions of the next word conditioned on the history:

$$p(\boldsymbol{w}) = \prod_{t=1}^{|\boldsymbol{w}|} p(w_t \mid \boldsymbol{w}_{<t}).$$

We make two modifications to the traditional RNN language model, which we describe in turn. First, we begin with a cache-less model we call the hierarchical character language model (HCLM; §2.1) which generates words as a sequence of characters and constructs a "word embedding" by encoding a character sequence with an LSTM (Ling et al., 2015). However, like conventional closed-vocabulary, word-based models, it is based on an LSTM that conditions on words represented by fixed-length vectors.[1]

The HCLM has no mechanism to reuse words that it has previously generated, so new forms will only be repeated with very low probability. However, since the HCLM is not merely generating sentences as a sequence of characters, but also segmenting them into words, we may add a word-based cache to which we add words keyed by the hidden state being used to generate them (§2.2). This cache mechanism is similar to the model proposed by Merity et al. (2017).

Notation. Our model assigns probabilities to sequences of words $\boldsymbol{w} = w_1, \ldots, w_{|\boldsymbol{w}|}$, where $|\boldsymbol{w}|$ is the length, and where each word w_i is represented by a sequence of characters $\boldsymbol{c}_i = c_{i,1}, \ldots, c_{i,|\boldsymbol{c}_i|}$ of length $|\boldsymbol{c}_i|$.

2.1 Hierarchical Character-level Language Model (HCLM)

This hierarchical model satisfies our linguistic intuition that written language has (at least) two different units, characters and words.

The HCLM consists of four components, three LSTMs (Hochreiter and Schmidhuber, 1997): a character encoder, a word-level context encoder, and a character decoder (denoted LSTM_{enc}, LSTM_{ctx}, and LSTM_{dec}, respectively), and a softmax output layer over the character vocabulary. Fig. 1 illustrates an unrolled HCLM.

Suppose the model reads word w_{t-1} and predicts the next word w_t. First, the model reads the character sequence representing the word $w_{t-1} = c_{t-1,1}, \ldots, c_{t-1,|\boldsymbol{c}_{t-1}|}$ where $|\boldsymbol{c}_{t-1}|$ is the length of the word generated at time $t-1$ in characters. Each character is represented as a vector $\mathbf{v}_{c_{t-1,1}}, \ldots, \mathbf{v}_{c_{t-1,|\boldsymbol{c}_{t-1}|}}$ and fed into the encoder LSTM_{enc} . The final hidden state of the encoder LSTM_{enc} is used as the vector representation of the previously generated word w_{t-1},

$$\mathbf{h}_t^{enc} = \text{LSTM}_{enc}(\mathbf{v}_{c_{t-1,1}}, \ldots, \mathbf{v}_{c_{t-1,|\boldsymbol{c}_t|}}).$$

Then all the vector representations of words $(\mathbf{v}_{w_1}, \ldots, \mathbf{v}_{w_{|\boldsymbol{w}|}})$ are processed with a context LSTM_{ctx} . Each of the hidden states of the context LSTM_{ctx} are considered representations of the history of the word sequence.

$$\mathbf{h}_t^{ctx} = \text{LSTM}_{ctx}(\mathbf{h}_1^{enc}, \ldots, \mathbf{h}_t^{enc})$$

Finally, the initial state of the decoder LSTM is set to be $\mathbf{h}_t^{ctx}$ and the decoder LSTM reads a vector representation of the start symbol $\mathbf{v}_{\langle \text{S} \rangle}$ and generates the next word w_{t+1} character by character. To predict the j-th character in w_t, the decoder

[1] The HCLM is an adaptation of the hierarchical recurrent encoder-decoder of Sordoni et al. (2015) which was used to model dialog as a sequence of actions sentences which are themselves sequences of words. The original model was proposed to compose words into query sequences but we use it to compose characters into word sequences.

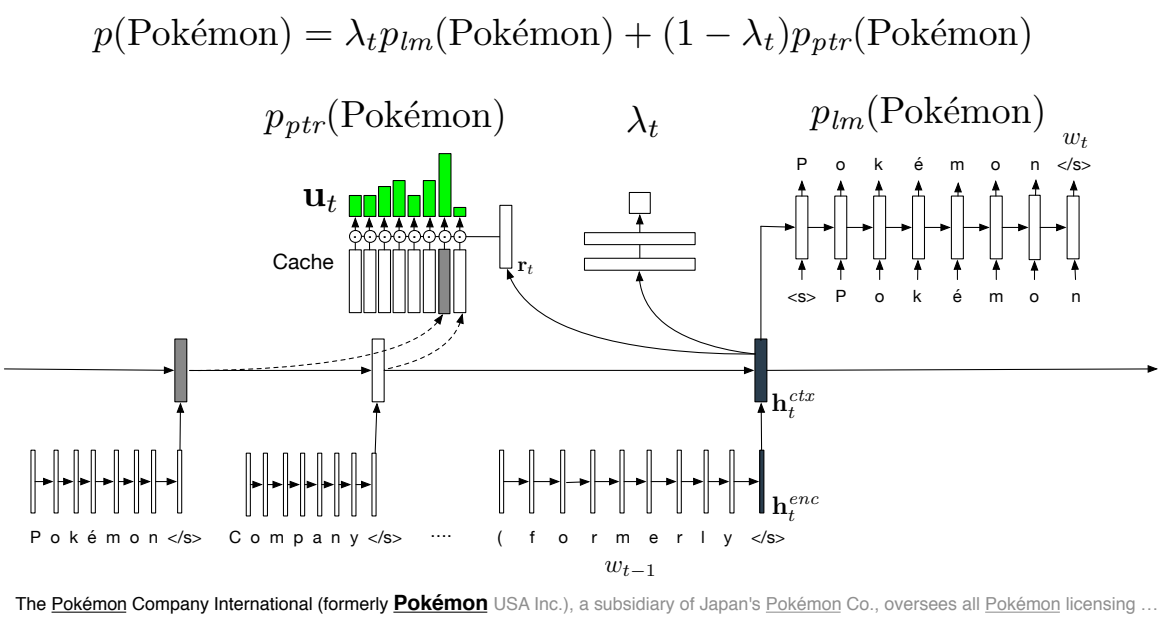


Figure 1: Description of Hierarchical Character Language Model with Cache.

LSTM reads vector representations of the previous characters in the word, conditioned on the context vector $\mathbf{h}_t^{ctx}$ and a start symbol.

$$\mathbf{h}_{t,j}^{dec} = \text{LSTM}_{dec}(\mathbf{v}_{c_{t,1}}, \ldots, \mathbf{v}_{c_{t,j-1}}, \mathbf{h}_t^{ctx}, \mathbf{v}_{\langle s \rangle}).$$

The character generation probability is defined by a softmax layer for the corresponding hidden representation of the decoder LSTM .

$$p(c_{t,j} \mid \boldsymbol{w}_{<t}, \boldsymbol{c}_{t,<j}) = \text{softmax}(\mathbf{W}_{dec}\mathbf{h}_{t,j}^{dec} + \mathbf{b}_{dec})$$

Thus, a word generation probability from HCLM is defined as follows.

$$p_{lm}(w_t \mid \boldsymbol{w}_{<t}) = \prod_{j=1}^{|\boldsymbol{c}_t|} p(c_{t,j} \mid \boldsymbol{w}_{<t}, \boldsymbol{c}_{t,<j})$$

2.2 Continuous cache component

The cache component is an external memory structure which store K elements of recent history. Similarly to the memory structure used in Grave et al. (2017), a word is added to a key-value memory after each generation of w_t. The key at position $i \in [1, K]$ is $\mathbf{k}_i$ and its value m_i. The memory slot is chosen as follows: if the w_t exists already in the memory, its key is updated (discussed below). Otherwise, if the memory is not full, an empty slot is chosen or the least recently used slot is overwritten. When writing a new word to memory, the key is the RNN representation that was used to generate the word ($\mathbf{h}_t$) and the value is the word itself (w_t). In the case when the word already exists in the cache at some position i, the $\mathbf{k}_i$ is updated to be the arithmetic average of $\mathbf{h}_t$ and the existing $\mathbf{k}_i$.

To define the copy probability from the cache at time t, a distribution over copy sites is defined using the attention mechanism of Bahdanau et al. (2015). To do so, we construct a query vector ($\mathbf{r}_t$) from the RNN's current hidden state $\mathbf{h}_t$,

$$\mathbf{r}_t = \tanh(\mathbf{W}_q\mathbf{h}_t + \mathbf{b}_q),$$

then, for each element i of the cache, a 'copy score,' $u_{i,t}$ is computed,

$$u_{i,t} = \mathbf{v}^T \tanh(\mathbf{W}_u\mathbf{k}_i + \mathbf{r}_t).$$

Finally, the probability of generating a word via the copying mechanism is:

$$p_{mem}(i \mid \mathbf{h}_t) = \text{softmax}_i(\mathbf{u}_t)$$
$$p_{ptr}(w_t \mid \mathbf{h}_t) = p_{mem}(i \mid \mathbf{h}_t)[m_i = w_t],$$

where $[m_i = w_t]$ is 1 if the ith value in memory is w_t and 0 otherwise. Since p_{mem} defines a distribution of slots in the cache, p_{ptr} translates it into word space.

2.3 Character-level Neural Cache Language Model

The word probability $p(w_t \mid \boldsymbol{w}_{<t})$ is defined as a mixture of the following two probabilities. The first

one is a language model probability, $p_{lm}(w_t \mid \boldsymbol{w}_{<t})$ and the other is pointer probability , $p_{ptr}(w_t \mid \boldsymbol{w}_{<t})$. The final probability $p(w_t \mid \boldsymbol{w}_{<t})$ is

$$\lambda_t p_{lm}(w_t \mid \boldsymbol{w}_{<t}) + (1 - \lambda_t) p_{ptr}(w_t \mid \boldsymbol{w}_{<t}),$$

where λ_t is computed by a multi-layer perceptron with two non-linear transformations using $\mathbf{h}_t$ as its input, followed by a transformation by the logistic sigmoid function:

$$\gamma_t = \text{MLP}(\mathbf{h}_t), \qquad \lambda_t = \frac{1}{1 - e^{-\gamma_t}}.$$

We remark that Grave et al. (2017) use a clever trick to estimate the probability, λ_t of drawing from the LM by augmenting their (closed) vocabulary with a special symbol indicating that a copy should be used. This enables word types that are highly predictive in context to compete with the probability of a copy event. However, since we are working with an open vocabulary, this strategy is unavailable in our model, so we use the MLP formulation.

2.4 Training objective

The model parameters as well as the character projection parameters are jointly trained by maximizing the following log likelihood of the observed characters in the training corpus,

$$\mathcal{L} = -\sum \log p(w_t \mid \boldsymbol{w}_{<t}).$$

3 Datasets

We evaluate our model on a range of datasets, employing preexisting benchmarks for comparison to previous published results, and a new multilingual corpus which specifically tests our model's performance across a range of typological settings.

3.1 Penn Tree Bank (PTB)

We evaluate our model on the Penn Tree Bank. For fair comparison with previous works, we followed the standard preprocessing method used by Mikolov et al. (2010). In the standard preprocessing, tokenization is applied, words are lowercased, and punctuation is removed. Also, less frequent words are replaced by unknown an token (UNK),[2] constraining the word vocabulary size to be 10k. Because of this preprocessing, we do not expect this dataset to benefit from the modeling innovations we have introduced in the paper. Fig.1 summarizes the corpus statistics.

[2]When the unknown token is used in character-level model, it is treated as if it were a normal word (i.e. UNK is the sequence U, N, and K). This is somewhat surprising modeling choice, but it has become conventional (Chung et al., 2017).

	Train	Dev	Test
Character types	50	50	48
Word types	10000	6022	6049
OOV rate	-	**0.00%**	**0.00%**
Word tokens	0.9M	0.1M	0.1M
Characters	5.1M	0.4M	0.4M

Table 1: PTB Corpus Statistics.

3.2 WikiText-2

Merity et al. (2017) proposed the WikiText-2 Corpus as a new benchmark dataset.[3] They pointed out that the preprocessed PTB is unrealistic for real language use in terms of word distribution. Since the vocabulary size is fixed to 10k, the word frequency does not exhibit a long tail. The wikiText-2 corpus is constructed from 720 articles. They provided two versions. The version for word level language modeling was preprocessed by discarding infrequent words. But, for character-level models, they provided raw documents without any removal of word or character types or lowercasing, but with tokenization. We make one change to this corpus: since Wikipedia articles make extensive use of characters from other languages; we replaced character types that occur fewer than 25 times were replaced with a dummy character (this plays the role of the ⟨UNK⟩ token in the character vocabulary). Tab. 2 summarizes the corpus statistics.

	Train	Dev	Test
Character types	255	128	138
Word types	76137	19813	21109
OOV rate	-	4.79%	5.87%
Word tokens	2.1M	0.2M	0.2M
Characters	10.9M	1.1M	1.3M

Table 2: WikiText-2 Corpus Statistics.

3.3 Multilingual Wikipedia Corpus (MWC)

Languages differ in what word formation processes they have. For character-level modeling it is therefore interesting to compare a model's performance

[3]http://metamind.io/research/the-wikitext-long-term-dependency-language-modeling-dataset/

across languages. Since there is at present no standard multilingual language modeling dataset, we created a new dataset, the Multilingual Wikipedia Corpus (MWC), a corpus of the same Wikipedia articles in 7 languages which manifest a range of morphological typologies. The MWC contains English (EN), French (FR), Spanish (ES), German (DE), Russian (RU), Czech (CS), and Finnish (FI).

To attempt to control for topic divergences across languages, every language's data consists of the same articles. Although these are only comparable (rather than true translations), this ensures that the corpus has a stable topic profile across languages.[4]

Construction & Preprocessing We constructed the MWC similarly to the WikiText-2 corpus. Articles were selected from Wikipedia in the 7 target languages. To keep the topic distribution to be approximately the same across the corpora, we extracted articles about entities which explained in all the languages. We extracted articles which exist in all languages and each consist of more than 1,000 words, for a total of 797 articles. These cross-lingual articles are, of course, not usually translations, but they tend to be comparable. This filtering ensures that the topic profile in each language is similar. Each language corpus is approximately the same size as the WikiText-2 corpus.

Wikipedia markup was removed with WikiExtractor,[5] to obtain plain text. We used the same thresholds to remove rare characters in the WikiText-2 corpus. No tokenization or other normalization (e.g., lowercasing) was done.

Statistics After the preprocessing described above, we randomly sampled 360 articles. The articles are split into 300, 30, 30 sets and the first 300 articles are used for training and the rest are used for dev and test respectively. Table 3 summarizes the corpus statistics.

Additionally, we show in Fig. 2 the distribution of frequencies of OOV word types (relative to the training set) in the dev+test portions of the corpus, which shows a power-law distribution, which is expected for the burstiness of rare words found in prior work. Curves look similar for all languages (see Appendix A).

[4]The Multilingual Wikipedia Corpus (MWC) is available for download from http://k-kawakami.com/research/mwc

[5]https://github.com/attardi/wikiextractor

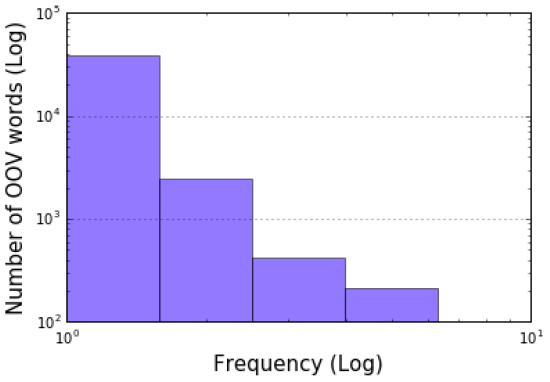


Figure 2: Histogram of OOV word frequencies in the dev+test part of the MWC Corpus (EN).

4 Experiments

We now turn to a series of experiments to show the value of our hierarchical character-level cache language model. For each dataset we trained the model with LSTM units. To compare our results with a strong baseline, we also train a model without the cache.

Model Configuration For HCLM and HCLM with cache models, We used 600 dimensions for the character embeddings and the LSTMs have 600 hidden units for all the experiments. This keeps the model complexity to be approximately the same as previous works which used an LSTM with 1000 dimension. Our baseline LSTM have 1000 dimensions for embeddings and reccurence weights.

For the cache model, we used cache size 100 in every experiment. All the parameters including character projection parameters are randomly sampled from uniform distribution from -0.08 to 0.08. The initial hidden and memory state of LSTM_{enc} and LSTM_{ctx} are initialized with zero. Mini-batches of size 25 are used for PTB experiments and 10 for WikiText-2, due to memory limitations. The sequences were truncated with 35 words. Then the words are decomposed to characters and fed into the model. A Dropout rate of 0.5 was used for all but the recurrent connections.

Learning The models were trained with the Adam update rule (Kingma and Ba, 2015) with a learning rate of 0.002. The maximum norm of the gradients was clipped at 10.

Evaluation We evaluated our models with bits-per-character (bpc) a standard evaluation metric

	Char. Types			Word Types			OOV rate		Tokens			Characters		
	Train	Valid	Test	Train	Valid	Test	Valid	Test	Train	Valid	Test	Train	Valid	Test
EN	307	160	157	193808	38826	35093	6.60%	5.46%	2.5M	0.2M	0.2M	15.6M	1.5M	1.3M
FR	272	141	155	166354	34991	38323	6.70%	6.96%	2.0M	0.2M	0.2M	12.4M	1.3M	1.6M
DE	298	162	183	238703	40848	41962	7.07%	7.01%	1.9M	0.2M	0.2M	13.6M	1.2M	1.3M
ES	307	164	176	160574	31358	34999	6.61%	7.35%	1.8M	0.2M	0.2M	11.0M	1.0M	1.3M
CS	238	128	144	167886	23959	29638	5.06%	6.44%	0.9M	0.1M	0.1M	6.1M	0.4M	0.5M
FI	246	123	135	190595	32899	31109	8.33%	7.39%	0.7M	0.1M	0.1M	6.4M	0.7M	0.6M
RU	273	184	196	236834	46663	44772	7.76%	7.20%	1.3M	0.1M	0.1M	9.3M	1.0M	0.9M

Table 3: Summary of MWC Corpus.

for character-level language models. Following the definition in Graves (2013), bits-per-character is the average value of $-\log_2 p(w_t \mid \boldsymbol{w}_{<t})$ over the whole test set,

$$bpc = -\frac{1}{|\boldsymbol{c}|} \log_2 p(\boldsymbol{w}),$$

where $|\boldsymbol{c}|$ is the length of the corpus in characters.

4.1 Results

PTB Tab. 4 summarizes results on the PTB dataset.[6] Our baseline HCLM model achieved 1.276 bpc which is better performance than the LSTM with Zoneout regularization (Krueger et al., 2017). And HCLM with cache outperformed the baseline model with 1.247 bpc and achieved competitive results with state-of-the-art models with regularization on recurrence weights, which was not used in our experiments.

Expressed in terms of per-word perplexity (i.e., rather than normalizing by the length of the corpus in characters, we normalize by words and exponentiate), the test perplexity on HCLM with cache is 94.79. The performance of the unregularized 2-layer LSTM with 1000 hidden units on word-level PTB dataset is 114.5 and the same model with dropout achieved 87.0. Considering the fact that our character-level models are dealing with an open vocabulary without unknown tokens, the results are promising.

WikiText-2 Tab. 5 summarizes results on the WikiText-2 dataset. Our baseline, LSTM achieved 1.803 bpc and HCLM model achieved 1.670 bpc. The HCLM with cache outperformed the baseline models and achieved 1.500 bpc. The word level perplexity is 227.30, which is quite high compared to the reported word level baseline result 100.9 with LSTM with ZoneOut and Variational Dropout regularization (Merity et al., 2017). However, the character-level model is dealing with 76,136 types in training set and 5.87% OOV rate where the word level models only use 33,278 types without OOV in test set. The improvement rate over the HCLM baseline is 10.2% which is much higher than the improvement rate obtained in the PTB experiment.

[6]Models designated with a * have more layers and more parameters.

Method	Dev	Test
CW-RNN (Koutnik et al., 2014)	-	1.46
HF-MRNN (Mikolov et al., 2012)	-	1.41
MI-RNN (Wu et al., 2016)	-	1.39
ME n-gram (Mikolov et al., 2012)	-	1.37
RBN (Cooijmans et al., 2017)	1.281	1.32
Recurrent Dropout (Semeniuta et al., 2016)	1.338	1.301
Zoneout (Krueger et al., 2017)	1.362	1.297
HM-LSTM (Chung et al., 2017)	-	1.27
HyperNetwork (Ha et al., 2017)	1.296	1.265
LayerNorm HyperNetwork (Ha et al., 2017)	1.281	1.250
2-LayerNorm HyperLSTM (Ha et al., 2017)*	-	1.219
2-Layer with New Cell (Zoph and Le, 2016)*	-	1.214
LSTM (Our Implementation)	1.369	1.331
HCLM	1.308	1.276
HCLM with Cache	**1.266**	**1.247**

Table 4: Results on PTB Corpus (bits-per-character). HCLM augmented with a cache obtains the best results among models which have approximately the same numbers of parameter as single layer LSTM with 1,000 hidden units.

Method	Dev	Test
LSTM	1.758	1.803
HCLM	1.625	1.670
HCLM with Cache	**1.480**	**1.500**

Table 5: Results on WikiText-2 Corpus .

Multilingual Wikipedia Corpus (MWC) Tab. 6 summarizes results on the MWC dataset. Similarly to WikiText-2 experiments, LSTM

is strong baseline. We observe that the cache mechanism improve performance in every languages. In English, HCLM with cache achieved 1.538 bpc where the baseline is 1.622 bpc. It is 5.2% improvement. For other languages, the improvement rates were 2.7%, 3.2%, 3.7%, 2.5%, 4.7%, 2.7% in FR, DE, ES, CS, FI, RU respectively. The best improvement rate was obtained in Finnish.

5 Analysis

In this section, we analyse the behavior of proposed model qualitatively. To analyse the model, we compute the following posterior probability which tell whether the model used the cache given a word and its preceding context. Let z_t be a random variable that says whether to use the cache or the LM to generate the word at time t. We would like to know, given the text $\boldsymbol{w}$, whether the cache was used at time t. This can be computed as follows:

$$p(z_t \mid \boldsymbol{w}) = \frac{p(z_t, w_t \mid \mathbf{h}_t, \text{cache}_t)}{p(w_t \mid \mathbf{h}_t, \text{cache}_t)} = \frac{(1-\lambda_t) p_{ptr}(w_t \mid \mathbf{h}_t, \text{cache}_t)}{p(w_t \mid \mathbf{h}_t, \text{cache}_t)},$$

where cache_t is the state of the cache at time t. We report the average posterior probability of cache generation excluding the first occurrence of w, $\overline{p(z \mid w)}$.

Tab. 7 shows the words in the WikiText-2 test set that occur more than 1 time that are most/least likely to be generated from cache and character language model (words that occur only one time cannot be cache-generated). We see that the model uses the cache for proper nouns: *Lesnar*, *Gore*, etc., as well as very frequent words which always stored somewhere in the cache such as single-token punctuation, *the*, and *of*. In contrast, the model uses the language model to generate numbers (which tend not to be repeated): *300*, *770* and basic content words: *sounds*, *however*, *unable*, etc. This pattern is similar to the pattern found in empirical distribution of frequencies of rare words observed in prior wors (Church and Gale, 1995; Church, 2000), which suggests our model is learning to use the cache to account for bursts of rare words.

To look more closely at rare words, we also investigate how the model handles words that occurred between 2 and 100 times in the test set, but fewer than 5 times in the training set. Fig. 3 is a scatter plot of $\overline{p(z \mid w)}$ vs the empirical frequency in the test set. As expected, more frequently repeated words types are increasingly likely to be drawn from the cache, but less frequent words show a range of cache generation probabilities.

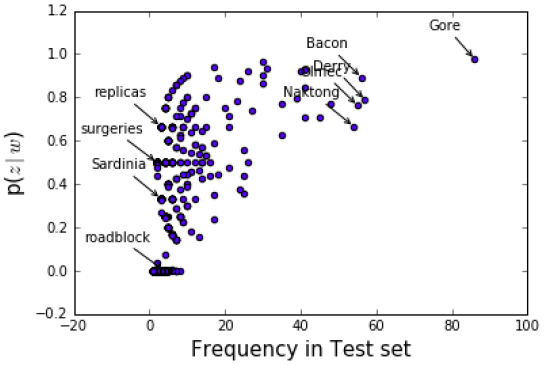


Figure 3: Average $p(z \mid w)$ of OOV words in test set vs. term frequency in the test set for words not obsered in the training set. The model prefers to copy frequently reused words from cache component, which tend to names (upper right) while character level generation is used for infrequent open class words (bottom left).

Tab. 8 shows word types with the highest and lowest average $p(z \mid w)$ that occur fewer than 5 times in the training corpus. The pattern here is similar to the unfiltered list: proper nouns are extremely likely to have been cache-generated, whereas numbers and generic (albeit infrequent) content words are less likely to have been.

6 Discussion

Our results show that the HCLM outperforms a basic LSTM. With the addition of the caching mechanism, the HCLM becomes consistently more powerful than both the baseline HCLM and the LSTM. This is true even on the PTB, which has no rare or OOV words in its test set (because of preprocessing), by caching repetitive common words such as *the*. In true open-vocabulary settings (i.e., WikiText-2 and MWC), the improvements are much more pronounced, as expected.

Computational complexity. In comparison with word-level models, our model has to read and generate each word character by character, and it also requires a softmax over the entire memory at every time step. However, the computation is still linear in terms of the length of the sequence, and the softmax over the memory cells and character

	EN		FR		DE		ES		CS		FI		RU	
	dev	test	dev	test	dev	test	dev	test	dev	test	dev	test	dev	test
LSTM	1.793	1.736	1.669	1.621	1.780	1.754	1.733	1.667	2.191	2.155	1.943	1.913	1.942	1.932
HCLM	1.683	1.622	1.553	1.508	1.666	1.641	1.617	1.555	2.070	2.035	1.832	1.796	1.832	1.810
HCLM with Cache	**1.591**	**1.538**	**1.499**	**1.467**	**1.605**	**1.588**	**1.548**	**1.498**	**2.010**	**1.984**	**1.754**	**1.711**	**1.777**	**1.761**

Table 6: Results on MWC Corpus (bits-per-character).

Word	$\overline{p(z \mid \boldsymbol{w})} \downarrow$	Word	$\overline{p(z \mid w)} \uparrow$
.	0.997	300	0.000
Lesnar	0.991	act	0.001
the	0.988	however	0.002
NY	0.985	770	0.003
Gore	0.977	put	0.003
Bintulu	0.976	sounds	0.004
Nerva	0.976	instead	0.005
,	0.974	440	0.005
UB	0.972	similar	0.006
Nero	0.967	27	0.009
Osbert	0.967	help	0.009
Kershaw	0.962	few	0.010
Manila	0.962	110	0.010
Boulter	0.958	Jersey	0.011
Stevens	0.956	even	0.011
Rifenburg	0.952	y	0.012
Arjona	0.952	though	0.012
of	0.945	becoming	0.013
31B	0.941	An	0.013
Olympics	0.941	unable	0.014

Table 7: Word types with the highest/lowest average posterior probability of having been copied from the cache while generating the test set. The probability tells whether the model used the cache given a word and its context. **Left:** Cache is used for frequent words (*the*, *of*) and proper nouns (*Lesnar*, *Gore*). **Right:** Character level generation is used for basic words and numbers.

Word	$\overline{p(z \mid \boldsymbol{w})} \downarrow$	Word	$\overline{p(z \mid w)} \uparrow$
Gore	0.977	770	0.003
Nero	0.967	246	0.037
Osbert	0.967	Lo	0.074
Kershaw	0.962	Pitcher	0.142
31B	0.941	Poets	0.143
Kirby	0.935	popes	0.143
CR	0.926	Yap	0.143
SM	0.924	Piso	0.143
impedance	0.923	consul	0.143
Blockbuster	0.900	heavyweight	0.143
Superfamily	0.900	cheeks	0.154
Amos	0.900	loser	0.164
Steiner	0.897	amphibian	0.167
Bacon	0.893	squads	0.167
filters	0.889	los	0.167
Lim	0.889	Keenan	0.167
Selfridge	0.875	sculptors	0.167
filter	0.875	Gen.	0.167
Lockport	0.867	Kipling	0.167
Germaniawerft	0.857	Tabasco	0.167

Table 8: Same as Table 7, except filtering for word types that occur fewer than 5 times in the training set. The cache component is used as expected even on rare words: proper nouns are extremely likely to have been cache-generated, whereas numbers and generic content words are less likely to have been; this indicates both the effectiveness of the prior at determining whether to use the cache and the burstiness of proper nouns.

vocabulary are much smaller than word-level vocabulary. On the other hand, since the recurrent states are updated once per character (rather than per word) in our model, the distribution of operations is quite different. Depending on the hardware support for these operations (repeated updates of recurrent states vs. softmaxes), our model may be faster or slower. However, our model will have fewer parameters than a word-based model since most of the parameters in such models live in the word projection layers, and we use LSTMs in place of these.

Non-English languages. For non-English languages, the pattern is largely similar for non-English languages. This is not surprising since morphological processes may generate forms that are related to existing forms, but these still have slight variations. Thus, they must be generated by the language model component (rather than from the cache). Still, the cache demonstrates consistent value in these languages.

Finally, our analysis of the cache on English does show that it is being used to model word reuse, particularly of proper names, but also of frequent words. While empirical analysis of rare word distributions predicts that names would be reused, the fact that cache is used to model frequent words suggests that effective models of language should have a means to generate common words as units. Finally, our model disfavors copying numbers from the cache, even when they are available. This suggests that it has learnt that numbers are not generally repeated (in contrast to names).

7 Related Work

Caching language models were proposed to account for burstiness by Kuhn and De Mori (1990), and recently, this idea has been incorporated to augment neural language models with a caching mechanism (Merity et al., 2017; Grave et al., 2017).

Open vocabulary neural language models have been widely explored (Sutskever et al., 2011; Mikolov et al., 2012; Graves, 2013, *inter alia*). Attempts to make them more aware of word-level dynamics, using models similar to our hierarchical formulation, have also been proposed (Chung et al., 2017).

The only models that are open vocabulary language modeling together with a caching mechanism are the nonparametric Bayesian language models based on hierarchical Pitman–Yor processes which generate a lexicon of word types using a character model, and then generate a text using these (Teh, 2006; Goldwater et al., 2009; Chahuneau et al., 2013). These, however, do not use distributed representations on RNNs to capture long-range dependencies.

8 Conclusion

In this paper, we proposed a character-level language model with an adaptive cache which selectively assign word probability from past history or character-level decoding. And we empirically show that our model efficiently model the word sequences and achieved better perplexity in every standard dataset. To further validate the performance of our model on different languages, we collected multilingual wikipedia corpus for 7 typologically diverse languages. We also show that our model performs better than character-level models by modeling burstiness of words in local context.

The model proposed in this paper assumes the observation of word segmentation. Thus, the model is not directly applicable to languages, such as Chinese and Japanese, where word segments are not explicitly observable. We will investigate a model which can marginalise word segmentation as latent variables in the future work.

Acknowledgements

We thank the three anonymous reviewers for their valuable feedback. The third author acknowledges the support of the EPSRC and nvidia Corporation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Victor Chahuneau, Noah A. Smith, and Chris Dyer. 2013. Knowledge-rich morphological priors for bayesian language models. In *Proc. NAACL*.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical multiscale recurrent neural networks. In *Proc. ICLR*.

Kenneth W Church. 2000. Empirical estimates of adaptation: the chance of two Noriegas is closer to $p/2$ than p^2. In *Proc. COLING*.

Kenneth W Church and William A Gale. 1995. Poisson mixtures. *Natural Language Engineering* 1(2):163–190.

Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. 2017. Recurrent batch normalization. In *Proc. ICLR*.

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition* 112(1):21–54.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In *Proc. ICLR*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

David Ha, Andrew Dai, and Quoc V Le. 2017. Hypernetworks. In *Proc. ICLR*.

Harold Stanley Heaps. 1978. *Information retrieval: Computational and theoretical aspects*. Academic Press, Inc.

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

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. ICLR*.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. 2014. A clockwork RNN. In *Proc. ICML*.

David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, Aaron Courville, et al. 2017. Zoneout: Regularizing rnns by randomly preserving hidden activations. In *Proc. ICLR*.

Roland Kuhn and Renato De Mori. 1990. A cache-based natural language model for speech recognition. *IEEE transactions on pattern analysis and machine intelligence* 12(6):570–583.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proc. EMNLP*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proc. ICLR*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. Interspeech*.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)* .

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2016. Recurrent dropout without memory loss. In *Proc. COLING*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proc. CIKM*.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proc. ICML*.

Yee Whye Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. ACL*.

Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan R Salakhutdinov. 2016. On multiplicative integration with recurrent neural networks. In *Proc. NIPS*.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* .

A Corpus Statistics

Fig. 4 show distribution of frequencies of OOV word types in 6 languages.

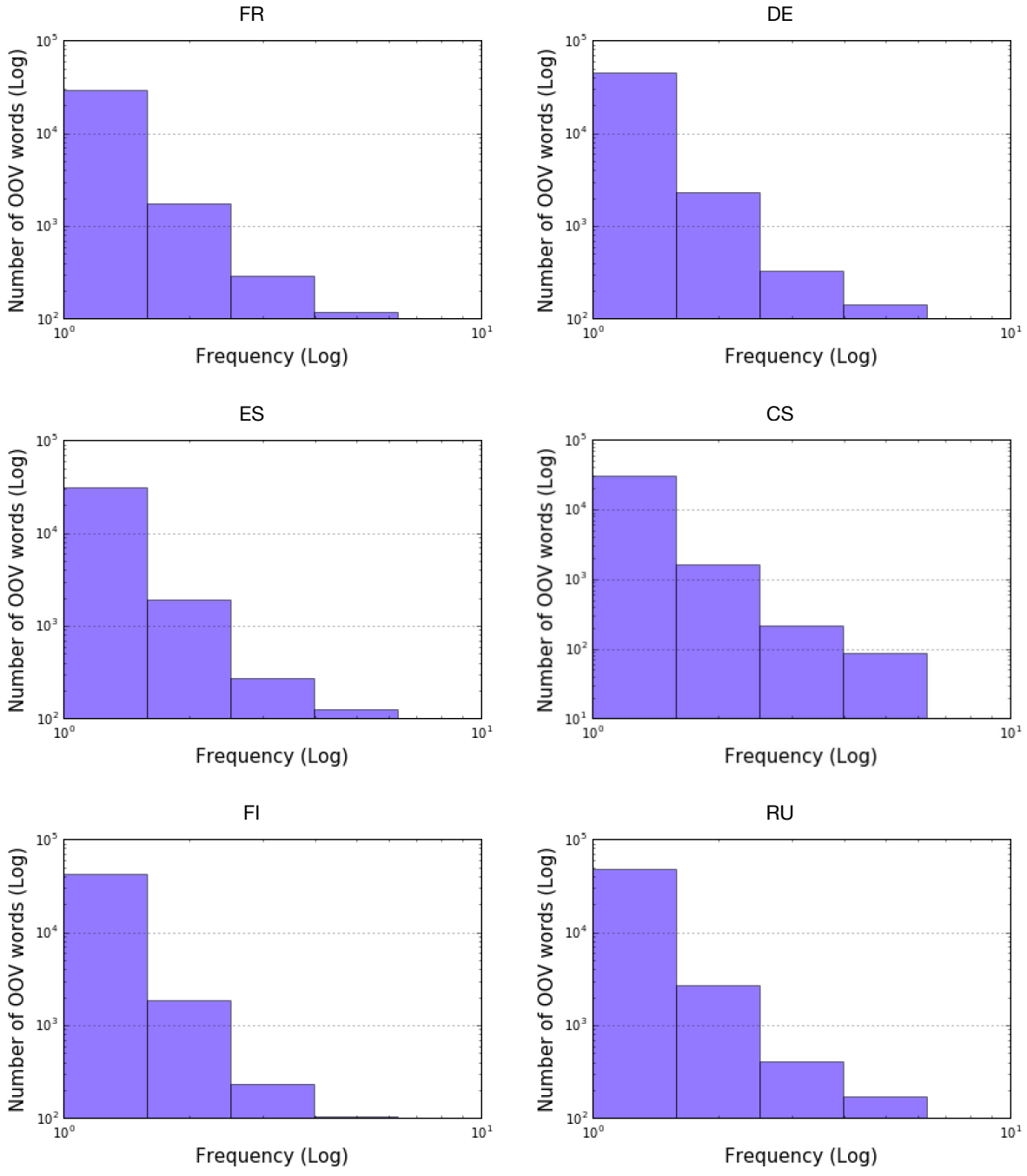


Figure 4: Histogram of OOV word frequencies in MWC Corpus in different languages.

Bandit Structured Prediction for Neural Sequence-to-Sequence Learning

Julia Kreutzer* and **Artem Sokolov*** and **Stefan Riezler†,***
*Computational Linguistics & †IWR, Heidelberg University, Germany
{kreutzer,sokolov,riezler}@cl.uni-heidelberg.de

Abstract

Bandit structured prediction describes a stochastic optimization framework where learning is performed from partial feedback. This feedback is received in the form of a task loss evaluation to a predicted output structure, without having access to gold standard structures. We advance this framework by lifting linear bandit learning to neural sequence-to-sequence learning problems using attention-based recurrent neural networks. Furthermore, we show how to incorporate control variates into our learning algorithms for variance reduction and improved generalization. We present an evaluation on a neural machine translation task that shows improvements of up to 5.89 BLEU points for domain adaptation from simulated bandit feedback.

1 Introduction

Many NLP tasks involve learning to predict a structured output such as a sequence, a tree or a graph. Sequence-to-sequence learning with neural networks has recently become a popular approach that allows tackling structured prediction as a mapping problem between variable-length sequences, e.g., from foreign language sentences into target-language sentences (Sutskever et al., 2014), or from natural language input sentences into linearized versions of syntactic (Vinyals et al., 2015) or semantic parses (Jia and Liang, 2016). A known bottleneck in structured prediction is the requirement of large amounts of gold-standard structures for supervised learning of model parameters, especially for data-hungry neural network models. Sokolov et al. (2016a,b) presented a framework for stochastic structured prediction under bandit feedback that alleviates the need for labeled output structures in learning: Following an online learning protocol, on each iteration the learner receives an input, predicts an output structure, and receives partial feedback in form of a task loss evaluation of the predicted structure.[1] They "banditize" several objective functions for linear structured predictions, and evaluate the resulting algorithms with simulated bandit feedback on various NLP tasks.

We show how to lift linear structured prediction under bandit feedback to non-linear models for sequence-to-sequence learning with attention-based recurrent neural networks (Bahdanau et al., 2015). Our framework is applicable to sequence-to-sequence learning from various types of weak feedback. For example, extracting learning signals from the execution of structured outputs against databases has been established in the communities of semantic parsing and grounded language learning since more than a decade (Zettlemoyer and Collins, 2005; Clarke et al., 2010; Liang et al., 2011). Our work can build the basis for neural semantic parsing from weak feedback.

In this paper, we focus on the application of machine translation via neural sequence-to-sequence learning. The standard procedure of training neural machine translation (NMT) models is to compare their output to human-generated translations and to infer model updates from this comparison. However, the creation of reference translations or post-edits requires professional expertise of users. Our framework allows NMT models to learn from feedback that is weaker than human references or post-edits. One could imagine a scenario of personalized machine translation where translations have to be adapted to the user's specific purpose and domain. The feedback required by our methods can be provided by laymen users or can even

[1] The name "bandit feedback" is inherited from the problem of maximizing the reward for a sequence of pulls of arms of so-called "one-armed bandit" slot machines.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1503–1513
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1138

be implicit, e.g., inferred from user interactions with the translated content on a web page.

Starting from the work of Sokolov et al. (2016a,b), we lift their objectives to neural sequence-to-sequence learning. We evaluate the resulting algorithms on the task of French-to-English translation domain adaptation where a seed model trained on Europarl data is adapted to the NewsCommentary and the TED talks domain with simulated weak feedback. By learning from this feedback, we find 4.08 BLEU points improvements on NewsCommentary, and 5.89 BLEU points improvement on TED. Furthermore, we show how control variates can be integrated in our algorithms, yielding faster learning and improved generalization in our experiments.

2 Related Work

NMT models are most commonly trained under a word-level maximum likelihood objective. Even though this objective has successfully been applied to many sequence-to-sequence learning tasks, the resulting models suffer from exposure bias, since they learn to generate output words based on the history of given reference words, not on their own predictions. Ranzato et al. (2016) apply techniques from reinforcement learning (Sutton and Barto, 1998; Sutton et al., 2000) and imitation learning (Schaal, 1999; Ross et al., 2011; Daumé et al., 2009) to learn from feedback to the model's own predictions. Furthermore, they address the mismatch between word-level loss and sequence-level evaluation metric by using a mixture of the REINFORCE (Williams, 1992) algorithm and the standard maximum likelihood training to directly optimize a sequence-level loss. Similarly, Shen et al. (2016) lift minimum risk training (Och, 2003; Smith and Eisner, 2006; Gimpel and Smith, 2010; Yuille and He, 2012; He and Deng, 2012) from linear models for machine translation to NMT. These works are closely related to ours in that they use the technique of score function gradient estimators (Fu, 2006; Schulman et al., 2015) for stochastic learning. However, the learning environment of Shen et al. (2016) is different from ours in that they approximate the true gradient of the risk objective in a full information setting by sampling a subset of translations and computing the expectation over their rewards. In our bandit setting, feedback to only a single sample per sentence is available, making the learning problem much harder. The approach by Ranzato et al. (2016) approximates the expectation with single samples, but still requires reference translations which are unavailable in the bandit setting.

To our knowledge, the only work on training NMT from weak feedback is the work by He et al. (2016). They propose a dual-learning mechanism where two translation models are jointly trained on monolingual data. The feedback in this case is a reward signal from language models and a reconstruction error. This is attractive because the feedback can automatically be generated from monolingual data and does not require any human references. However, we are interested in using estimates of human feedback on translation quality to directly adapt the model to the users' needs.

Our approach follows most closely the work of Sokolov et al. (2016a,b). They introduce bandit learning objectives for structured prediction and apply them to various NLP tasks, including machine translation with linear models. Their approach can be seen as an instantiation of reinforcement learning to one-state Markov decision processes under linear policy models. In this paper, we transfer their algorithms to non-linear sequence-to-sequence learning. Sokolov et al. (2016a) showed applications of linear bandit learning to tasks such as multiclass-classification, OCR, and chunking, where learning can be done from scratch. We focus on lifting their linear machine translation experiments to the more complex NMT that requires a warm start for training. This is done by training a seed model on one domain and adapting it to a new domain based on bandit feedback only. For this task we build on the work of Freitag and Al-Onaizan (2016), who investigate strategies to find the best of both worlds: models that adapt well to the new domain without deteriorating on the old domain. In contrast to previous approaches to domain adaptation for NMT, we do not require in-domain parallel data, but consult direct feedback to the translations generated for the new domain.

3 Neural Machine Translation

Neural models for machine translation are based on a sequence-to-sequence learning architecture consisting of an encoder and a decoder (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). An encoder Recurrent Neural Network (RNN) reads in the source sentence and a decoder

RNN generates the target sentence conditioned on the encoded source.

The input to the encoder is a sequence of vectors $\mathbf{x} = (x_1, \ldots, x_{T_x})$ representing a sequence of source words of length T_x. In the approach of Sutskever et al. (2014), they are encoded into a single vector $c = q(\{h_1, \ldots, h_{T_x}\})$, where $h_t = f(x_t, h_{t-1})$ is the hidden state of the RNN at time t. Several choices are possible for the non-linear functions f and q: Here we are using a Gated Recurrent Unit (GRU) (Chung et al., 2014) for f, and for q an attention mechanism that defines the context vector as a weighted sum over encoder hidden states (Bahdanau et al., 2015; Luong et al., 2015a).

The decoder RNN predicts the next target word y_t at time t given the context vector c and the previous target words $\mathbf{y}_{<t} = \{y_1, \ldots, y_{t-1}\}$ from a probability distribution over the target vocabulary V. This distribution is the result of a softmax transformation of the decoder outputs $\mathbf{o} = \{o_1, \ldots, o_{T_y}\}$, such that

$$p_\theta(y_t = w_i | \mathbf{y}_{<t}, c) = \frac{\exp(o_{w_i})}{\sum_{v=1}^{V} \exp(o_{w_v})}.$$

The probability of a full sequence of outputs $\mathbf{y} = (y_1, \ldots, y_{T_y})$ of length T_y is defined as the product of the conditional word probabilities:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T_y} p_\theta(y_t | \mathbf{y}_{<t}, c).$$

Since this encoder-decoder architecture is fully differentiable, it can be trained with gradient descent methods. Given a parallel training set of S source sentences and their reference translations $D = \{(\mathbf{x}^{(s)}, \mathbf{y}^{(s)})\}_{s=1}^{S}$, we can define a word-level Maximum Likelihood Estimation (MLE) objective, which aims to find the parameters

$$\hat{\theta}^{\text{MLE}} = \arg\max_\theta L^{\text{MLE}}(\theta)$$

of the following loss function:

$$\begin{aligned} L^{\text{MLE}}(\theta) &= \sum_{s=1}^{S} \log p_\theta(\mathbf{y}^{(s)} | \mathbf{x}^{(s)}) \\ &= \sum_{s=1}^{S} \sum_{t=1}^{T_y} \log p_\theta(y_t | \mathbf{x}^{(s)}, \mathbf{y}^{(s)}_{<t}). \end{aligned}$$

This loss function is non-convex for the case of neural networks. Clever initialization strategies, adaptive learning rates and momentum techniques are required to find good local maxima and to speed up convergence (Sutskever et al., 2013). Another trick of the trade is to ensemble several models with different random initializations to improve over single models (Luong et al., 2015a).

At test time, we face a search problem to find the sequence of target words with the highest probability. Beam search reduces the search error in comparison to greedy search, but also exponentially increases decoding time.

Algorithm 1 Neural Bandit Structured Prediction

Input: Sequence of learning rates γ_k
Output: Optimal parameters $\hat{\theta}$

Initialize θ_0
for $k = 0, \ldots, K$ **do**
  Observe $\mathbf{x}_k$
  Sample $\tilde{\mathbf{y}}_k \sim p_\theta(\mathbf{y}|\mathbf{x}_k)$
  Obtain feedback $\Delta(\tilde{\mathbf{y}}_k)$
  $\theta_{k+1} = \theta_k - \gamma_k \, s_k$
Choose a solution $\hat{\theta}$ from the list $\{\theta_0, \ldots, \theta_K\}$

4 Neural Bandit Structured Prediction

Algorithm 1 is an adaptation of the Bandit Structured Prediction algorithm of Sokolov et al. (2016b) to neural models: For K rounds, a model with parameters θ receives an input, samples an output structure, and receives user feedback. Based on this feedback, a stochastic gradient s_k is computed and the model parameters are updated. As a post-optimization step, a solution $\hat{\theta}$ is selected from the iterates. This is done with online-to-batch conversion by choosing the model with optimal performance on held-out data.

The core of the algorithm is the sampling: if the model distribution is very peaked, the model exploits, i.e., it presents the most probable outputs to the user. If the distribution is close to uniform, the model explores, i.e., it presents random outputs to the user. The balance between exploitation and exploration is crucial to the learning process: in the beginning the model is rather uninformed and needs to explore in order to find outputs with high reward, while in the end it ideally converges towards a peaked distribution that exactly fits the user's needs. Pre-training the model, i.e. setting θ_0 wisely, ensures a reasonable exploitation-exploration trade-off.

This online learning algorithm can be applied

to any objective L provided the stochastic gradients s_k are unbiased estimators of the true gradient of the objective, i.e., we require $\nabla L = \mathbb{E}[s_k]$. In the following, we will present objectives from Sokolov et al. (2016b) transferred to neural models, and explain how they can be enhanced by control variates.

4.1 Expected Loss (EL) Minimization

The first objective is defined as the expectation of a task loss $\Delta(\tilde{\mathbf{y}})$, e.g. $-\text{BLEU}(\tilde{\mathbf{y}})$, over all input and output structures:

$$L^{\text{EL}}(\theta) = \mathbb{E}_{p(\mathbf{x})\, p_\theta(\tilde{\mathbf{y}}|\mathbf{x})} \left[\Delta(\tilde{\mathbf{y}})\right]. \tag{1}$$

In the case of full-information learning where reference outputs are available, we could evaluate all possible outputs against the reference to obtain an exact estimation of the loss function. However, this is not feasible in our setting since we only receive partial feedback for a single output structure per input. Instead, we use stochastic approximation to optimize this loss. The stochastic gradient for this objective is computed as follows:

$$s_k^{\text{EL}} = \Delta(\tilde{\mathbf{y}}) \frac{\partial \log p_\theta(\tilde{\mathbf{y}}|\mathbf{x}_k)}{\partial \theta}. \tag{2}$$

Objective (1) is known from minimum risk training (Och, 2003) and has been lifted to NMT by Shen et al. (2016) – but not for learning from weak feedback. Equation (2) is an instance of the score function gradient estimator (Fu, 2006) where

$$\nabla \log p_\theta(\tilde{\mathbf{y}}|\mathbf{x}_k) \tag{3}$$

denotes the score function. We give an algorithm to sample structures from an encoder-decoder model in Algorithm 2. It corresponds to the algorithm presented by Shen et al. (2016) with the difference that it samples single structures, does not assume a reference structure, and additionally returns the sample probabilities. A similar objective has also been used in the REINFORCE algorithm (Williams, 1992) which has been adapted to NMT by Ranzato et al. (2016).

4.2 Pairwise Preference Ranking (PR)

The previous objective requires numerical feedback as an estimate of translation quality. Alternatively, we can learn from pairwise preference judgments that are formalized in preference ranking objectives. Let $\mathcal{P}(\mathbf{x}) = \{\langle \mathbf{y}_i, \mathbf{y}_j \rangle \,|\mathbf{y}_i, \mathbf{y}_j \in \mathcal{Y}(\mathbf{x})\}$ denote the set of output pairs for an input $\mathbf{x}$, and let $\Delta(\langle \mathbf{y}_i, \mathbf{y}_j \rangle) : \mathcal{P}(\mathbf{x}) \rightarrow [0, 1]$ denote a task loss function that specifies a dispreference of y_i over y_j. In our experimental simulations we use two types of pairwise feedback. Firstly, continuous pairwise feedback[2] is computed as

$$\Delta(\langle \mathbf{y}_i, \mathbf{y}_j \rangle) = \Delta(\mathbf{y}_j) - \Delta(\mathbf{y}_i),$$

and secondly, binary feedback is computed as

$$\Delta(\langle \mathbf{y}_i, \mathbf{y}_j \rangle) = \begin{cases} 1 & \text{if } \Delta(\mathbf{y}_j) > \Delta(\mathbf{y}_i), \\ 0 & \text{otherwise.} \end{cases}$$

Analogously to the sequence-level sampling for linear models (Sokolov et al., 2016b), we define the following probabilities for word-level sampling:

$$p_\theta^+(\tilde{y}_t = w_i|\mathbf{x}, \hat{\mathbf{y}}_{<t}) = \frac{\exp(o_{w_i})}{\sum_{v=1}^{V} \exp(o_{w_v})},$$
$$p_\theta^-(\tilde{y}_t = w_j|\mathbf{x}, \hat{\mathbf{y}}_{<t}) = \frac{\exp(-o_{w_j})}{\sum_{v=1}^{V} \exp(-o_{w_v})}.$$

The effect of the negation within the softmax is that the two distributions p_θ^+ and p_θ^- rank the next candidate target words $\tilde{y}_t$ (given the same history, here the greedy output $\hat{\mathbf{y}}_{<t}$) in opposite order. Globally normalized models as in the linear case, or LSTM-CRFs (Huang et al., 2015) for the non-linear case would allow sampling full structures such that the ranking over full structures is reversed. But in the case of locally normalized RNNs we retrieve only locally reversed-rank samples. Since we want the model to learn to rank $\tilde{\mathbf{y}}_i$ over $\tilde{\mathbf{y}}_j$, we would have to sample $\tilde{\mathbf{y}}_i$ word-by-word from p_θ^+ and $\tilde{\mathbf{y}}_j$ from p_θ^-. However, sampling all words of $\tilde{\mathbf{y}}_j$ from p_θ^- leads to translations that are neither fluent nor source-related, so we propose to randomly choose one position of $\tilde{\mathbf{y}}_j$ where the next word is sampled from p_θ^- and sample the remaining words from p_θ^+. We found that this method produces suitable negative samples, which are only slightly perturbed and still relatively fluent and source-related. A detailed algorithm is given in Algorithm 3.

In the same manner as for linear models, we define the probability of a pair of sequences as

$$p_\theta(\langle \tilde{\mathbf{y}}_i, \tilde{\mathbf{y}}_j \rangle |\mathbf{x}) = p_\theta^+(\tilde{\mathbf{y}}_i|\mathbf{x}) \times p_\theta^-(\tilde{\mathbf{y}}_j|\mathbf{x}).$$

[2]Note that our definition of continuous feedback is slightly different from the one proposed in Sokolov et al. (2016b) where updates are only made for misrankings.

Algorithm 2 Sampling Structures

Input: Model θ, target sequence length limit T_y
Output: Sequence of words $\mathbf{w} = (w_1, \ldots, w_{Ty})$ and log-probability p

$w_0 = \text{START}, p_0 = 0$
$\mathbf{w} = (w_0)$
for $t \leftarrow 1 \ldots T_y$ **do**
  $w_t \sim p_\theta(w|\mathbf{x}, \mathbf{w}_{<t})$
  $p_t = p_{t-1} + \log p_\theta(w|\mathbf{x}, \mathbf{w}_{<t})$
  $\mathbf{w} = (w_1, \ldots, w_{t-1}, w_t)$
end for
Return $\mathbf{w}$ and p_T

Note that with the word-based sampling scheme described above, the sequence $\tilde{\mathbf{y}}_j$ also includes words sampled from p_θ^+.
The pairwise preference ranking objective expresses an expectation over losses over these pairs:

$$L^{\text{PR}}(\theta) = \mathbb{E}_{p(\mathbf{x})\, p_\theta(\langle \tilde{\mathbf{y}}_i, \tilde{\mathbf{y}}_j \rangle | \mathbf{x})} \left[\Delta(\langle \tilde{\mathbf{y}}_i, \tilde{\mathbf{y}}_j \rangle) \right]. \quad (4)$$

The stochastic gradient for this objective is

$$s_k^{\text{PR}} = \Delta(\langle \tilde{\mathbf{y}}_i, \tilde{\mathbf{y}}_j \rangle) \times \left(\frac{\partial \log p_\theta^+(\tilde{\mathbf{y}}_i|\mathbf{x}_k)}{\partial \theta} + \frac{\partial \log p_\theta^-(\tilde{\mathbf{y}}_j|\mathbf{x}_k)}{\partial \theta} \right). \quad (5)$$

This training procedure resembles well-known approaches for noise contrastive estimation (Gutmann and Hyvärinen, 2010) with negative sampling that are commonly used for neural language modeling (Collobert et al., 2011; Mnih and Teh, 2012; Mikolov et al., 2013). In these approaches, negative samples are drawn from a non-parametric noise distribution, whereas we draw them from the perturbed model distribution.

4.3 Control Variates

The stochastic gradients defined in equations (2) and (5) can be used in stochastic gradient descent optimization (Bottou et al., 2016) where the full gradient is approximated using a minibatch or a single example in each update. The stochastic choice, in our case on inputs and outputs, introduces noise that leads to slower convergence and degrades performance. In the following, we explain how antithetic and additive control variate techniques from Monte Carlo simulation (Ross, 2013) can be used to remedy these problems.

The idea of additive control variates is to augment a random variable X whose expectation is sought, by another random variable Y to which X is highly correlated. Y is then called the control variate. Let $\bar{Y}$ furthermore denote its expectation. Then the following quantity $X - \hat{c}\,Y + \hat{c}\,\bar{Y}$ is an unbiased estimator of $\mathbb{E}[X]$. In our case, the random variable of interest is the noisy gradient $X = s_k$ from Equation (2). The variance reduction effect of control variates can be seen by computing the variance of this quantity:

$$\text{Var}(X - \hat{c}\,Y) = \text{Var}(X) + \hat{c}^2 \text{Var}(Y) - 2\hat{c}\,\text{Cov}(X, Y). \quad (6)$$

Choosing a control variate such that $\text{Cov}(X, Y)$ is positive and high enough, the variance of the gradient estimate will be reduced.

An example is the average reward baseline known from reinforcement learning (Williams, 1992), yielding

$$Y_k = \nabla \log p_\theta(\tilde{\mathbf{y}}|\mathbf{x}_k) \frac{1}{k} \sum_{j=1}^{k} \Delta(\tilde{\mathbf{y}}_j). \quad (7)$$

The optimal scalar $\hat{c}$ can be derived easily by taking the derivative of (6), leading to $\hat{c} = \frac{\text{Cov}(X,Y)}{\text{Var}(X)}$. This technique has been applied to using the score

Algorithm 3 Sampling Pairs of Structures

Input: Model θ, target sequence length limit T_y
Output: Pair of sequences $\langle \mathbf{w}, \mathbf{w}' \rangle$ and their log-probability p

$p_0 = 0$
$\mathbf{w}, \mathbf{w}', \hat{\mathbf{w}} = (\text{START})$
$i \sim \mathcal{U}(1, T)$
for $t \leftarrow 1 \ldots T_y$ **do**
  $\hat{w}_t = \arg\max_{w \in V} p_\theta^+(w|\mathbf{x}, \hat{\mathbf{w}}_{<t})$
  $w_t \sim p_\theta^+(w|\mathbf{x}, \hat{\mathbf{w}}_{<t})$
  $p_t = p_{t-1} + \log p_\theta^+(w_t|\mathbf{x}, \hat{\mathbf{w}}_{<t})$
  if $i = t$ **then**
    $w'_t \sim p_\theta^-(w|\mathbf{x}, \hat{\mathbf{w}}_{<t})$
    $p_t = p_t + \log p_\theta^-(w'_t|\mathbf{x}, \hat{\mathbf{w}}_{<t})$
  else
    $w'_t \sim p_\theta^+(w|\mathbf{x}, \hat{\mathbf{w}}_{<t})$
    $p_t = p_t + \log p_\theta^+(w'_t|\mathbf{x}, \hat{\mathbf{w}}_{<t})$
  end if
  $\mathbf{w} = (w_1, \ldots, w_{t-1}, w_t)$
  $\mathbf{w}' = (w'_1, \ldots, w'_{t-1}, w'_t)$
  $\hat{\mathbf{w}} = (\hat{w}_1, \ldots, \hat{w}_{t-1}, \hat{w}_t)$
end for
Return $\langle \mathbf{w}, \mathbf{w}' \rangle$ and p_T

function (Equation (3)) as control variate in Ranganath et al. (2014), yielding the following control variate:

$$Y_k = \nabla \log p_\theta(\tilde{\mathbf{y}}|\mathbf{x}_k). \quad (8)$$

Note that for both types of control variates, (7) and (8), the expectation $\bar{Y}$ is zero, simplifying the implementation. However, the optimal scalar $\hat{c}$ has to be estimated for every entry of the gradient separately for the score function control variate. We will explore both types of control variates for the stochastic gradient (2) in our experiments.

A further effect of control variates is to reduce the magnitude of the gradient, the more so the more the stochastic gradient and the control variate covary. For L-Lipschitz continuous functions, a reduced gradient norm directly leads to a bound on L which appears in the algorithmic stability bounds of Hardt et al. (2016). This effect of improved generalization by control variates is empirically validated in our experiments.

A similar variance reduction effect can be obtained by antithetic control variates. Here $\mathbb{E}[X]$ is approximated by the estimator $\frac{X_1+X_2}{2}$ whose variance is

$$\mathrm{Var}\left(\frac{X_1+X_2}{2}\right) = \frac{1}{4}\big(\mathrm{Var}(X_1) + \mathrm{Var}(X_2) + 2\mathrm{Cov}(X_1, X_2)\big). \quad (9)$$

Choosing the variates X_1 and X_2 such that $\mathrm{Cov}(X_1, X_2)$ is negative will reduce the variance of the gradient estimate. Under certain assumptions, the stochastic gradient (5) of the pairwise preference objective can be interpreted as an antithetic estimator of the score function (3). The antithetic variates in this case would be

$$X_1 = \nabla \log p^+_\theta(\tilde{\mathbf{y}}_i|\mathbf{x}_k), \quad (10)$$
$$X_2 = \nabla \log p^-_\theta(\tilde{\mathbf{y}}_j|\mathbf{x}_k),$$

where an antithetic dependence of X_2 on X_1 can be achieved by construction of p^+_θ and p^-_θ (see Capriotti (2008) which is loosely related to our approach). Similar to control variates, antithetic variates have the effect of shrinking the gradient norm, the more so the more the variates are antithetically correlated, leading to possible improvements in algorithmic stability (Hardt et al., 2016).

5 Experiments

In the following, we present an experimental evaluation of the learning objectives presented above on machine translation domain adaptation. We compare how the presented neural bandit learning objectives perform in comparison to linear models, then discuss the handling of unknown words and eventually investigate the impact of techniques for variance reduction.

Domain	Version	Train	Valid.	Test
Europarl	v.5	1.6M	2k	2k
News Commentary	WMT07	40k	1k	2k
TED	TED2013	153k	2k	2k

Table 1: Number of parallel sentences for training, validation and test sets for French-to-English domain adaptation.

5.1 Setup

Data. We perform domain adaptation from Europarl (EP) to News Commentary (NC) and TED talks (TED) for translations from French to English. Table 1 provides details about the datasets. For data pre-processing we follow the procedure of Sokolov et al. (2016a,b) using `cdec` tools for filtering, lowercasing and tokenization. The challenge for the bandit learner is to adapt from the EP domain to NC or TED with weak feedback only.

NMT Models. We choose a standard encoder-decoder architecture with single-layer GRU RNNs with 800 hidden units, a word embedding size of 300 and `tanh` activations. The encoder consists of a bidirectional RNN, where the hidden states of backward and forward RNN are concatenated. The decoder uses the attention mechanism proposed by Bahdanau et al. (2015).[3] Source and target vocabularies contain the 30k most frequent words of the respective parts of the training corpus. We limit the maximum sentence length to 50. Dropout (Srivastava et al., 2014) with a probability of 0.5 is applied to the network in several places: on the embedded inputs, before the output layer, and on the initial state of the decoder RNN. The gradient is clipped when its norms exceeds 1.0 to prevent exploding gradients and stabilize learning (Pascanu et al., 2013). All models are implemented and trained with the sequence learning framework `Neural Monkey` (Libovický et al.,

[3] We do not use beam search nor ensembling, although we are aware that higher performance is almost guaranteed with these techniques. Our goal is to show relative differences between different models, so a simple setup is sufficient for the purpose of our experiments.

2016; Bojar et al., 2016).[4] They are trained with a minibatch size of 20, fitting onto single 8GB GPU machines. The training dataset is shuffled before each epoch.

Baselines. The out-of-domain baseline is trained on the EP training set with standard MLE. For both NC and TED domains, we train two full-information in-domain baselines: The first in-domain baseline is trained on the relatively small in-domain training data. The second in-domain baseline starts from the out-of-domain model and is further trained on the in-domain data. All baselines are trained with MLE and Adam (Kingma and Ba, 2014) ($\alpha = 1 \times 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$) until their performance stops increasing on respective held-out validation sets. The gap between the performance of the out-of-domain model and the in-domain models defines the range of possible improvements for bandit learning. All models are evaluated with Neural Monkey's `mteval`. For statistical significance tests we used Approximate Randomization testing (Noreen, 1989).

Bandit Learning. Bandit learning starts with the parameters of the out-of-domain baseline. The bandit models are expected to improve over the out-of-domain baseline by receiving feedback from the new domain, but at most to reach the in-domain baseline since the feedback is weak. The models are trained with Adam on in-domain data for at most 20 epochs. Adam's step-size parameter α was tuned on the validation set and was found to perform best when set to 1×10^{-5} for non-pairwise, 1×10^{-6} for pairwise objectives on NC, 1×10^{-7} for pairwise objectives on TED. The best model parameters, selected with early stopping on the in-domain validation set, are evaluated on the held-out in-domain test set. In the spirit of Freitag and Al-Onaizan (2016) they are additionally evaluated on the out-of-domain test set to investigate how much knowledge of the old domain the models lose while adapting to the new domain. Bandit learning experiments are repeated two times, with different random seeds, and mean BLEU scores with standard deviation are reported.

[4]The `Neural Monkey` fork `https://github.com/juliakreutzer/bandit-neuralmonkey` contains bandit learning objectives and the configuration files for our experiments.

Feedback Simulation. Weak feedback is simulated from the target side of the parallel corpus, but references are never revealed to the learner. Sokolov et al. (2016a,b) used a smoothed version of per-sentence BLEU for simulating the weak feedback for generated translations from the comparison with reference translations. Here, we use gGLEU instead, which Wu et al. (2016) recently introduced for learning from sentence-level reward signals correlating well with corpus BLEU. This metric is closely related to BLEU, but does not have a brevity penalty and considers the recall of matching n-grams. It is defined as the minimum of recall and precision over the total n-grams up to a certain n. Hence, for our experiments $\Delta(\tilde{\mathbf{y}}) = -\text{gGLEU}(\tilde{\mathbf{y}}, \mathbf{y})$, where $\tilde{\mathbf{y}}$ is a sample translation and $\mathbf{y}$ is the reference translation.

Unknown words. One drawback of NMT models is their limitation to a fixed source- and target vocabulary. In a domain adaptation setting, this limitation has a critical impact to the translation quality. The larger the distance between old and new domain, the more words in the new domain are unknown to the models trained on the old domain (represented with a special UNK token). We consider two strategies for this problem for our experiments:

1. UNK-Replace: Jean et al. (2015) and Luong et al. (2015b) replace generated UNK tokens with aligned source words or their lexical translations in a post-processing step. Freitag and Al-Onaizan (2016) and Hashimoto et al. (2016) demonstrated that this technique is beneficial for NMT domain adaptation.

2. BPE: Sennrich et al. (2016) introduce byte pair encoding (BPE) for word segmentation to build translation models on sub-word units. Rare words are decomposed into sub-word units, while the most frequent words remain single vocabulary items.

For UNK-Replace we use `fast_align` to generate lexical translations on the EP training data. When an UNK token is generated, we look up the attention weights and find the source token that receives most attention in this step. If possible, we replace the UNK token by its lexical translation. If it is not included in the lexical translations, it is replaced by the source token. The main benefit of this technique is that it deals well

Algorithm	Train data	Iter.	EP	NC	TED
MLE	EP	12.3M	31.44	26.98	23.48
MLE-UNK			31.82	28.00	24.59
MLE-BPE		12.0M	31.81	27.20	24.35

Table 2: Out-of-domain NMT baseline results (BLEU) on in- and out-of-domain test sets trained only on EP data.

with unknown named entities that are just passed through from source to target. However, since it is a non-differentiable post-processing step, the NMT model cannot directly be trained for this behavior. Therefore we also train sub-word level NMT with BPE. We apply 29,800 merge operations to obtain a vocabulary of 29,908 sub-words. The procedure for training these models is exactly the same as for the word-based models. The advantage of this method is that the model is in principle able to generate any word composing it from sub-word units. However, training sequences become longer and candidate translations are sampled on a sub-word level, which introduces the risk of sampling nonsense words.

Control variates. We implement the average baseline control variate as defined in Equation 7, which results in keeping an running average over previous losses. Intuitively, absolute gGLEU feedback is turned into relative feedback that reflects the current state of the model. The sign of the update is switched when the gGLEU for the current sample is worse than the average gGLEU, so the model makes a step away from it, while in the case of absolute feedback it would still make a small step towards it. In addition, we implement the score function control variate with a running estimate $\hat{c}_k = \frac{1}{k}\sum_{j=1}^{k} \frac{\mathrm{Cov}(s_j, \nabla \log p_\theta(\tilde{\mathbf{y}}_j|\mathbf{x}_j))}{\mathrm{Var}(s_j)}$.

5.2 Results

In the following, we discuss the results of the experimental evaluation of the models described above. The out-of-domain baseline results are given in Table 2, those for the in-domain baselines in 3. The results for bandit learning on NC and TED are reported in Table 4. For bandit learning we give mean improvements over the respective out-of-domain baselines in the Diff.-columns.

Baselines. The NMT out-of-domain baselines, reported in Table 2, perform comparable to the linear baseline from Sokolov et al. (2016a,b) on

Algorithm	Train data	Iter.	EP	NC
MLE	NC	978k	13.67	22.32
MLE-UNK			13.83	22.56
MLE-BPE		1.0M	14.09	23.01
MLE	EP→NC	160k	26.66	31.91
MLE-UNK			27.19	33.19
MLE-BPE		160k	27.14	33.31
Algorithm	**Train data**	**Iter.**	**EP**	**TED**
MLE	TED	2.2M	14.16	32.71
MLE-UNK			15.15	33.16
MLE-BPE		3.0M	14.18	32.81
MLE	EP→TED	460k	23.88	33.65
MLE-UNK			24.64	35.57
MLE-BPE		2.2M	23.39	36.23

Table 3: In-domain NMT baselines results (BLEU) on in- and out-of-domain test sets. The EP→NC is first trained on EP, then fine-tuned on NC. The EP→TED is first trained on EP, then fine-tuned on TED.

NC, but the in-domain EP→NC (Table 3) baselines outperform the linear baseline by more than 3 BLEU points. Continuing training of a pre-trained out-of-domain model on a small amount of in domain data is very hence effective, whilst the performance of the models solely trained on small in-domain data is highly dependent on the size of this training data set. For TED, the in-domain dataset is almost four times as big as the NC training set, so the in-domain baselines perform better. This effect was previously observed by Luong and Manning (2015) and Freitag and Al-Onaizan (2016).

Bandit Learning. The NMT bandit models that optimize the EL objective yield generally a much higher improvement over the out-of-domain models than the corresponding linear models: As listed in Table 4, we find improvements of between 2.33 and 2.89 BLEU points on the NC domain, and between 4.18 and 5.18 BLEU points on the TED domain. In contrast, the linear models with sparse features and hypergraph re-decoding achieved a maximum improvement of 0.82 BLEU points on NC.

Optimization of the PR objective shows improvements of up to 1.79 BLEU points on NC (compared to 0.6 BLEU points for linear models), but no significant improvement on TED. The biggest impact of this variance reduction tech-

Algorithm	Iter.	EP	NC	Diff.
EL	317k	$30.36_{\pm 0.20}$	$29.34_{\pm 0.29}$	2.36
EL-UNK*	317k	$30.73_{\pm 0.20}$	$30.33_{\pm 0.42}$	2.33
EL-UNK**	349k	$30.67_{\pm 0.04}$	$30.45_{\pm 0.27}$	2.45
EL-BPE	543k	$30.09_{\pm 0.31}$	$30.09_{\pm 0.01}$	2.89
PR-UNK** (bin)	22k	$30.76_{\pm 0.03}$	$29.40_{\pm 0.02}$	1.40
PR-BPE (bin)	14k	$31.02_{\pm 0.09}$	$28.92_{\pm 0.03}$	1.72
PR-UNK** (cont)	12k	$30.81_{\pm 0.02}$	$29.43_{\pm 0.02}$	1.43
PR-BPE (cont)	8k	$30.91_{\pm 0.01}$	$28.99_{\pm 0.00}$	1.79
SF-EL-UNK**	713k	$29.97_{\pm 0.09}$	$30.61_{\pm 0.05}$	2.61
SF-EL-BPE	375k	$30.46_{\pm 0.10}$	$30.20_{\pm 0.11}$	3.00
BL-EL-UNK**	531k	$30.19_{\pm 0.37}$	$31.47_{\pm 0.09}$	3.47
BL-EL-BPE	755k	$29.88_{\pm 0.07}$	$31.28_{\pm 0.24}$	**4.08**

(a) Domain adaptation from EP to NC.

Algorithm	Iter.	EP	TED	Diff.
EL	976k	$29.34_{\pm 0.42}$	$27.66_{\pm 0.03}$	4.18
EL-UNK*	976k	$29.68_{\pm 0.29}$	$29.44_{\pm 0.06}$	4.85
EL-UNK**	1.1M	$29.62_{\pm 0.15}$	$29.77_{\pm 0.15}$	5.18
EL-BPE	831k	$30.03_{\pm 0.43}$	$28.54_{\pm 0.04}$	4.18
PR-UNK** (bin)	14k	$31.84_{\pm 0.01}$	$24.85_{\pm 0.00}$	0.26
PR-BPE (bin)	69k	$31.77_{\pm 0.01}$	$24.55_{\pm 0.01}$	0.20
PR-UNK** (cont)	9k	$31.85_{\pm 0.02}$	$24.85_{\pm 0.01}$	0.26
PR-BPE (cont)	55k	$31.79_{\pm 0.02}$	$24.59_{\pm 0.01}$	0.24
SF-EL-UNK**	658k	$30.18_{\pm 0.15}$	$29.12_{\pm 0.10}$	4.53
SF-EL-BPE	590k	$30.32_{\pm 0.26}$	$28.51_{\pm 0.18}$	4.16
BL-EL-UNK**	644k	$29.91_{\pm 0.03}$	$30.44_{\pm 0.13}$	5.85
BL-EL-BPE	742k	$29.84_{\pm 0.61}$	$30.24_{\pm 0.46}$	**5.89**

(b) Domain adaptation from EP to TED.

Table 4: Bandit NMT results (BLEU) on EP, NC and TED test sets. UNK* models involve UNK replacement only during testing, UNK** include UNK replacement already during training. For PR, either binary (bin) or continuous feedback (cont) was used. Control variates: average reward baseline (BL) and score function (SF). Results are averaged over two independent runs and standard deviation is given in subscripts. Improvements over respective out-of-domain models are given in the Diff.-columns.

nique is a considerable speedup of training speed of 1 to 2 orders of magnitude compared to EL.

A beneficial side-effect of NMT learning from weak feedback is that the knowledge from the out-domain training is not simply "overwritten". This happens to full-information in-domain tuning where more than 4 BLEU points are lost in an evaluation on the out-domain data. On the contrary, the bandit learning models still achieve high results on the original domain. This is useful for conservative domain adaptation, where the performance of the models in the old domain is still relevant.

Unknown words. By handling unknown words with UNK-Replace or BPEs, we find consistent improvements over the plain word-based models for all baselines and bandit learning models. We observe that the models with UNK replacement essentially benefit from passing through source tokens, and only marginally from lexical translations. Bandit learning models take particular advantage of UNK replacement when it is included already during training. The sub-word models achieve the overall highest improvement over the baselines, although sometimes generating nonsense words.

Control variates. Applying the score function control variate to EL optimization does not largely change learning speed or BLEU results. However, the average reward control variate leads to improvements of around 1 BLEU over the EL optimization without variance reduction on both domains.

6 Conclusion

In this paper, we showed how to lift structured prediction under bandit feedback from linear models to non-linear sequence-to-sequence learning using recurrent neural networks with attention. We introduced algorithms to train these models under numerical feedback to single output structures or under preference rankings over pairs of structures. In our experimental evaluation on the task of neural machine translation domain adaptation, we found relative improvements of up to 5.89 BLEU points over out-of-domain seed models, outperforming also linear bandit models. Furthermore, we argued that pairwise ranking under bandit feedback can be interpreted as a use of antithetic variates, and we showed how to include average reward and score function baselines as control variates for improved training speed and generalization. In future work, we would like to apply the presented non-linear bandit learners to other structured prediction tasks.

Acknowledgments

This research was supported in part by the German research foundation (DFG), and in part by a research cooperation grant with the Amazon Development Center Germany.

References

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

Ondřej Bojar, Roman Sudarikov, Tom Kocmi, Jindřich Helcl, and Ondřej Cıfka. 2016. UFAL submissions to the IWSLT 2016 MT track. In *IWSLT*. Seattle, WA.

Leon Bottou, Frank E. Curtis, and Jorge Nocedal. 2016. Optimization methods for large-scale machine learning. *eprint arXiv:1606.04838v1* .

Luca Capriotti. 2008. Reducing the variance of likelihood ratio greeks in Monte Carlo. In *WCS*. Miami, FL.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*. Doha, Qatar.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *eprint arXiv:1412.3555* .

James Clarke, Dan Goldwasser, Wing-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*. Portland, OR.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR* 12:2461–2505.

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

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *eprint arXiv:1612.06897* .

Michael C. Fu. 2006. Gradient estimation. In S.G. Henderson and B.L. Nelson, editors, *Handbook in Operations Research and Management Science*, volume 13, pages 575–616.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin training for structured log-linear models. Technical Report CMU-LTI-10-008, Carnegie Mellon University.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*. Sardinia, Italy.

Moritz Hardt, Ben Recht, and Yoram Singer. 2016. Train faster, generalize better: Stability of stochastic gradient descent. In *ICML*. New York, NY.

Kazuma Hashimoto, Akiko Eriguchi, and Yoshimasa Tsuruoka. 2016. Domain adaptation and attention-based unknown word replacement in chinese-to-japanese neural machine translation. In *COLING Workshop on Asian Translation*. Osaka, Japan.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *NIPS*. Barcelona, Spain.

Xiaodong He and Li Deng. 2012. Maximum expected BLEU training of phrase and lexicon translation models. In *ACL*. Jeju Island, Korea.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *eprint arXiv:1508.01991* .

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for WMT'15. In *WMT*. Lisbon, Portugal.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *ACL*. Berlin, Germany.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *eprint arXiv:1412.6980* .

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *ACL-HLT*. Portland, OR.

Jindřich Libovickỳ, Jindřich Helcl, Marek Tlustỳ, Pavel Pecina, and Ondřej Bojar. 2016. CUNI system for WMT16 automatic post-editing and multimodal translation tasks. In *WMT*. Berlin, Germany.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *IWSLT*. Da Nang, Vietnam.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *EMNLP*. Lisbon, Portugal.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *ACL*. Beijing, China.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. Lake Tahoe, CA.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *ICML*. Edinburgh, Scotland.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *HLT-NAACL*. Edmonton, Canada.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*. Atlanta, GA.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. 2014. Black box variational inference. In *AISTATS*. Reykjavik, Iceland.

MarcAurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*. San Juan, Puerto Rico.

Sheldon M. Ross. 2013. *Simulation*. Elsevier, fifth edition.

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*. Ft. Lauderdale, FL.

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

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. 2015. Gradient estimation using stochastic computation graphs. In *NIPS*. Montreal, Canada.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*. Berlin, Germany.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *ACL*. Berlin, Germany.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *COLING-ACL*. Sydney, Australia.

Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. 2016a. Learning structured predictors from bandit feedback for interactive NLP. In *ACL*. Berlin, Germany.

Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. 2016b. Stochastic structured prediction under bandit feedback. In *NIPS*. Barcelona, Spain.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *ICML*. Atlanta, GA.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. Montreal, Canada.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning. An Introduction*. The MIT Press.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*. Vancouver, Canada.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*. Montreal, Canada.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 20:229–256.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *eprint arXiv:1609.08144* .

Alan Yuille and Xuming He. 2012. Probabilistic models of vision and max-margin methods. *Frontiers of Electrical and Electronic Engineering* 7(1):94–106.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*. Edinburgh, Scotland.

Prior Knowledge Integration for Neural Machine Translation using Posterior Regularization

Jiacheng Zhang[†], **Yang Liu**[†‡*], **Huanbo Luan**[†], **Jingfang Xu**[#] **and Maosong Sun**[†‡]
[†]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[‡]Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China
[#]Sogou Inc., Beijing, China

Abstract

Although neural machine translation has made significant progress recently, how to integrate multiple overlapping, arbitrary prior knowledge sources remains a challenge. In this work, we propose to use posterior regularization to provide a general framework for integrating prior knowledge into neural machine translation. We represent prior knowledge sources as features in a log-linear model, which guides the learning process of the neural translation model. Experiments on Chinese-English translation show that our approach leads to significant improvements.

1 Introduction

The past several years have witnessed the rapid development of neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015), which aims to model the translation process using neural networks in an end-to-end manner. With the capability of capturing long-distance dependencies due to the gating (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) and attention (Bahdanau et al., 2015) mechanisms, NMT has shown remarkable superiority over conventional statistical machine translation (SMT) across a variety of natural languages (Junczys-Dowmunt et al., 2016).

Despite the apparent success, NMT still suffers from one significant drawback: it is difficult to integrate prior knowledge into neural networks. On one hand, neural networks use *continuous* real-valued vectors to represent all language structures involved in the translation process. While these vector representations prove to be capable of capturing translation regularities implicitly (Sutskever et al., 2014), it is hard to interpret each hidden state in neural networks from a linguistic perspective. On the other hand, prior knowledge in machine translation is usually represented in *discrete* symbolic forms such as dictionaries and rules (Nirenburg, 1989) that explicitly encode translation regularities. It is difficult to transform prior knowledge represented in discrete forms to continuous representations required by neural networks.

Therefore, a number of authors have endeavored to integrate prior knowledge into NMT in recent years, either by modifying model architectures (Tu et al., 2016; Cohn et al., 2016; Tang et al., 2016; Feng et al., 2016) or by modifying training objectives (Cohn et al., 2016; Feng et al., 2016; Cheng et al., 2016). For example, to address the over-translation and under-translation problems widely observed in NMT, Tu et al. (2016) directly extend standard NMT to model the coverage constraint that each source phrase should be translated into exactly one target phrase (Koehn et al., 2003). Alternatively, Cohn et al. (2016) and Feng et al. (2016) propose to control the fertilities of source words by appending additional additive terms to training objectives.

Although these approaches have demonstrated clear benefits of incorporating prior knowledge into NMT, how to combine multiple overlapping, arbitrary prior knowledge sources still remains a major challenge. It is difficult to achieve this end by directly modifying model architectures because neural networks usually impose strong independence assumptions between hidden states. As a result, extending a neural model requires that the interdependence of information sources be modeled explicitly (Tu et al., 2016; Tang et al., 2016), making it hard to extend. While this drawback can be partly alleviated by appending additional additive terms to training objectives (Cohn et al., 2016; Feng et al., 2016), these terms are restricted to a

*Corresponding author: Yang Liu.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1514–1523

https://doi.org/10.18653/v1/P17-1139

limited number of simple constraints.

In this work, we propose a general framework for integrating multiple overlapping, arbitrary prior knowledge sources into NMT using *posterior regularization* (Ganchev et al., 2010). Our framework is capable of incorporating indirect supervision via posterior distributions of neural translation models. To represent prior knowledge sources as arbitrary real-valued features, we define the posterior distribution as a log-linear model instead of a constrained posterior set (Ganchev et al., 2010). This treatment not only leads to a simpler and more efficient training algorithm but also achieves better translation performance. Experiments show that our approach is able to incorporate a variety of features and achieves significant improvements over posterior regularization using constrained posterior sets on NIST Chinese-English datasets.

2 Background

2.1 Neural Machine Translation

Given a source sentence $\mathbf{x} = x_1, \ldots, x_i, \ldots, x_I$ and a target sentence $\mathbf{y} = y_1, \ldots, y_j, \ldots, y_J$, a neural translation model (Sutskever et al., 2014; Bahdanau et al., 2015) is usually factorized as a product of word-level translation probabilities:

$$P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \prod_{j=1}^{J} P(y_j|\mathbf{x},\mathbf{y}_{<j};\boldsymbol{\theta}), \quad (1)$$

where $\boldsymbol{\theta}$ is a set of model parameters and $\mathbf{y}_{<j} = y_1, \ldots, y_{j-1}$ denotes a partial translation.

The word-level translation probability is defined using a softmax function:

$$P(y_j|\mathbf{x},\mathbf{y}_{<j};\boldsymbol{\theta}) \propto \exp\Big(f(\mathbf{v}_{y_j}, \mathbf{v}_{\mathbf{x}}, \mathbf{v}_{\mathbf{y}_{<j}}, \boldsymbol{\theta})\Big), \quad (2)$$

where $f(\cdot)$ is a non-linear function, $\mathbf{v}_{y_j}$ is a vector representation of the j-th target word y_j, $\mathbf{v}_{\mathbf{x}}$ is a vector representation of the source sentence $\mathbf{x}$ that encodes the context on the source side, and $\mathbf{v}_{\mathbf{y}_{<j}}$ is a vector representation of the partial translation $\mathbf{y}_{<j}$ that encodes the context on the target side.

Given a training set $\{\langle \mathbf{x}^{(n)}, \mathbf{y}^{(n)} \rangle\}_{n=1}^{N}$, the standard training objective is to maximize the log-likelihood of the training set:

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \Big\{\mathcal{L}(\theta)\Big\}, \quad (3)$$

where

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)};\boldsymbol{\theta}). \quad (4)$$

Although the introduction of vector representations into machine translation has resulted in substantial improvements in terms of translation quality (Junczys-Dowmunt et al., 2016), it is difficult to incorporate prior knowledge represented in discrete symbolic forms into NMT. For example, given a Chinese-English dictionary containing ground-truth translational equivalents such as $\langle$*baigong*, *the White House*$\rangle$, it is non-trivial to leverage the dictionary to guide the learning process of NMT. To address this problem, Tang et al. (2016) propose a new architecture called phraseNet on top of RNNsearch (Bahdanau et al., 2015) that equips standard NMT with an external memory storing phrase tables.

Another important prior knowledge source is the coverage constraint (Koehn et al., 2003): *each source phrase should be translated into exactly one target phrase*. To encode this linguistic intuition into NMT, Tu et al. (2016) extend standard NMT with a coverage vector to keep track of the attention history.

While these approaches are capable of incorporating individual prior knowledge sources separately, how to combine multiple overlapping, arbitrary knowledge sources still remains a major challenge. This can be hardly addressed by modifying model architectures because of the lack of interpretability in NMT and the incapability of neural networks in modeling arbitrary knowledge sources. Although modifying training objectives to include additional knowledge sources as additive terms can partially alleviate this problem, these terms have been restricted to a limited number of simple constraints (Cheng et al., 2016; Cohn et al., 2016; Feng et al., 2016) and incapable of combining arbitrary knowledge sources.

Therefore, it is important to develop a new framework for integrating arbitrary prior knowledge sources into NMT.

2.2 Posterior Regularization

Ganchev et al. (2010) propose *posterior regularization* for incorporating indirect supervision via constraints on posterior distributions of structured latent-variable models. The basic idea is to penalize the log-likelihood of a neural translation model

with the KL divergence between a desired distribution that incorporates prior knowledge and the model posteriors. The posterior regularized likelihood is defined as

$$\begin{aligned} & F(\boldsymbol{\theta}, \boldsymbol{q}) \\ = \; & \lambda_1 \mathcal{L}(\boldsymbol{\theta}) - \\ & \lambda_2 \sum_{n=1}^{N} \min_{q \in \mathcal{Q}} \mathrm{KL}\Big(q(\mathbf{y}) \Big|\Big| P(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta}), \Big) \end{aligned} \quad (5)$$

where λ_1 and λ_2 are hyper-parameters to balance the preference between likelihood and posterior regularization, $\mathcal{Q}$ is a set of constrained posteriors:

$$\mathcal{Q} = \{q(\mathbf{y}) : \mathbb{E}_q[\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})] \leq \mathbf{b}\}, \quad (6)$$

where $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ is constraint feature and b is the bound of constraint feature expectations. Ganchev et al. (2010) use constraint features to encode structural bias and define the set of valid distributions with respect to the expectations of constraint features to facilitate inference.

As maximizing $F(\boldsymbol{\theta}, \boldsymbol{q})$ involves minimizing the KL divergence, Ganchev et al. (2010) present a minorization-maximization algorithm akin to EM at sentence level:

$$\mathrm{E}: q^{(t+1)} = \operatorname*{argmin}_{q} \mathrm{KL}\Big(q(\mathbf{y}) \Big|\Big| P(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta}^{(t)})\Big)$$

$$\mathrm{M}: \boldsymbol{\theta}^{(t+1)} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{q^{(t+1)}}\Big[\log P(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta})\Big]$$

However, directly applying posterior regularization to neural machine translation faces a major difficulty: it is hard to specify the hyper-parameter $\mathbf{b}$ to effectively bound the expectation of features, which are usually real-valued in translation (Och and Ney, 2002; Koehn et al., 2003; Chiang, 2005). For example, the coverage penalty constraint (Wu et al., 2016) proves to be an essential feature for controlling the length of a translation in NMT. As the value of coverage penalty varies significantly over different sentences, it is difficult to set an appropriate bound for all sentences on the training data. In addition, the minorization-maximization algorithm involves an additional step to find $q^{(t+1)}$ as compared with standard NMT, which increases training time significantly.

3 Posterior Regularization for Neural Machine Translation

3.1 Modeling

In this work, we propose to adapt posterior regularization (Ganchev et al., 2010) to neural machine translation. The major difference is that we represent the desired distribution as a log-linear model (Och and Ney, 2002) rather than a constrained posterior set as described in (Ganchev et al., 2010):

$$\begin{aligned} & \mathcal{J}(\boldsymbol{\theta}, \boldsymbol{\gamma}) \\ = \; & \lambda_1 \mathcal{L}(\boldsymbol{\theta}) - \\ & \lambda_2 \sum_{n=1}^{N} \mathrm{KL}\Big(Q(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\gamma}) \Big|\Big| P(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta})\Big), \end{aligned} \quad (7)$$

where the desired distribution that encodes prior knowledge is defined as: [1]

$$Q(\mathbf{y}|\mathbf{x}; \boldsymbol{\gamma}) = \frac{\exp\Big(\boldsymbol{\gamma} \cdot \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})\Big)}{\sum_{\mathbf{y}'} \exp\Big(\boldsymbol{\gamma} \cdot \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}')\Big)}. \quad (8)$$

As compared to previous work on integrating prior knowledge into NMT (Tu et al., 2016; Cohn et al., 2016; Tang et al., 2016), our approach provides a general framework for combining arbitrary knowledge sources. This is due to log-linear models that offer sufficient flexibility to represent arbitrary prior knowledge sources as features. We tackle the representation discrepancy problem by associating the Q distribution that encodes *discrete* representations of prior knowledge with neural models using *continuous* representations learned from data in the KL divergence. Another advantage of our approach is the transparency to model architectures. In principle, our approach can be applied to any neural models for natural language processing.

Our approach also differs from the original version of posterior regularization (Ganchev et al., 2010) in the definition of desired distribution. We resort to log-linear models (Och and Ney, 2002) to incorporate features that have proven effective in SMT. Another benefit of using log-linear models is the differentiability of our training objective (see Eq. (7)). It is easy to leverage standard stochastic gradient descent algorithms to optimize model parameters (Section 3.3).

3.2 Feature Design

In this section, we introduce how to design features to encode prior knowledge in the desired dis-

[1] Ideally, the desired distribution Q should be fixed to guide the learning process of P. However, it is hard to manually specify the feature weights $\boldsymbol{\gamma}$. Therefore, we propose to train both $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ jointly (see Section 3.3). We find that joint training results in significant improvements in practice (see Table 1).

tribution.

Note that not all features in SMT can be adopted to our framework. This is because features in SMT are defined on latent structures such as phrase pairs and synchronous CFG rules, which are not accessible to the decoding process of NMT. Fortunately, we can still leverage internal information in neural models that is linguistically meaningful such as the attention matrix $\mathbf{a}$ (Bahdanau et al., 2015).

We will introduce a number of features used in our experiments as follows.

3.2.1 Bilingual Dictionary

It is natural to leverage a bilingual dictionary $\mathcal{D}$ to improve neural machine translation. Arthur et al. (2016) propose to incorporate discrete translation lexicons into NMT by using the attention vector to select lexical probabilities on which to be focused.

In our work, for each entry $\langle x, y \rangle \in \mathcal{D}$ in the dictionary, a *bilingual dictionary* (BD) feature is defined at the sentence level:

$$\phi_{\mathrm{BD}_{\langle x,y \rangle}}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } x \in \mathbf{x} \wedge y \in \mathbf{y} \\ 0 & \text{otherwise} \end{cases} . \quad (9)$$

Note that number of bilingual dictionary features depends on the vocabulary of the neural translation model. Entries containing out-of-vocabulary words has to be discarded.

3.2.2 Phrase Table

Phrases, which are sequences of consecutive words, are capable of memorizing local context to deal with word ordering within phrases and translation of short idioms, word insertions or deletions (Koehn et al., 2003; Chiang, 2005). As a result, phrase tables that specify phrase-level correspondences between the source and target languages also prove to be an effective knowledge source in NMT (Tang et al., 2016).

Similar to the bilingual dictionary features, we define a *phrase table* (PT) feature for each entry $\langle \tilde{x}, \tilde{y} \rangle$ in a phrase table $\mathcal{P}$:

$$\phi_{\mathrm{PT}_{\langle \tilde{x}, \tilde{y} \rangle}}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } \tilde{x} \in \mathbf{x} \wedge \tilde{y} \in \mathbf{y} \\ 0 & \text{otherwise} \end{cases} . \quad (10)$$

The number of phrase table features also depends on the vocabulary of the neural translation model.

3.2.3 Coverage Penalty

To overcome the over-translation and under-translation problems widely observed in NMT, a number of authors have proposed to model the fertility (Brown et al., 1993) and converge constraint (Koehn et al., 2003) to improve the adequacy of translation (Tu et al., 2016; Cohn et al., 2016; Feng et al., 2016; Wu et al., 2016; Mi et al., 2016).

We follow Wu et al. (2016) to define a *coverage penalty* (CP) feature to penalize source words with lower sum of attention weights: [2]

$$\phi_{\mathrm{CP}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{|\mathbf{x}|} \log \left(\min \left(\sum_{j=1}^{|\mathbf{y}|} \mathbf{a}_{i,j}, 1.0 \right) \right), \quad (11)$$

where $\mathbf{a}_{i,j}$ is the attention probability of the j-th target word on the i-th source word. Note that the value of coverage penalty feature varies significantly over sentences of different lengths.

3.2.4 Length Ratio

Controlling the length of translations is very important in NMT as neural models tend to generate short translations for long sentences, which deteriorates the translation performance of NMT for long sentences as compared with SMT (Shen et al., 2016).

Therefore, we define the *length ratio* (LR) feature to encourage the length of a translation to fall in a reasonable range:

$$\phi_{\mathrm{LR}}(\mathbf{x}, \mathbf{y}) = \begin{cases} (\beta|\mathbf{x}|)/|\mathbf{y}| & \text{if } \beta|\mathbf{x}| < |\mathbf{y}| \\ |\mathbf{y}|/(\beta|\mathbf{x}|) & \text{otherwise} \end{cases} , \quad (12)$$

where β is a hyper-parameter for penalizing too long or too short translations.

For example, to convey the same meaning, an English sentence is usually about 1.2 times longer than a Chinese sentence. As a result, we can set $\beta = 1.2$. If the length of a Chinese sentence $|\mathbf{x}|$ is 10 and the length of an English sentence $|\mathbf{y}|$ is 12, then, $\phi_{\mathrm{LR}}(\mathbf{x}, \mathbf{y}) = 1$. If the translation is too long (e.g., $|\mathbf{y}| = 100$), then the feature value is 0.12. If the translation is too short (e.g., $|\mathbf{y}| = 6$), the feature value is 0.5.

3.3 Training

In training, our goal is to find a set of model parameters that maximizes the posterior regularized likelihood:

$$\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}} = \underset{\boldsymbol{\theta}, \boldsymbol{\gamma}}{\mathrm{argmax}} \left\{ \mathcal{J}(\boldsymbol{\theta}, \boldsymbol{\gamma}) \right\}. \quad (13)$$

[2] For simplicity, we omit the attention matrix $\mathbf{a}$ in the input of the coverage feature function.

Note that unlike the original version of posterior regularization (Ganchev et al., 2010) that relies on a minorization-maximization algorithm to optimize model parameters, our training objective is differentiable with respect to model parameters. Therefore, it is easy to use standard stochastic gradient descent algorithms to train our model.

However, a major difficulty in calculating gradients is that the algorithm needs to sum over all candidate translations in an exponential search space for KL divergence. For example, the partial derivative of $\mathcal{J}(\boldsymbol{\theta}, \boldsymbol{\gamma})$ with respect to γ is given by

$$\begin{aligned}&\frac{\partial \mathcal{J}(\boldsymbol{\theta}, \boldsymbol{\gamma})}{\partial \boldsymbol{\gamma}} \\ &= -\lambda_2 \times \\ &\sum_{n=1}^{N} \frac{\partial}{\partial \boldsymbol{\gamma}} \mathrm{KL}\Big(Q(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\gamma}) \Big|\Big| P(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta})\Big). \quad (14)\end{aligned}$$

The KL divergence is defined as

$$\begin{aligned}&\mathrm{KL}\Big(Q(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\gamma}) \Big|\Big| P(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta})\Big) \\ &= \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}^{(n)})} Q(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\gamma}) \log \frac{Q(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\gamma})}{P(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta})}, \quad (15)\end{aligned}$$

where $\mathcal{Y}(\mathbf{x}^{(n)})$ is a set of all possible candidate translations for the source sentence $\mathbf{x}^{(n)}$.

To alleviate this problem, we follow Shen et al. (2016) to approximate the full search space $\mathcal{Y}(\mathbf{x}^{(n)})$ with a sampled sub-space $\mathcal{S}(\mathbf{x}^{(n)})$. Therefore, the KL divergence can be approximated as

$$\begin{aligned}&\mathrm{KL}\Big(Q(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\gamma}) \Big|\Big| P(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta})\Big) \\ &\approx \sum_{\mathbf{y} \in \mathcal{S}(\mathbf{x}^{(n)})} \tilde{Q}(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\gamma}) \log \frac{\tilde{Q}(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\gamma})}{\tilde{P}(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta})}. \quad (16)\end{aligned}$$

Note that the Q distribution is also approximated on the sub-space:

$$\begin{aligned}&\tilde{Q}(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\gamma}) \\ &= \frac{\exp(\boldsymbol{\gamma} \cdot \boldsymbol{\phi}(\mathbf{x}^{(n)}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{S}(\mathbf{x}^{(n)})} \exp(\boldsymbol{\gamma} \cdot \boldsymbol{\phi}(\mathbf{x}^{(n)}, \mathbf{y}'))}. \quad (17)\end{aligned}$$

We follow Shen et al. (2016) to control the sharpness of approximated neural translation distribution normalized on the sampled sub-space:

$$\tilde{P}(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta}) = \frac{P(\mathbf{y}|\mathbf{x}^{(n)}; \boldsymbol{\theta})^{\alpha}}{\sum_{\mathbf{y}' \in \mathcal{S}(\mathbf{x}^{(n)})} P(\mathbf{y}'|\mathbf{x}^{(n)}; \boldsymbol{\theta})^{\alpha}}. \quad (18)$$

3.4 Search

Given learned model parameters $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\gamma}}$, the decision rule for translating an unseen source sentence $\mathbf{x}$ is given by

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathcal{Y}(\mathbf{x})} \Big\{ P(\mathbf{y}|\mathbf{x}; \hat{\boldsymbol{\theta}}) \Big\}. \quad (19)$$

The search process can be factorized at the word level:

$$\hat{y}_j = \operatorname*{argmax}_{y \in \mathcal{V}_y} \Big\{ P(y|\mathbf{x}, \hat{\mathbf{y}}_{<j}; \hat{\boldsymbol{\theta}}) \Big\}, \quad (20)$$

where $\mathcal{V}_y$ is the target language vocabulary.

Although this decision rule shares the same efficiency and simplicity with standard NMT (Bahdanau et al., 2015), it does not involve prior knowledge in decoding. Previous studies reveal that incorporating prior knowledge in decoding also significantly boosts translation performance (Arthur et al., 2016; He et al., 2016; Wang et al., 2016).

As directly incorporating prior knowledge into the decoding process of NMT depends on both model structure and the locality of features, we resort to a coarse-to-fine approach instead to keep the architecture transparency of our approach. Given a source sentence $\mathbf{x}$ in the test set, we first use the neural translation model $P(\mathbf{y}|\mathbf{x}; \hat{\boldsymbol{\theta}})$ to generate a k-best list of candidate translation $\mathcal{C}(\mathbf{x})$. Then, the algorithm decides on the most probable candidate translation using the following decision rule:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{C}(\mathbf{x})} \Big\{ \log P(\mathbf{y}|\mathbf{x}; \hat{\boldsymbol{\theta}}) + \hat{\boldsymbol{\gamma}} \cdot \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \Big\}. \quad (21)$$

4 Experiments

4.1 Setup

We evaluate our approach on Chinese-English translation. The evaluation metric is case-insensitive BLEU calculated by the *multi-bleu.perl* script. Our training set[3] consists of 1.25M sentence pairs with 27.9M Chinese words and 34.5M English words. We use the NIST 2002 dataset as validation set and the NIST 2003, 2004, 2005, 2006, 2008 datasets as test sets.

In the experiments, we compare our approach with the following two baseline approaches:

[3]The training set includes LDC2002E18, LDC2003E07, LDC2003E14, part of LDC2004T07, LDC2004T08 and LDC2005T06.

Method	Feature	MT02	MT03	MT04	MT05	MT06	MT08	All
RNNSEARCH	N/A	33.45	30.93	32.57	29.86	29.03	21.85	29.11
CPR	N/A	33.84	31.18	33.26	30.67	29.63	22.38	29.72
POSTREG	BD	34.65	31.53	33.82	30.66	29.81	22.55	29.97
	PT	34.56	31.32	33.89	30.70	29.84	22.62	29.99
	LR	34.39	31.41	34.19	30.80	29.82	22.85	30.14
	BD+PT	34.66	32.05	34.54	31.22	30.70	22.84	30.60
	BD+PT+LR	34.37	31.42	34.18	30.99	29.90	22.87	30.20
this work	BD	**36.61**	33.47	36.04	32.96	32.46	**24.78**	32.27
	PT	35.07	32.11	34.73	31.84	30.82	23.23	30.86
	CP	34.68	31.99	34.67	31.37	30.80	23.34	30.76
	LR	34.57	31.89	34.95	31.80	31.43	23.75	31.12
	BD+PT	36.30	**33.83**	36.02	32.98	32.53	24.54	32.29
	BD+PT+CP	36.11	33.64	36.36	**33.11**	32.53	24.57	32.39
	BD+PT+CP+LR	36.10	33.64	**36.48**	33.08	**32.90**	24.63	**32.51**

Table 1: Comparison of BLEU scores on the Chinese-English datasets. RNNSEARCH is an attention-based neural machine translation model (Bahdanau et al., 2015) that does not incorporate prior knowledge. CPR extends RNNSEARCH by introducing coverage penalty refinement (Eq. (11)) in decoding. POSTREG extends RNNSEARCH with posterior regularization (Ganchev et al., 2010), which uses constraint features to represent prior knowledge and a constrained posterior set to denote the desired distribution. Note that POSTREG cannot use the CP feature (Section 3.2.3) because it is hard to bound the feature value appropriately. On top of RNNSEARCH, our approach also exploits posterior regularization to incorporate prior knowledge but uses a log-linear model to denote the desired distribution. All results of *this work* are significantly better than RNNSEARCH ($p < 0.01$).

1. RNNSEARCH (Bahdanau et al., 2015): a standard attention-based neural machine translation model,

2. CPR (Wu et al., 2016): extending RNNSEARCH by introducing coverage penalty refinement (Eq. (11)) in decoding,

3. POSTREG (Ganchev et al., 2010): extending RNNSEARCH with posterior regularization using constrained posterior set.

For RNNSEARCH, we use an in-house attention-based NMT system that achieves comparable translation performance with GROUNDHOG (Bahdanau et al., 2015), which serves as a baseline approach in our experiments. We limit vocabulary size to 30K for both languages. The word embedding dimension is set to 620. The dimension of hidden layer is set to 1,000. In training, the batch size is set to 80. We use the AdaDelta algorithm (Zeiler, 2012) for optimizing model parameters. In decoding, the beam size is set to 10.

For CPR, we simply follow Wu et al. (2016) to incorporate the coverage penalty into the beam search algorithm of RNNSEARCH.

For POSTREG, we adapt the original version of posterior regularization (Ganchev et al., 2010) to NMT on top of RNNSEARCH. Following Ganchev et al. (2010), we use a ten-step projected gradient descent algorithm to search for an approximate desired distribution in the E step and a one-step gradient descent for the M step.

Our approach extends RNNSEARCH by incorporating prior knowledge. For each source sentence, we sample 80 candidate translations to approximate the $\tilde{P}$ and $\tilde{Q}$ distributions. The hyper-parameter α is set to 0.2. The batch size is 1. The hyper-parameters λ_1 and λ_2 are set to 8×10^{-5} and 2.5×10^{-4}. Note that they not only balance the preference between likelihood and posterior regularization, but also control the values of gradients to fall in a reasonable range for optimization.

We construct bilingual dictionary and phrase table in an automatic way. First, we run the statistical machine translation system MOSES (Koehn and Hoang, 2007) to obtain probabilistic bilingual dictionary and phrase table. For the bilingual dictionary, we retain entries with probabilities higher than 0.1 in both source-to-target and

Feature	Rerank	MT02	MT03	MT04	MT05	MT06	MT08	All
BD	w/o	36.06	32.99	35.62	32.59	32.13	24.36	31.87
	w/	**36.61**	33.47	36.04	32.96	32.46	**24.78**	32.27
PT	w/o	34.98	32.01	34.71	31.77	30.77	23.20	30.81
	w/	35.07	32.11	34.73	31.84	30.82	23.23	30.86
CP	w/o	34.68	31.99	34.67	31.37	30.80	23.34	30.76
	w/	34.68	31.99	34.67	31.37	30.80	23.34	30.76
LR	w/o	34.60	31.89	34.79	31.72	31.39	23.63	31.03
	w/	34.57	31.89	34.95	31.80	31.43	23.75	31.12
BD+PT	w/o	35.76	33.27	35.64	32.47	32.03	24.17	31.83
	w/	36.30	**33.83**	36.02	32.98	32.53	24.54	32.29
BD+PT+CP	w/o	35.71	33.15	35.81	32.52	32.16	24.11	31.89
	w/	36.11	33.64	36.36	**33.11**	32.53	24.57	32.39
BD+PT+CP+LR	w/o	36.06	33.01	35.86	32.70	32.24	24.27	31.96
	w/	36.10	33.64	**36.48**	33.08	**32.90**	24.63	**32.51**

Table 2: Effect of reranking on translation quality.

target-to-source directions. For phrase table, we first remove phrase pairs that occur less than 10 times and then retain entries with probabilities higher than 0.5 in both directions. As a result, both bilingual dictionary and phrase table contain high-quality translation correspondences. We estimate the length ratio on Chinese-English data and set the hyper-parameter β to 1.236.

By default, both POSTREG and our approach use reranking to search for the most probable translations (Section 3.4).

4.2 Main Results

Table 1 shows the BLEU scores obtained by RNNSEARCH, POSTREG, and our approach on the Chinese-English datasets.

We find POSTREG achieves significant improvements over RNNSEARCH by adding features that encode prior knowledge. The most effective single feature for POSTREG seems to be the length ratio (LR) feature, suggesting that it is important for NMT to control the length of translation to improve translation quality. Note that POSTREG is unable to include the coverage penalty (CP) feature because the feature value varies significantly over different sentences. It is hard to specify an appropriate bound $\mathbf{b}$ for constraining the expected feature value. We observe that a loose bound often makes the training process very unstable and fail to converge. Combining features obtains further modest improvements.

Our approach outperforms both RNNSEARCH and POSTREG significantly. The bilingual dictionary (BD) feature turns out to make the most contribution. Compared with CPR that imposes coverage penalty during decoding, our approach that using a single CP feature obtains a significant improvement (i.e., 30.76 over 29.72), suggesting that incorporating prior knowledge sources in modeling might be more beneficial than in decoding.

We find that combining features only results in modest improvements for our approach. One possible reason is that the bilingual dictionary and phrase table features overlap on single word pairs.

4.3 Effect of Reranking

Table 2 shows the effect of reranking on translation quality. We find that using prior knowledge features to rescore the k-best list produced by the neural translation model usually leads to improvements. This finding confirms that adding prior knowledge is beneficial for NMT, either in the training or decoding process.

4.4 Training Speed

Initialized with the best RNNSEARCH model trained for 300K iterations, our model converges after about 100K iterations. For each iteration, our approach is 1.5 times slower than RNNSEARCH. On a single GPU device Tesla M40, it takes four days to train the RNNSEARCH model and three extra days to train our model.

4.5 Example Translations

Table 3 gives four examples to demonstrate the benefits of adding features.

Source	*lijing liang tian yu bingxue de **fenzhan** , 31ri shenye 23 shi 50 fen , shanghai jichang jituan **yuangong** **yinglai** le 2004nian de zuihou yige hangban .*
Reference	after **fighting** with ice and snow for two days , **staff** members of shanghai airport group **welcomed** the last flight of 2004 at 23 : 50pm on the 31st .
RNNSEARCH	after a two - day and two - day journey , the team of shanghai 's airport in shanghai has ushered in the last flight in 2004 .
+ BD	after two days and nights **fighting** with ice and snow , the shanghai airport group 's **staff** **welcomed** the last flight in 2004 .
Source	*suiran tonghuopengzhang **weilai ji ge yue** reng jiang weizhi zai **baifenzhier yishang** , buguo niandi zhiqian keneng jiangdi .*
Reference	although inflation will remain **above 2 % for the coming few months** , it may decline by the end of the year .
RNNSEARCH	although inflation has been maintained for more than two months from the year before the end of the year , it may be lower .
+ PT	although inflation will remain at **more than 2 percent in the next few months** , it may be lowered before the end of the year .
Source	*qian ji tian ta ganggang **chuyuan** , jintian **jianchi** lai yu lao pengyou **daobie** .*
Reference	just **discharged from the hospital** a few days ago , he **insisted on** coming to **say farewell** to his old friend today .
RNNSEARCH	during the previous few days , he had just been given treatment to the old friends .
+ CP	during the previous few days , he had just been **discharged from the hospital** , and he **insisted on goodbye** to his old friend today .
Source	*(guoji) yiselie fuzongli fouren jihua kuojian **gelan gaodi** dingjudian*
Reference	(international) israeli deputy prime minister denied plans to expand **golan heights** settlements
RNNSEARCH	(world) israeli deputy prime minister denies the plan to expand **the golan heights** in **the golan heights**
+ LR	(international) israeli deputy prime minister denies planning to expand **golan heights**

Table 3: Example translations that demonstrate the effect of adding features.

In the first example, source words "fenzhan" (*fighting*), "yuangong" (*staff*), and "yinglai" (*welcomed*) are untranslated in the output of RNNSEARCH. Adding the bilingual dictionary (BD) feature encourages the model to translate these words if they occur in the dictionary.

In the second example, while RNNSEARCH fails to capture phrase cohesion, adding the phrase table (PT) feature is beneficial for translating short idioms, word insertions or deletions that are sensitive to local context.

In the third example, RNNSEARCH tends to omit many source content words such as "chuyuan" (*discharged from the hospital*), "jianchi" (*insisted on*), and "daobie" (*say farewell*). The coverage penalty (CP) feature helps to alleviate the word omission problem.

In the fourth example, the translation produced by RNNSEARCH is too long and "the golan heights" occurs twice. The length ratio (LR) feature is capable of controlling the sentence length in a reasonable range.

5 Related Work

Our work is directly inspired by posterior regularization (Ganchev et al., 2010). The major difference is that we use a log-linear model to represent the desired distribution rather than a constrained posterior set. Using log-linear models not only enables our approach to incorporate arbitrary knowledge sources as real-valued features, but also is differentiable to be jointly trained with

neural translation models efficiently.

Our work is closely related to recent work on injecting prior knowledge into NMT (Arthur et al., 2016; Tu et al., 2016; Cohn et al., 2016; Tang et al., 2016; Feng et al., 2016; Wang et al., 2016). The major difference is that our approach aims to provide a general framework for incorporating arbitrary prior knowledge sources while keeping the neural translation model unchanged.

He et al. (2016) also propose to combine the strengths of neural networks on learning representations and log-linear models on encoding prior knowledge. But they treat neural translation models as a feature in the log-linear model. In contrast, we connect the two models via KL divergence to keep the transparency of our approach to model architectures. This enables our approach to be easily applied to other neural models in NLP.

6 Conclusion

We have presented a general framework for incorporating prior knowledge into end-to-end neural machine translation based on posterior regularization (Ganchev et al., 2010). The basic idea is to guide NMT models towards desired behavior using a log-linear model that encodes prior knowledge. Experiments show that incorporating prior knowledge leads to significant improvements over both standard NMT and posterior regularization using constrained posterior sets.

Acknowledgments

We thank Shiqi Shen for useful discussions and anonymous reviewers for insightful comments. This work is supported by the National Natural Science Foundation of China (No.61432013), the 973 Program (2014CB340501), and the National Natural Science Foundation of China (No.61522204). This research is also supported by Sogou Inc. and the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme.

References

Philip Arthur, Graham Neubug, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. arXiv:1606.02006v2.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics* .

Yong Cheng, Shiqi Shen, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Agreement-based learning of parallel lexicons and phrases from non-parallel corpora. In *Proceedings of IJCAI*.

David Chiang. 2005. A hirarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*.

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

Shi Feng, Shujie Liu, Nan Yang, Mu Li, Ming Zhou, and Kenny Q. Zhu. 2016. Improving attention modeling with implicit distortion and fertility for machine translation. In *Proceedings of COLING*.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research* .

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved nerual machine translation with SMT features. In *Proceedings of AAAI*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* .

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? a case study on 30 translation directions. arXiv:1610.01108v2.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of EMNLP*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proceedings of EMNLP*.

Sergei Nirenburg. 1989. Knowledge-based machine translation. *Machine Translation* .

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Yaohua Tang, Fandong Meng, Zhengdong Lu, Hang Li, and Philip L. H. Yu. 2016. Neural machine translation with external phrase memory. arXiv:1606.01792v1.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL*.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2016. Neural machine translation advised by statistical machine translation. arXiv:1610.05150.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144v2.

Matthew D. Zeiler. 2012. Adadelta: an adaptive learning rate method. arXiv:1212.5701.

Incorporating Word Reordering Knowledge into Attention-based Neural Machine Translation

Jinchao Zhang[1] **Mingxuan Wang**[1] **Qun Liu**[3,1] **Jie Zhou**[2]
[1]Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
{zhangjinchao,wangmingxuan,liuqun}@ict.ac.cn
[2]Baidu Research - Institute of Deep Learning,
Baidu Inc.,Beijing,China
{zhoujie01}@baidu.com
[3]ADAPT Centre, School of Computing, Dublin City University

Abstract

This paper proposes three distortion models to explicitly incorporate the word reordering knowledge into attention-based Neural Machine Translation (NMT) for further improving translation performance. Our proposed models enable attention mechanism to attend to source words regarding both the semantic requirement and the word reordering penalty. Experiments on Chinese-English translation show that the approaches can improve word alignment quality and achieve significant translation improvements over a basic attention-based NMT by large margins. Compared with previous works on identical corpora, our system achieves the state-of-the-art performance on translation quality.

1 Introduction

Word reordering model is one of the most crucial sub-components in Statistical Machine Translation (SMT) (Brown et al., 1993; Koehn et al., 2003; Chiang, 2005) which provides word reordering knowledge to ensure reasonable translation order of source words. It is separately trained and then incorporated into the SMT framework in a pipeline style.

In recent years, end-to-end NMT (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015) has made tremendous progress (Jean et al., 2015; Luong et al., 2015b; Shen et al., 2016; Sennrich et al., 2016; Tu et al., 2016; Zhou et al., 2016; Johnson et al., 2016). An encoder-decoder framework (Cho et al., 2014b; Sutskever et al., 2014) with attention mechanism (Bahdanau et al., 2015) is widely used, in which an encoder compresses the source sentence, an attention mechanism evaluates related source words and a decoder generates target words.

The attention mechanism evaluates the distribution of to-be-translated source words in a content-based addressing fashion (Graves et al., 2014) which tends to attend to the source words regarding the content relation with current translation status. Lack of explicit models to exploit the word reordering knowledge may lead to attention faults and generate fluent but inaccurate or inadequate translations. Table 1 shows a translation instance and Figure 1 depicts the corresponding word alignment matrix that produced by the attention mechanism. In this example, even though the word "zuixin (latest)" is a common adjective in Chinese and its following word should be translated soon in Chinese to English translation direction, the word "yiju (evidence)" does not obtain appropriate attention which leads to the incorrect translation.

src	youguan(related) baodao(report) shi(is) zhichi(support) tamen(their) lundian(arguments) de('s) zuixin(latest) yiju(evidence) .
ref	the report is the latest evidence that supports their arguments .
NMT	the report supports their perception of the latest .
count	zuixin yiju {0}

Table 1: An instance in Chinese-English translation task. The row "count" represents the frequency of the word collocation in the training corpus. The collocation "zuixin yiju" does not appear in the training data.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1524–1534
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1140

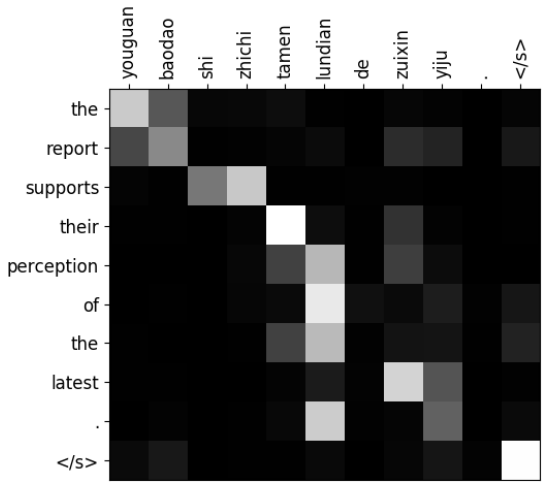


Figure 1: The source word "yiju" does not obtain appropriate attention and its word sense is completely neglected.

To enhance the attention mechanism, implicit word reordering knowledge needs to be incorporated into attention-based NMT. In this paper, we introduce three distortion models that originated from SMT (Brown et al., 1993; Koehn et al., 2003; Och et al., 2004; Tillmann, 2004; Al-Onaizan and Papineni, 2006), so as to model the word reordering knowledge as the probability distribution of the relative jump distances between the newly translated source word and the to-be-translated source word. Our focus is to extend the attention mechanism to attend to source words regarding both the semantic requirement and the word reordering penalty.

Our models have three merits:

1. *Extended word reordering knowledge.* Our models capture explicit word reordering knowledge to guide the attending process for attention mechanism.

2. *Convenient to be incorporated into attention-based NMT.* Our distortion models are differentiable and can be trained in the end-to-end style. The interpolation approach ensures that the proposed models can coordinately work with the original attention mechanism.

3. *Flexible to utilize variant context for computing the word reordering penalty.* In this paper, we exploit three categories of information as distortion context conditions to compute the word reordering penalty, but variant context information can be utilized due to our model's flexibility.

We validate our models on the Chinese-English translation task and achieve notable improvements:

- On 16K vocabularies, NMT models are usually inferior in comparison with the phrase-based SMT, but our model surpasses phrase-based Moses by average **4.43** BLEU points and outperforms the attention-based NMT baseline system by **5.09** BLEU points.

- On 30K vocabularies, the improvements over the phrase-based Moses and the attention-based NMT baseline system are average **6.06** and **1.57** BLEU points respectively.

- Compared with previous work on identical corpora, we achieve the state-of-the-art translation performance on average.

The word alignment quality evaluation shows that our model can effectively improve the word alignment quality that is crucial for improving translation quality.

2 Background

We aim to capture word reordering knowledge for the attention-based NMT by incorporating distortion models. This section briefly introduces attention-based NMT and distortion models in SMT.

2.1 Attention-based Neural Machine Translation

Formally, given a source sentence $\mathbf{x} = \mathbf{x}_1, ..., \mathbf{x}_m$ and a target sentence $\mathbf{y} = \mathbf{y}_1, ..., \mathbf{y}_n$, NMT models the translation probability as

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{n} P(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{x}), \quad (1)$$

where $\mathbf{y}_{<t} = \mathbf{y}_1, ..., \mathbf{y}_{t-1}$. The generation probability of $\mathbf{y}_t$ is

$$P(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{x}) = g(\mathbf{y}_{t-1}, \mathbf{c}_t, \mathbf{s}_t), \quad (2)$$

where $g(\cdot)$ is a softmax regression function, $\mathbf{y}_{t-1}$ is the newly translated target word and

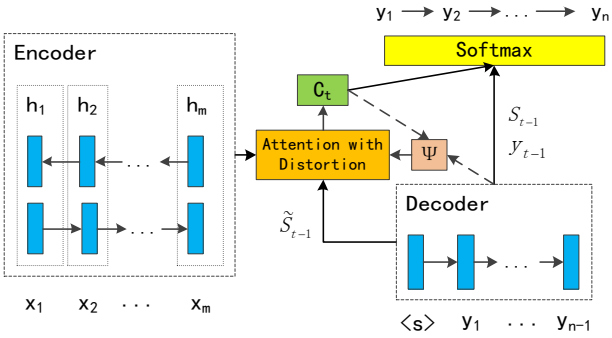


Figure 2: The general architecture of our proposed models. The dash line represents variant context can be utilized to determine the word reordering penalty.

$\mathbf{s}_t$ is the hidden states of decoder which represents the translation status.

The attention $\mathbf{c}_t$ denotes the related source words for generating $\mathbf{y}_t$ and is computed as the weighted-sum of source representation $\mathbf{h}$ upon an alignment vector α_t shown in Eq.(3) where the $align(\cdot)$ function is a feedforward network with $softmax$ normalization.

$$\begin{aligned} \mathbf{c}_t &= \sum_{j=1}^{m} \alpha_{t,j}\mathbf{h}_j \\ \alpha_{t,j} &= align(\mathbf{s}_t, \mathbf{h}_j) \end{aligned} \tag{3}$$

The hidden states $\mathbf{s}_t$ is updated as

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c}_t), \tag{4}$$

where $f(\cdot)$ is a recurrent function.

We adopt a varietal attention mechanism[1] in our in-house RNNsearch model which is implemented as

$$\begin{aligned} \widetilde{\mathbf{s}}_t &= f_1(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}), \\ \alpha_{t,j} &= align(\widetilde{\mathbf{s}}_t, \mathbf{h}_j), \\ \mathbf{s}_t &= f_2(\widetilde{\mathbf{s}}_t, \mathbf{c}_t), \end{aligned} \tag{5}$$

where $f_1(\cdot)$ and $f_2(\cdot)$ are recurrent functions.

As shown in Eq.(3), the attention mechanism attends to source words in a content-based addressing way without considering any explicit word reordering knowledge. We introduce distortion models to capture explicit word reordering knowledge for enhancing the attention mechanism and improving translation quality.

[1]https://github.com/nyu-dl/dl4mt-tutorial/tree/master/session2

2.2 Distortion Models in SMT

In SMT, distortion models are linearly combined with other features, as follows,

$$\begin{aligned} \mathbf{y}^* = \arg\max_{\mathbf{y}} \exp[&\lambda_d d(\mathbf{x}, \mathbf{y}, b) + \\ &\sum_{r=1}^{R-1} \lambda_r h_r(\mathbf{x}, \mathbf{y}, b)], \end{aligned} \tag{6}$$

where $d(\cdot)$ is the distortion feature, $h_r(\cdot)$ represents other features, λ_d and λ_r are the weights, b is the latent variable that represents translation knowledge and R is the number of features.

IBM Models (Brown et al., 1993) depicted the word reordering knowledge as positional relations between source and target words. Koehn et al. (2003) proposed a distortion model for phrase-based SMT based on jump distances between the newly translated phrases and to-be-translated phrases which does not consider specific lexical information. Och et al. (2004) and Tillmann (2004) proposed orientation-based distortion models that consider translation orientations. Yaser and Papineni (2006) proposed a distortion model to estimate probability distribution on possible relative jumps conditioned on source words.

These models are proposed for SMT and separately trained as sub-components. Inspired by these previous work, we introduce the distortion models into NMT model for modeling the word reordering knowledge. Our proposed models are designed for NMT which can be trained in the end-to-end style.

3 Distortion Models for attention-based NMT

The basic idea of our proposed distortion models is to estimate the probability distribution of the possible relative jump distances between the newly translated source word and the to-be-translated source word upon the context condition. Figure 2 shows the general architecture of our proposed model.

3.1 General Architecture

We employ an interpolation approach to incorporate distortion models into attention-based NMT as

$$\alpha_t = \lambda \cdot \mathbf{d}_t + (1-\lambda)\hat{\alpha}_t, \tag{7}$$

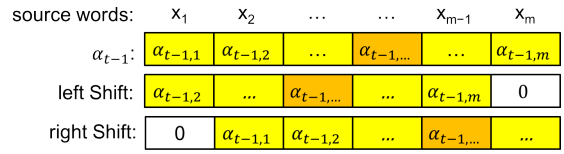


Figure 3: Illustration of shift actions of the alignment vector α_{t-1}. If α_t is the left shift of α_{t-1}, it represents the translation orientation of the source sentence is backward and if α_t is the right shift of α_{t-1}, the translation orientation is forward.

where α_t is the ultimate alignment vector for computing the related source context $\mathbf{c}_t$, $\mathbf{d}_t$ is the alignment vector calculated by the distortion model, $\hat{\alpha}_t$ is the alignment vector computed by the basic attention mechanism and λ is a hyper-parameter to control the weight of the distortion model.

In the proposed distortion model, relative jumps on source words are depicted as the "shift" actions of the alignment vector α_{t-1} which is shown in the Figure 3. The right shift of α_{t-1} indicates that the translation orientation of source words is forward and the left shift represents that the translation orientation is backward. The extent of a shift action measures the word reordering distance. Alignment vector $\mathbf{d}_t$, which is produced by the distortion model, is the expectation of all possible shifts of α_{t-1} conditioned on certain context.

Formally, the proposed distortion model is

$$\begin{aligned}\mathbf{d}_t &= E[\Gamma(\alpha_{t-1})] \\ &= \sum_{k=-l}^{l} P(k|\Psi) \cdot \Gamma(\alpha_{t-1}, k),\end{aligned} \tag{8}$$

where $k \in [-l, l]$ is the possible relative jump distance, l is the window size parameter and $P(k|\Psi)$ stands for the probability of jump distance k that conditioned on the context Ψ. Function $\Gamma(\cdot)$ for shifting the alignment vector is defined as

$$\Gamma(\alpha_{t-1}, k) = \begin{cases} \{\alpha_{t-1,-k}, ..., \alpha_{t-1,m}, 0, ..., 0\}, & \text{k<0} \\ \alpha_{t-1}, & \text{k= 0} \\ \{0, ..., 0, \alpha_{t-1,1}, ..., \alpha_{t-1,m-k}\}, & \text{k>0} \end{cases} \tag{9}$$

which can be implemented as matrix multiplication computations.

We respectively exploit source context, target context and translation status context (hidden states of decoder) as Ψ and derive three distortion models: Source-based Distortion (**S-Distortion**) model , Target-based Distortion (**T-Distortion**) model and Translation-status-based Distortion (**H-Distortion**) model. Our framework is capable of utilizing arbitrary context as the condition Ψ to predict the relative jump distances.

3.2 S-Distortion model

S-Distortion model adopts previous source context $\mathbf{c}_{t-1}$ as the context Ψ with the intuition that certain source word indicate certain jump distance. The to-be-translated source word have intense positional relations with the newly translated one.

The underlying linguistic intuition is that synchronous grammars (Yamada and Knight, 2001; Galley et al., 2004) can be extracted from language pairs. Word categories such as verb, adjective and preposition carry general word reordering knowledge and words carry specific word reordering knowledge.

To further illustrate this idea, we present some common synchronous grammar rules that can be extracted from the example in Table 1 as follows,

$$\begin{aligned} &NP \longrightarrow JJ \quad NN \,|\, JJ \quad NN \\ &JJ \longrightarrow zuixin \,|\, latest. \end{aligned} \tag{10}$$

From the above grammar, we can conjecture the speculation that after the word "zuixin(latest)" is translated, the translation orientation is forward with shift distance 1.

The probability function in S-Distortion model is defined as follows,

$$\begin{aligned} P(\cdot|\Psi) &= z(\mathbf{c}_{t-1}) \\ &= softmax(W_c \mathbf{c}_{t-1} + b_c), \end{aligned} \tag{11}$$

where $W_c \in \mathbb{R}^{(2l+1)\times dim(c_{t-1})}$ and $b_c \in \mathbb{R}^{2l+1}$ are weight matrix and bias parameters.

3.3 T-Distortion Model

T-Distortion model exploits the embedding of the previous generated target word $\mathbf{y}_{t-1}$ as the context condition to predict the probability distribution of distortion distances. It focuses on the word reordering knowledge upon target

word context. As illustrated in Eq.(10), the target word "latest" possesses word reordering knowledge that is identical with source word "zuixin".

The probability function in T-Distortion model is defined as follows,

$$\begin{aligned} P(\cdot|\Psi) &= z(\mathbf{y}_{t-1}) \\ &= softmax(W_y emb(\mathbf{y}_{t-1}) + b_{\mathbf{y}}), \end{aligned} \tag{12}$$

where $emb(\mathbf{y}_{t-1})$ is the embedding of $\mathbf{y}_{t-1}$, $W_y \in \mathbb{R}^{(2l+1)\times dim(emb(\mathbf{y}_{t-1}))}$ and $b_{\mathbf{y}} \in \mathbb{R}^{2l+1}$ are weight matrix and bias parameters.

3.4 H-Distortion Model

The hidden states $\tilde{s}_{t-1}$ reflect the translation status and contains both source context and target context information. Therefore, we exploit $\tilde{s}_{t-1}$ as context Ψ in the H-Distortion model to predict shift distances.

The probability function in H-Distortion model is defined as follows,

$$\begin{aligned} P(\cdot|\Psi) &= z(\tilde{s}_{t-1}) \\ &= softmax(W_s \tilde{s}_{t-1} + b_s) \end{aligned} \tag{13}$$

where $W_s \in \mathbb{R}^{(2l+1)\times dim(\tilde{s}_{t-1})}$ and $b_s \in \mathbb{R}^{2l+1}$ are the weight matrix and bias parameters.

4 Experiments

We carry the translation task on the Chinese-English direction to evaluate the effectiveness of our models. To investigate the word alignment quality, we take the word alignment quality evaluation on the manually aligned corpus. We also conduct the experiments to observe effects of hyper-parameters and the training strategies.

4.1 Data and Metrics

Data: Our Chinese-English training corpus consists of 1.25M sentence pairs extracted from LDC corpora[2] with 27.9M Chinese words and 34.5M English words respectively. 16K vocabularies cover approximately 95.8% and 98.3% words and 30K vocabularies cover approximately 97.7% and 99.3% words in Chinese and English respectively. We choose NIST 2002 dataset as the validation set. NIST 2003-2006 are used as test sets. To assess the word alignment quality, we employ Tsinghua dataset (Liu and Sun, 2015) which contains 900 manually aligned sentence pairs.

Metrics: The translation quality evaluation metric is the case-insensitive 4-gram BLEU[3] (Papineni et al., 2002). *Sign-test* (Collins et al., 2005) is exploited for statistical significance test. Alignment error rate (AER) (Och and Ney, 2003) is calculated to assess the word alignment quality.

4.2 Comparison Systems

We compare our approaches with three baseline systems:

Moses (Koehn et al., 2007): An open source phrase-based SMT system with default settings. Words are aligned with GIZA++ (Och and Ney, 2003). The 4-gram language model with modified Kneser-Ney smoothing is trained on the target portion of training data by SRILM (Stolcke et al., 2002).

Groundhog[4]: An open source attention-based NMT system with default settings.

RNNsearch*: Our in-house implementation of NMT system with the varietal attention mechanism and other settings that presented in section 4.3.

4.3 Training

Hyper parameters: The sentence length for training NMTs is up to 50, while SMT model exploits whole training data without any restrictions. Following Bahdanau et al. (2015), we use bi-directional Gated Recurrent Unit (GRU) as the encoder. The forward representation and the backward representation are concatenated at the corresponding position as the ultimate representation of a source word. The word embedding dimension is set to 620 and the hidden layer size is 1000. The interpolation parameter λ is 0.5 and the window size l is set to 3.

Training details:

Square matrices are initialized in a random orthogonal way. Non-square matrices are initialized by sampling each element from the

[2]The corpora includes LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

[3]ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

[4]https://github.com/lisa-groundhog/GroundHog

Systems	MT03	MT04	MT05	MT06	Average	Average Increase
Moses	31.61	33.48	30.75	30.85	31.67	–
Groundhog(16K)	29.14	31.23	28.11	27.77	29.06	–
RNNsearch*(16K)	30.77	33.92	30.82	28.56	31.02	–
+ T-Distortion	35.71‡	37.81‡	33.78‡	33.79‡	35.27	+(**4.26**, **3.60**, **6.21**)
+ S-Distortion	**36.58**‡	38.47 ‡	34.85‡	33.86‡	35.94	+(**4.92**, **4.27**, **6.88**)
+ H-Distortion	35.95‡	**38.77**‡	**35.33**‡	**34.36**‡	**36.10**	+(**5.09**, **4.43**, **7.04**)
Groundhog(30K)	31.92	34.09	31.56	31.12	32.17	–
RNNsearch*(30K)	36.47	39.17	35.04	33.97	36.16	–
+ T-Distortion	37.93†	40.40‡	**36.81**‡	**35.77**‡	**37.73**	+(**1.57**, **6.06**, **5.56**)
+ S-Distortion	37.47†	**40.52**‡	36.16‡	35.32	37.37	+(**1.21**, **5.70**, **5.20**)
+ H-Distortion	**38.33**‡	40.11‡	36.71†	35.29‡	37.61	+(**1.45**, **5.94**, **5.44**)

Table 2: BLEU-4 scores (%) on NIST test set 03-06 of Moses (default settings), Groundhog (default settings), RNNsearch* and RNNsearch* with distortion models respectively. The values in brackets are increases on RNNsearch*, Moses and Groundhog respectively. ‡ indicates statistical significant difference ($p<0.01$) from RNNsearch* and † means statistical significant difference ($p<0.05$) from RNNsearch*.

Gaussian distribution with mean 0 and variance 0.01^2. All bias are initialized to 0.

Parameters are updated by Mini-batch Gradient Descent and the learning rate is controlled by the AdaDelta (Zeiler, 2012) algorithm with decay constant $\rho = 0.95$ and denominator constant $\epsilon = 1e-6$. The batch size is 80. Dropout strategy (Srivastava et al., 2014) is applied to the output layer with the dropout rate 0.5 to avoid over-fitting. The gradients of the cost function which have $L2$ norm larger than a predefined threshold 1.0 is normalized to the threshold to avoid gradients explosion (Pascanu et al., 2013). We exploit length normalization (Cho et al., 2014a) on candidate translations and the beam size for decoding is 12. For NMT with distortion models, we use trained RNNsearch* model to initialize parameters except for those related to distortions.

4.4 Results

The translation quality experiment results are shown in Table 2. We carry the experiments on different vocabulary sizes for that different vocabulary sizes cause different degrees of the rare word collocations. Through this way, we can validate the effects of our proposed models in alleviating the rare word collocations problem that leads to incorrect word alignments.

On 16K vocabularies: The phrase-based Moses performs better than the basic NMTs including Groundhog and RNNsearch*. Besides the differences between model architectures, restricted vocabularies and sentence length also affect the performance of NMTs. However, RNNsearch* with distortion models surpass phrase-based Moses by average 3.60, 4.27 and 4.43 BLEU points. RNNsearch* outperforms Groundhog by average 1.96 BLEU points due to the varietal attention mechanism, length normalization and dropout strategies. Distortion models bring about remarkable improvements as 4.26, 4.92 and 5.09 BLEU points over the RNNsearch* model.

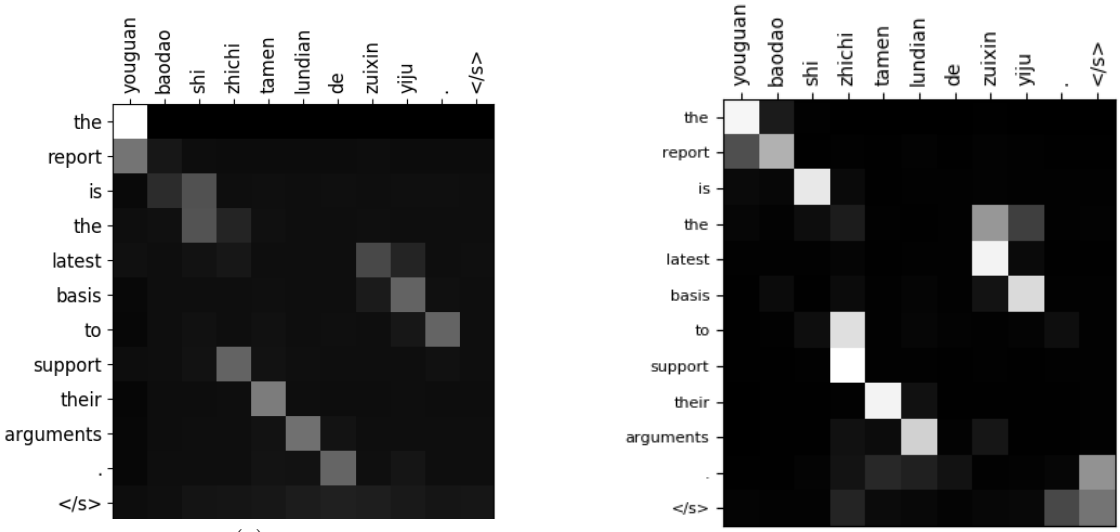
On 30K vocabularies: RNNsearch* with distortion models yield average gains by 1.57, 1.21 and 1.45 BLEU points over RNNsearch* and outperform phrase-based Moses by average 6.06, 5.70 and 5.94 BLEU points and surpass GroundHog by average 5.56, 5.20 and 5.44 BLEU points. RNNsearch*(16K) with distortion models achieve close performances with RNNsearch*(30K). The improvements on 16K vocabularies are larger than that on 30K vocabularies for the intuition that more "UNK" words lead to more rare word collocations, which results in serious attention ambiguities.

The RNNsearch* with distortion models yield tremendous improvements on BLEU scores proves the effectiveness of proposed approaches in improving translation quality.

Comparison with previous work: We present the performance comparison with pre-

System	Length	MT03	MT04	MT05	MT06	Average
Coverage	80	-	-	32.73	32.47	-
MEMDEC	50	36.16	39.81	35.91	**35.98**	36.95
NMT_{IA}	80	35.69	39.24	35.74	35.10	36.44
Our work	50	**37.93**	**40.40**	**36.81**	35.77	**37.73**

Table 3: Comparison with previous work on identical training corpora. Coverage (Tu et al., 2016) is a basic RNNsearch model with a coverage model to alleviate the over-translation and under-translation problems. MEMDEC (Wang et al., 2016) is to improve translation quality with external memory. NMT_{IA} (Meng et al., 2016) exploits a readable and writable attention mechanism to keep track of interactive history in decoding. Our work is NMT with H-Distortion model. The vocabulary sizes of all work are 30K and maximum lengths of sentence differ.



(a) (b)

Figure 4: (a) is the output of the distortion model and is calculated on shift actions of previous alignment vector. (b) is the ultimate word alignment matrix of attention-based NMT with H-Distortion model. Compared with Figure 1, (b) is more centralized and accurate.

Systems	BLEU	AER
RNNsearch*(30K)	20.90	49.73
+ T-Distortion	24.33‡	46.92
+ S-Distortion	24.10‡	47.37
+ H-Distortion	24.42‡	47.05

Table 4: BLEU-4 scores (%) and AER scores on Tsinghua manually aligned Chinese-English evaluation set. The lower the AER score, the better the alignment quality.

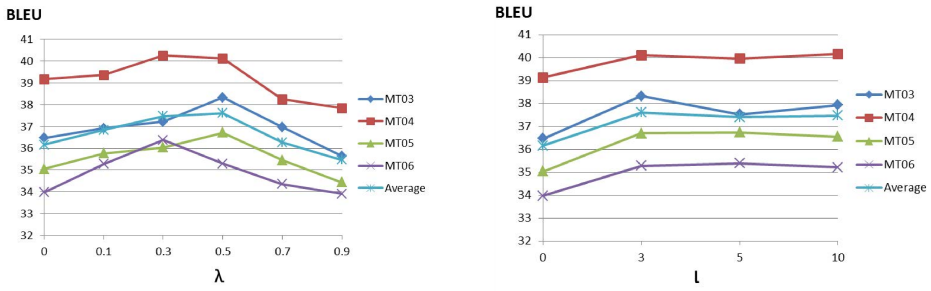
vious work that employ identical training corpora in Table 3. Our work evidently outperforms previous work on average performance. Although we restrict the maximum length of sentence to 50, our model achieves the state-of-the-art BLEU scores on almost all test sets except NIST2006.

4.5 Analysis

We investigate the effects on the alignment quality of our models and conduct the experiments to evaluate the influence of the hyper-parameter settings and the training strategies.

4.5.1 Alignment Quality

Distortion models concentrate on attending to to-be-translated words based on the word reordering knowledge and can intuitively enhance the word alignment quality. To investigate the effect on word alignment quality, we apply the BLEU and AER evaluations on Tsinghua manually aligned data set.



(a) (b)

Figure 5: Translation performance on the test sets with respect to the hyper-parameter λ and l.

System	MT03	MT04	MT05	MT06	Average
Pre-training	35.95	**38.77**	**35.33**	**34.36**	**36.10**
No pre-training	**36.99**	38.42	34.56	34.01	36.00

Table 5: Comparison between pre-training and no pre-training H-Distortion model. The performances are consistent.

Table 4 lists the BLEU and AER scores of Chinese-English translation with 30K vocabulary. RNNsearch*(30K) with distortion models achieve significant improvements on BLEU scores and obvious decrease on AER scores. The results shows that the proposed model can effectively improve the word alignment quality

Figure 4 shows the output of distortion model and ultimate alignment matrix of the above-mentioned instance. Compared with Figure 1, the alignment matrix produced by NMT with distortion models is more concentrated and accurate. The output of distortion model shows its capacity of modeling word reordering knowledge.

4.5.2 Effect of Hyper-parameters

To investigate the effect of the weight hyper-parameter λ and window hyper-parameter l in the proposed model, we carry experiments on H-Distortion model with variable hyper-parameter settings. We fix $l = 3$ for exploring the effect of λ and fix $\lambda = 0.5$ for observing the effect of l. Figure 5 presents the translation performances with respect to hyper-parameters. With the increase of weight λ, the BLEU scores first rise and then drop, which shows the distortion model provides additional helpful information while can not fully cover the attention mechanism for its insufficient content searching ability. For window l, the experiments show that larger windows bring slight further improvements, which indicates that distortion model pays more attention to the short-distance reordering knowledge.

4.5.3 Pre-training VS No Pre-training

We conduct the experiment without using pre-training strategy to observe the effect of the initialization. As is shown in Table 5, the no-pre-training model achieves consistent improvements with the pre-training one which verifies the stable effectiveness of our approach. Initialization with pre-training strategy provides a fast approach to obtain the model for it needs fewer training iterations.

5 Related Work

Our work is inspired by the distortion models that widely used in SMT. The most related work in SMT is the distortion model proposed by Yaser and Papineni (2006). Their model is identical to our S-Distortion model that captures the relative jump distance knowledge on source words. However, our approach is deliberately designed for the attention-based NMT system and is capable of exploiting variant context information to predict the relative jump distances.

Our work is related to the work (Luong et al., 2015a; Feng et al., 2016; Tu et al., 2016;

Cohn et al., 2016; Meng et al., 2016; Wang et al., 2016) that concentrate on the improvement of the attention mechanism. To remit the computing cost of the attention mechanism when dealing with long sentences, Luong et al. (2015a) proposed the local attention mechanism by just focusing on a subscope of source positions. Cohn et al. (2016) incorporated structural alignment biases into the attention mechanism and obtained improvements across several challenging language pairs in low-resource settings. Feng et al. (2016) passed the previous attention context to the attention mechanism by adding recurrent connections as the implicit distortion model. Tu et al. (2016) maintained a coverage vector for keeping the attention history to acquire accurate translations. Meng et al. (2016) proposed the interactive attention with the attentive read and attentive write operation to keep track of the interaction history. Wang et al. (2016) utilized an external memory to store additional information for guiding the attention computation. These works are different from ours, as our distortion models explicitly capture word reordering knowledge through estimating the probability distribution of relative jump distances on source words to incorporate word reordering knowledge into the attention-based NMT.

6 Conclusions

We have presented three distortion models to enhance attention-based NMT through incorporating the word reordering knowledge. The basic idea of proposed distortion models is to enable the attention mechanism to attend to the source words regarding both semantic requirement and the word reordering penalty. Experiments show that our models can evidently improve the word alignment quality and translation performance. Compared with previous work on identical corpora, our model achieves the state-of-the-art performance on average. Our model is convenient to be applied in the attention-based NMT and can be trained in the end-to-end style. We also investigated the effect of hyper-parameters and pre-training strategy and further proved the stable effectiveness of our model. In the future, we plan to validate the effectiveness of our model on more language pairs.

7 Acknowledgement

Qun Liu's work is partially supported by Science Foundation Ireland in the ADAPT Centre for Digital Content Technology (www.adaptcentre.ie) at Dublin City University funded under the SFI Research Centres Programme (Grant 13/RC/2106) co-funded under the European Regional Development Fund. We are grateful to Qiuye Zhao, Fandong Meng and Daqi Zheng for their helpful suggestions. We thank the anonymous reviewers for their insightful comments.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of ACL2006*. pages 529–536.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR2015*.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation:parameter estimation. *Computational Linguistics* 19(2):263—311.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL2005*. pages 263–270.

Kyunghyun Cho, Bart Van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Eighth Workshop on Syntax,Semantics and Structure in Statistical Translation*.

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

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of NAACL2016*. pages 876—885.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL2005*. pages 531–540.

Shi Feng, Shu jie Liu, Mu Li, and Ming Zhou. 2016. Implicit distortion and fertility models for attention-based encoder-decoder nmt model. *arXiv preprint arXiv:1601.03317* .

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule. In *Proceedings of HLT/NAACL*. Boston, volume 4, pages 273–280.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* .

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL2014*. volume 1, pages 1–10.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, and Greg Corrado. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. In *arXiv preprint arXiv:1609.08144*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP2013*. Seattle, Washington, USA, pages 1700–1709.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the ACL2007 Demo and Poster Sessions*. Prague, Czech Republic, pages 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings NAACL2003*. pages 48–54.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Proceedings of AAAI2015*. pages 2295–2301.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP2015*. Lisbon, Portugal.

Minh Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. *Proceedings of ACL2015* 27(2):82–86.

Fandong Meng, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Interactive attention for neural machine translation. In *Proceedings of COLING2016*.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alexander M Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, et al. 2004. A smorgasbord of features for statistical machine translation. In *HLT-NAACL*. pages 161–168.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics* 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL2002*. Association for Computational Linguistics, pages 311–318.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28:1310–1318.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL2016*. pages 1715–1725.

Shiqi Shen, Yong Cheng, Zhongjun He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL2016*. pages 1683–1692.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings of the international conference on spoken language processing*. volume 2, pages 901–904.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS2014*.

Christoph Tillmann. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*. pages 101–104.

Zhaopeng Tu, Zhengdong Lu, yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL*. pages 76–85.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. In *Proceedings of EMNLP2016*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL2001*. pages 523–530.

Al-Onaizan Yaser and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of ACL2006*. pages 529–536.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. In *Proceedings of EMNLP2016*.

Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search

Chris Hokamp
ADAPT Centre
Dublin City University
chris.hokamp@computing.dcu.ie

Qun Liu
ADAPT Centre
Dublin City University
qun.liu@dcu.ie

Abstract

We present Grid Beam Search (GBS), an algorithm which extends beam search to allow the inclusion of pre-specified lexical constraints. The algorithm can be used with any model that generates a sequence $\hat{\mathbf{y}} = \{y_0 \dots y_T\}$, by maximizing $p(\mathbf{y}|\mathbf{x}) = \prod_t p(y_t|\mathbf{x}; \{y_0 \dots y_{t-1}\})$. Lexical constraints take the form of phrases or words that must be present in the output sequence. This is a very general way to incorporate additional knowledge into a model's output without requiring any modification of the model parameters or training data. We demonstrate the feasibility and flexibility of Lexically Constrained Decoding by conducting experiments on Neural Interactive-Predictive Translation, as well as Domain Adaptation for Neural Machine Translation. Experiments show that GBS can provide large improvements in translation quality in interactive scenarios, and that, even without any user input, GBS can be used to achieve significant gains in performance in domain adaptation scenarios.

1 Introduction

The output of many natural language processing models is a sequence of text. Examples include automatic summarization (Rush et al., 2015), machine translation (Koehn, 2010; Bahdanau et al., 2014), caption generation (Xu et al., 2015), and dialog generation (Serban et al., 2016), among others.

In some real-world scenarios, additional information that could inform the search for the optimal output sequence may be available at inference time. Humans can provide corrections after viewing a system's initial output, or separate classification models may be able to predict parts of the output with high confidence. When the domain of the input is known, a domain terminology may be employed to ensure specific phrases are present in a system's predictions. Our goal in this work is to find a way to force the output of a model to contain such *lexical constraints*, while still taking advantage of the distribution learned from training data.

For Machine Translation (MT) usecases in particular, final translations are often produced by combining automatically translated output with user inputs. Examples include Post-Editing (PE) (Koehn, 2009; Specia, 2011) and Interactive-Predictive MT (Foster, 2002; Barrachina et al., 2009; Green, 2014). These interactive scenarios can be unified by considering user inputs to be lexical constraints which guide the search for the optimal output sequence.

In this paper, we formalize the notion of lexical constraints, and propose a decoding algorithm which allows the specification of subsequences that are required to be present in a model's output. Individual constraints may be single tokens or multi-word phrases, and any number of constraints may be specified simultaneously.

Although we focus upon interactive applications for MT in our experiments, lexically constrained decoding is relevant to any scenario where a model is asked to generate a sequence $\hat{\mathbf{y}} = \{y_0 \dots y_T\}$ given both an input $\mathbf{x}$, and a set $\{\mathbf{c_0} \dots \mathbf{c_n}\}$, where each $\mathbf{c_i}$ is a sub-sequence $\{c_{i0} \dots c_{ij}\}$, that must appear somewhere in $\hat{\mathbf{y}}$. This makes our work applicable to a wide range of text generation scenarios, including image description, dialog generation, abstractive summarization, and question answering.

The rest of this paper is organized as follows: Section 2 gives the necessary background for our

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1535–1546
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1141

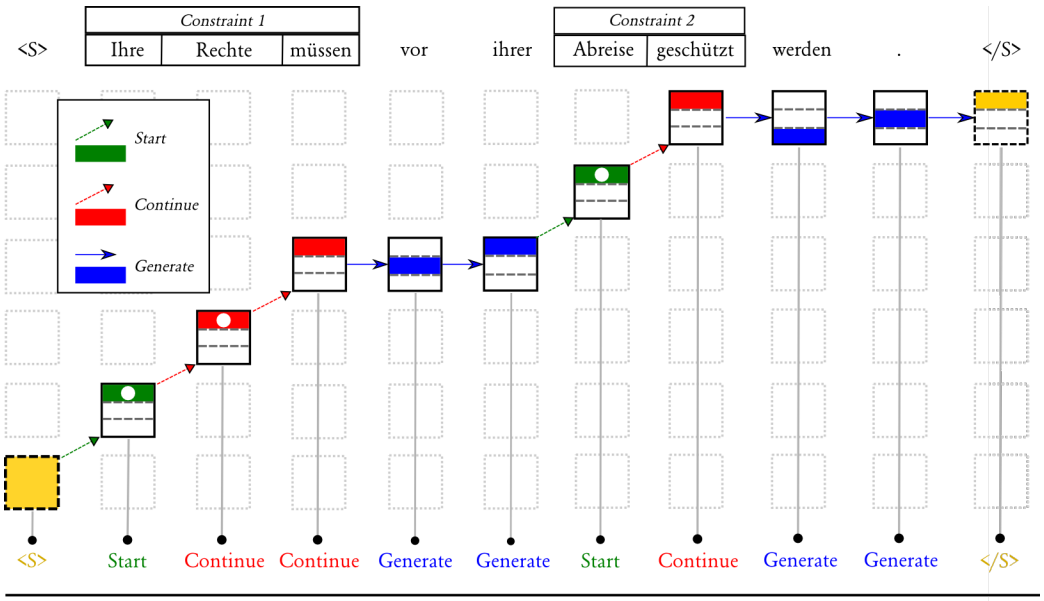


Figure 1: A visualization of the decoding process for an actual example from our English-German MT experiments. The output token at each timestep appears at the top of the figure, with lexical constraints enclosed in boxes. *Generation* is shown in blue, *Starting* new constraints in green, and *Continuing* constraints in red. The function used to create the hypothesis at each timestep is written at the bottom. Each box in the grid represents a beam; a colored strip inside a beam represents an individual hypothesis in the beam's k-best stack. Hypotheses with circles inside them are *closed*, all other hypotheses are *open*. (Best viewed in colour).

discussion of GBS, Section 3 discusses the lexically constrained decoding algorithm in detail, Section 4 presents our experiments, and Section 5 gives an overview of closely related work.

2 Background: Beam Search for Sequence Generation

Under a model parameterized by θ, let the best output sequence $\hat{\mathbf{y}}$ given input $\mathbf{x}$ be Eq. 1.

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \{\mathbf{y}^{[\mathbf{T}]}\}}{\operatorname{argmax}} p_\theta(\mathbf{y}|\mathbf{x}), \tag{1}$$

where we use $\{\mathbf{y}^{[\mathbf{T}]}\}$ to denote the set of all sequences of length T. Because the number of possible sequences for such a model is $|\mathbf{v}|^T$, where $|\mathbf{v}|$ is the number of output symbols, the search for $\hat{\mathbf{y}}$ can be made more tractable by factorizing $p_\theta(\mathbf{y}|\mathbf{x})$ into Eq. 2:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \prod_{t=0}^{T} p_\theta(y_t|\mathbf{x}; \{y_0 \dots y_{t-1}\}). \tag{2}$$

The standard approach is thus to generate the output sequence from beginning to end, conditioning the output at each timestep upon the input $\mathbf{x}$, and the already-generated symbols $\{y_0 \dots y_{i-t}\}$. However, greedy selection of the most probable output at each timestep, i.e.:

$$\hat{y}_t = \underset{y_i \in \{\mathbf{v}\}}{\operatorname{argmax}} p(y_i|\mathbf{x}; \{y_0 \dots y_{t-1}\}), \tag{3}$$

risks making locally optimal decisions which are actually globally sub-optimal. On the other hand, an exhaustive exploration of the output space would require scoring $|\mathbf{v}|^T$ sequences, which is intractable for most real-world models. Thus, a search or *decoding* algorithm is often used as a compromise between these two extremes. A common solution is to use a heuristic search to attempt to find the best output efficiently (Pearl, 1984; Koehn, 2010; Rush et al., 2013). The key idea is to discard bad options early, while trying to avoid discarding candidates that may be locally risky, but could eventually result in the best overall output.

Beam search (Och and Ney, 2004) is probably the most popular search algorithm for decoding sequences. Beam search is simple to implement, and is flexible in the sense that the semantics of the

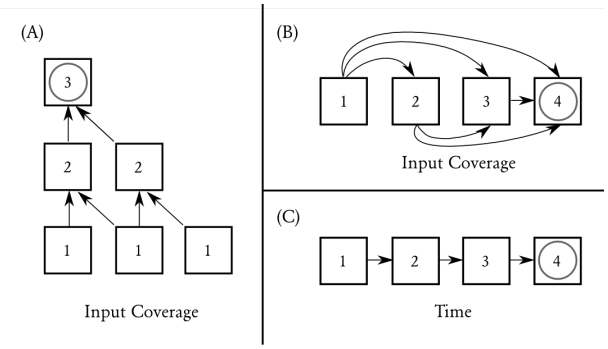


Figure 2: Different structures for beam search. Boxes represent beams which hold k-best lists of hypotheses. (A) Chart Parsing using SCFG rules to cover spans in the input. (B) Source coverage as used in PB-SMT. (C) Sequence timesteps (as used in Neural Sequence Models), GBS is an extension of (C). In (A) and (B), hypotheses are finished once they reach the final beam. In (C), a hypothesis is only complete if it has generated an end-of-sequence (EOS) symbol.

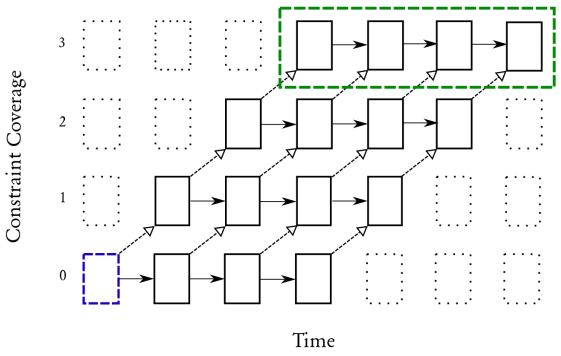


Figure 3: Visualizing the lexically constrained decoder's complete search graph. Each rectangle represents a beam containing k hypotheses. Dashed (diagonal) edges indicate *starting* or *continuing* constraints. Horizontal edges represent *generating* from the model's distribution. The horizontal axis covers the timesteps in the output sequence, and the vertical axis covers the constraint tokens (one row for each token in each constraint). Beams on the top level of the grid contain hypotheses which cover all constraints.

graph of beams can be adapted to take advantage of additional structure that may be available for specific tasks. For example, in Phrase-Based Statistical MT (PB-SMT) (Koehn, 2010), beams are organized by the number of source words that are covered by the hypotheses in the beam – a hypothesis is "finished" when it has covered all source words. In chart-based decoding algorithms such as CYK, beams are also tied to coverage of the input, but are organized as cells in a chart, which facilitates search for the optimal latent structure of the output (Chiang, 2007). Figure 2 visualizes three common ways to structure search. (A) and (B) depend upon explicit structural information between the input and output, (C) only assumes that the output is a sequence where later symbols depend upon earlier ones. Note also that (C) corresponds exactly to the bottom rows of Figures 1 and 3.

With the recent success of neural models for text generation, beam search has become the de-facto choice for decoding optimal output sequences (Sutskever et al., 2014). However, with neural sequence models, we cannot organize beams by their explicit coverage of the input. A simpler alternative is to organize beams by output timesteps from $t_0 \cdots t_N$, where N is a hyperparameter that can be set heuristically, for example by multiplying a factor with the length of the input to make an educated guess about the maximum length of the output (Sutskever et al., 2014). Output sequences are generally considered complete once a special "end-of-sentence"(EOS) token has been generated. Beam size in these models is also typically kept small, and recent work has shown that the performance of some architectures can actually degrade with larger beam size (Tu et al., 2016).

3 Grid Beam Search

Our goal is to organize decoding in such a way that we can constrain the search space to outputs which contain one or more pre-specified sub-sequences. We thus wish to use a model's distribution both to "place" lexical constraints correctly, and to generate the parts of the output which are not covered by the constraints.

Algorithm 1 presents the pseudo-code for lexically constrained decoding, see Figures 1 and 3 for visualizations of the search process. Beams in the grid are indexed by t and c. The t variable tracks the timestep of the search, while the c variable indicates how many constraint tokens are covered by the hypotheses in the current beam. Note that each step of c covers a single constraint $token$. In other words, $constraints$ is an array of sequences, where individual tokens can be indexed as $constraints_{ij}$, i.e. $token_j$ in $constraint_i$. The $numC$ parameter in Algorithm 1 represents the total number of tokens in all constraints.

The hypotheses in a beam can be separated into two types (see lines 9-11 and 15-19 of Algorithm 1):

1. ***open*** hypotheses can either generate from the model's distribution, or start available constraints,

2. ***closed*** hypotheses can only generate the next

Algorithm 1 Pseudo-code for Grid Beam Search, note that t and c indices are 0-based

procedure CONSTRAINEDSEARCH($model$, $input$, $constraints$, $maxLen$, $numC$, k)
  $\mathbf{startHyp} \Leftarrow$ model.getStartHyp($input, constraints$)
  $\mathbf{Grid} \Leftarrow$ initGrid($maxLen$, $numC$, k) ▷ initialize beams in grid
  $\mathbf{Grid}[0][0] = \mathbf{startHyp}$
  for $t = 1,\ \ t{+}{+},\ \ t < maxLen$ **do**
    for $c = max(0, (numC + t) - maxLen),\ \ c{+}{+},\ \ c \leq min(t, numC)$ **do**
      $n, s, g = \varnothing$
      for each $hyp \in \mathbf{Grid}[t-1][c]$ **do**
        if $hyp.isOpen()$ **then**
          $g \Leftarrow g \bigcup$ model.generate($hyp, input, constraints$) ▷ generate new open hyps
        end if
      end for
      if $c > 0$ **then**
        for each $hyp \in \mathbf{Grid}[t-1][c-1]$ **do**
          if $hyp.isOpen()$ **then**
            $n \Leftarrow n \bigcup$ model.start($hyp, input, constraints$) ▷ start new constrained hyps
          else
            $s \Leftarrow s \bigcup$ model.continue($hyp, input, constraints$) ▷ continue unfinished
          end if
        end for
      end if
      $\mathbf{Grid}[t][c] = \underset{h \in n \bigcup s \bigcup g}{\text{k-argmax}}$ model.score(h) ▷ k-best scoring hypotheses stay on the beam
    end for
  end for
  $topLevelHyps \Leftarrow \mathbf{Grid}[:][numC]$ ▷ get hyps in top-level beams
  $finishedHyps \Leftarrow hasEOS(topLevelHyps)$ ▷ finished hyps have generated the EOS token
  $bestHyp \Leftarrow \underset{h \in finishedHyps}{\text{argmax}}$ model.score(h)
  return $bestHyp$
end procedure

token for in a currently unfinished constraint.

At each step of the search the beam at $\mathbf{Grid}[t][c]$ is filled with candidates which may be created in three ways:

1. the **open** hypotheses in the beam to the left ($\mathbf{Grid}[t-1][c]$) may ***generate*** continuations from the model's distribution $p_\theta(y_i|\mathbf{x}, \{y_0 \dots y_{i-1}\})$,

2. the **open** hypotheses in the beam to the left and below ($\mathbf{Grid}[t-1][c-1]$) may ***start*** new constraints,

3. the **closed** hypotheses in the beam to the left and below ($\mathbf{Grid}[t-1][c-1]$) may ***continue*** constraints.

Therefore, the **model** in Algorithm 1 implements an interface with three functions: $generate$, $start$, and $continue$, which build new hypotheses in each of the three ways. Note that the scoring function of the model does not need to be aware of the existence of constraints, but it may be, for example via a feature which indicates if a hypothesis is part of a constraint or not.

The beams at the top level of the grid (beams where $c = numConstraints$) contain hypotheses which cover all of the constraints. Once a hypothesis on the top level generates the EOS token, it can be added to the set of finished hypotheses. The highest scoring hypothesis in the set of finished hypotheses is the best sequence which covers all constraints.[1]

[1]Our implementation of GBS is available at https://github.com/chrishokamp/constrained_decoding

3.1 Multi-token Constraints

By distinguishing between **open** and **closed** hypotheses, we can allow for arbitrary multi-token phrases in the search. Thus, the set of constraints for a particular output may include both individual tokens and phrases. Each hypothesis maintains a coverage vector to ensure that constraints cannot be repeated in a search path – hypotheses which have already covered $constraint_i$ can only $generate$, or $start$ constraints that have not yet been covered.

Note also that discontinuous lexical constraints, such as phrasal verbs in English or German, are easy to incorporate into GBS, by adding *filters* to the search, which require that one or more conditions must be met before a constraint can be used. For example, adding the phrasal verb "ask ⟨someone⟩ out" as a constraint would mean using "ask" as $constraint_0$ and "out" as $constraint_1$, with two filters: one requiring that $constraint_1$ cannot be used before $constraint_0$, and another requiring that there must be at least one $generated$ token between the constraints.

3.2 Subword Units

Both the computation of the score for a hypothesis, and the granularity of the tokens (character, subword, word, etc...) are left to the underlying model. Because our decoder can handle arbitrary constraints, there is a risk that constraints will contain tokens that were never observed in the training data, and thus are unknown by the model. Especially in domain adaptation scenarios, some user-specified constraints are very likely to contain unseen tokens. Subword representations provide an elegant way to circumvent this problem, by breaking unknown or rare tokens into character n-grams which are part of the model's vocabulary (Sennrich et al., 2016; Wu et al., 2016). In the experiments in Section 4, we use this technique to ensure that no input tokens are unknown, even if a constraint contains words which never appeared in the training data.[2]

3.3 Efficiency

Because the number of beams is multiplied by the number of constraints, the runtime complexity of a naive implementation of GBS is $\mathcal{O}(ktc)$. Standard time-based beam search is $\mathcal{O}(kt)$; therefore, some consideration must be given to the efficiency of this algorithm. Note that the beams in each column c of Figure 3 are independent, meaning that GBS can be parallelized to allow all beams at each timestep to be filled simultaneously. Also, we find that the most time is spent computing the states for the hypothesis candidates, so by keeping the beam size small, we can make GBS significantly faster.

3.4 Models

The models used for our experiments are state-of-the-art Neural Machine Translation (NMT) systems using our own implementation of NMT with attention over the source sequence (Bahdanau et al., 2014). We used Blocks and Fuel to implement our NMT models (van Merrinboer et al., 2015). To conduct the experiments in the following section, we trained baseline translation models for English–German (EN-DE), English–French (EN-FR), and English–Portuguese (EN-PT). We created a shared subword representation for each language pair by extracting a vocabulary of 80000 symbols from the concatenated source and target data. See the Appendix for more details on our training data and hyperparameter configuration for each language pair. The $beamSize$ parameter is set to 10 for all experiments.

Because our experiments use NMT models, we can now be more explicit about the implementations of the $generate$, $start$, and $continue$ functions for this GBS instantiation. For an NMT model at timestep t, $generate(hyp_{t-1})$ first computes a vector of output probabilities $\mathbf{o}_t = softmax(g(y_{t-1}, s_i, c_i))$[3] using the state information available from hyp_{t-1}. and returns the best k continuations, i.e. Eq. 4:

$$\mathbf{g}_t = \text{k-argmax}_i \, \mathbf{o}_{ti}. \tag{4}$$

The $start$ and $continue$ functions simply index into the softmax output of the model, selecting specific tokens instead of doing a k-argmax over the entire target language vocabulary. For example, to *start* constraint $\mathbf{c}_i$, we find the score of token $\mathbf{c}_{i0}$, i.e. $\mathbf{o}_{tc_{i0}}$.

4 Experiments

4.1 Pick-Revise for Interactive Post Editing

Pick-Revise is an interaction cycle for MT Post-Editing proposed by Cheng et al. (2016). Starting

[2] If a *character* that was not observed in training data is observed at prediction time, it will be unknown. However, we did not observe this in any of our experiments.

[3] we use the notation for the g function from Bahdanau et al. (2014)

ITERATION	0	1	2	3
Strict Constraints				
EN-DE	18.44	27.64 (+9.20)	36.66 (+9.01)	**43.92** (+7.26)
EN-FR	28.07	36.71 (+8.64)	44.84 (+8.13)	**45.48** +(0.63)
EN-PT*	15.41	23.54 (+8.25)	31.14 (+7.60)	**35.89** (+4.75)
Relaxed Constraints				
EN-DE	18.44	26.43 (+7.98)	34.48 (+8.04)	**41.82** (+7.34)
EN-FR	28.07	33.8 (+5.72)	40.33 (+6.53)	**47.0** (+6.67)
EN-PT*	15.41	23.22 (+7.80)	33.82 (+10.6)	**40.75** (+6.93)

Table 1: Results for four simulated editing cycles using WMT test data. EN-DE uses *newstest2013*, EN-FR uses *newstest2014*, and EN-PT uses the Autodesk corpus discussed in Section 4.2. Improvement in BLEU score over the previous cycle is shown in parentheses. * indicates use of our test corpus created from Autodesk post-editing data.

with the original translation hypothesis, a (simulated) user first picks a part of the hypothesis which is incorrect, and then provides the correct translation for that portion of the output. The user-provided correction is then used as a constraint for the next decoding cycle. The Pick-Revise process can be repeated as many times as necessary, with a new constraint being added at each cycle.

We modify the experiments of Cheng et al. (2016) slightly, and assume that the user only provides sequences of up to three words which are missing from the hypothesis.[4] To simulate user interaction, at each iteration we chose a phrase of up to three tokens from the reference translation which does not appear in the current MT hypotheses. In the *strict* setting, the complete phrase must be missing from the hypothesis. In the *relaxed* setting, only the first word must be missing. Table 1 shows results for a simulated editing session with four cycles. When a three-token phrase cannot be found, we backoff to two-token phrases, then to single tokens as constraints. If a hypothesis already matches the reference, no constraints are added. By specifying a new constraint of up to three words at each cycle, an increase of over 20 BLEU points is achieved in all language pairs.

4.2 Domain Adaptation via Terminology

The requirement for use of domain-specific terminologies is common in real-world applications of MT (Crego et al., 2016). Existing approaches incorporate placeholder tokens into NMT systems, which requires modifying the pre- and post- processing of the data, and training the system with data that contains the same placeholders which occur in the test data (Crego et al., 2016). The MT system also loses any possibility to model the tokens in the terminology, since they are represented by abstract tokens such as "⟨TERM_1⟩". An attractive alternative is to simply provide term mappings as constraints, allowing any existing system to adapt to the terminology used in a new test domain.

For the target domain data, we use the Autodesk Post-Editing corpus (Zhechev, 2012), which is a dataset collected from actual MT post-editing sessions. The corpus is focused upon software localization, a domain which is likely to be very different from the WMT data used to train our general domain models. We divide the corpus into approximately 100,000 training sentences, and 1000 test segments, and automatically generate a terminology by computing the Pointwise Mutual Information (PMI) (Church and Hanks, 1990) between source and target n-grams in the training set. We extract all n-grams from length 2-5 as terminology candidates.

$$\mathbf{pmi}(\mathbf{x};\mathbf{y}) = \mathbf{log}\frac{p(x,y)}{p(x)p(y)} \quad (5)$$

$$\mathbf{npmi}(\mathbf{x};\mathbf{y}) = \frac{\mathbf{pmi}(\mathbf{x};\mathbf{y})}{\boldsymbol{h}(\mathbf{x},\mathbf{y})} \quad (6)$$

Equations 5 and 6 show how we compute the normalized PMI for a terminology candidate pair. The PMI score is normalized to the range $[-1,+1]$ by dividing by the entropy $\boldsymbol{h}$ of the joint probability $\mathbf{p}(\mathbf{x},\mathbf{y})$. We then filter the candidates to only include pairs whose PMI is ≥ 0.9, and where both the source and target phrases occur at least five times in the corpus. When source phrases that match the terminology are observed in the test

[4]NMT models do not use explicit alignment between source and target, so we cannot use alignment information to map target phrases to source phrases

data, the corresponding target phrase is added to the constraints for that segment. Results are shown in Table 2.

As a sanity check that improvements in BLEU are not merely due to the presence of the terms somewhere in the output, i.e. that the *placement* of the terms by GBS is reasonable, we also evaluate the results of randomly inserting terms into the baseline output, and of prepending terms to the baseline output.

This simple method of domain adaptation leads to a significant improvement in the BLEU score without any human intervention. Surprisingly, even an automatically created terminology combined with GBS yields performance improvements of approximately +2 BLEU points for En-De and En-Fr, and a gain of almost 14 points for En-Pt. The large improvement for En-Pt is probably due to the training data for this system being very different from the IT domain (see Appendix). Given the performance improvements from our automatically extracted terminology, manually created domain terminologies with good coverage of the test domain are likely to lead to even greater gains. Using a terminology with GBS is likely to be beneficial in any setting where the test domain is significantly different from the domain of the model's original training data.

System	BLEU
EN-DE	
Baseline	26.17
Random	25.18 (-0.99)
Beginning	26.44 (+0.26)
GBS	**27.99** (+1.82)
EN-FR	
Baseline	32.45
Random	31.48 (-0.97)
Beginning	34.51 (+2.05)
GBS	**35.05** (+2.59)
EN-PT	
Baseline	15.41
Random	18.26 (+2.85)
Beginning	20.43 (+5.02)
GBS	**29.15** (+13.73)

Table 2: BLEU Results for EN-DE, EN-FR, and EN-PT terminology experiments using the Autodesk Post-Editing Corpus. "Random' indicates inserting terminology constraints at random positions in the baseline translation. "Beginning" indicates prepending constraints to baseline translations.

4.3 Analysis

Subjective analysis of decoder output shows that phrases added as constraints are not only placed correctly within the output sequence, but also have global effects upon translation quality. This is a desirable effect for user interaction, since it implies that users can bootstrap quality by adding the most critical constraints (i.e. those that are most essential to the output), first. Table 3 shows several examples from the experiments in Table 1, where the addition of lexical constraints was able to guide our NMT systems away from initially quite low-scoring hypotheses to outputs which perfectly match the reference translations.

5 Related Work

Most related work to date has presented modifications of SMT systems for specific usecases which constrain MT output via auxilliary inputs. The largest body of work considers **Interactive Machine Translation** (IMT): an MT system searches for the optimal target-language suffix given a complete source sentence and a desired prefix for the target output (Foster, 2002; Barrachina et al., 2009; Green, 2014). IMT can be viewed as subcase of constrained decoding, where there is only one constraint which is guaranteed to be placed at the beginning of the output sequence. Wuebker et al. (2016) introduce **prefix-decoding**, which modifies the SMT beam search to first ensure that the *target prefix* is covered, and only then continues to build hypotheses for the suffix using beams organized by coverage of the remaining phrases in the source segment. Wuebker et al. (2016) and Knowles and Koehn (2016) also present a simple modification of NMT models for IMT, enabling models to predict suffixes for user-supplied prefixes.

Recently, some attention has also been given to SMT decoding with multiple lexical constraints. The Pick-Revise (PRIMT) (Cheng et al., 2016) framework for Interactive Post Editing introduces the concept of *edit cycles*. Translators specify constraints by editing a part of the MT output that is incorrect, and then asking the system for a new hypothesis, which must contain the user-provided correction. This process is repeated, maintaining constraints from previous iterations and adding new ones as needed. Importantly, their approach relies upon the phrase segmentation provided by the SMT system. The decoding algorithm can

EN-DE	
Source	
He was also an anti- smoking activist and took part in several campaigns .	
Original Hypothesis	
Es war auch ein Anti- Rauch- Aktiv- ist und nahmen an mehreren Kampagnen teil .	
Reference	**Constraints**
Ebenso setzte er sich gegen das Rauchen ein und nahm an mehreren Kampagnen teil .	(1) Ebenso setzte er
Constrained Hypothesis	(2) gegen das Rauchen
Ebenso setzte er sich **gegen das Rauchen** ein und **nahm** an mehreren Kampagnen teil .	(3) nahm
EN-FR	
Source	
At that point I was no longer afraid of him and I was able to love him .	
Original Hypothesis	
Je n'avais plus peur de lui et j'ètais capable de l'aimer .	
Reference	**Constraints**
Lá je n'ai plus eu peur de lui et j'ai pu l'aimer .	(1) Lá je n'ai
Constrained Hypothesis	(2) j'ai pu
Lá je n'ai plus **eu** peur de lui et **j'ai pu** l'aimer .	(3) eu
EN-PT	
Source	
Mo- dif- y drain- age features by selecting them individually .	
Original Hypothesis	
- Já temos as características de extracção de idade , com eles individualmente .	
Reference	**Constraints**
Modi- fique os recursos de drenagem ao selec- ion- á-los individualmente .	(1) drenagem ao selec-
Constrained Hypothesis	(2) Modi- fique os
Modi- fique os recursos de **drenagem ao selec-** ion- á-los individualmente .	(3) recursos

Table 3: Manual analysis of examples from lexically constrained decoding experiments. "-" followed by whitespace indicates the internal segmentation of the translation model (see Section 3.2)

only make use of constraints that match phrase boundaries, because constraints are implemented as "rules" enforcing that source phrases must be translated as the aligned target phrases that have been selected as constraints. In contrast, our approach decodes at the token level, and is not dependent upon any explicit structure in the underlying model.

Domingo et al. (2016) also consider an interactive scenario where users first choose portions of an MT hypothesis to keep, then query for an updated translation which preserves these portions. The MT system decodes the source phrases which are not aligned to the user-selected phrases until the source sentence is fully covered. This approach is similar to the system of Cheng et al., and uses the "XML input" feature in Moses (Koehn et al., 2007).

Some recent work considers the inclusion of soft lexical constraints directly into deep models for dialog generation, and special cases, such as recipe generation from a list of ingredients (Wen et al., 2015; Kiddon et al., 2016). Such constraint-aware models are complementary to our work, and could be used with GBS decoding without any change to the underlying models.

To the best of our knowledge, ours is the first work which considers general lexically constrained decoding for any model which outputs sequences, without relying upon alignments between input and output, and without using a search organized by coverage of the input.

6 Conclusion

Lexically constrained decoding is a flexible way to incorporate arbitrary subsequences into the output of any model that generates output sequences token-by-token. A wide spectrum of popular text generation models have this characteristic, and GBS should be straightforward to use with any model that already uses beam search.

In translation interfaces where translators can provide corrections to an existing hypothesis, these user inputs can be used as constraints, generating a new output each time a user fixes an error. By simulating this scenario, we have shown that such a workflow can provide a large improvement in translation quality at each iteration.

By using a domain-specific terminology to generate target-side constraints, we have shown that a general domain model can be adapted to a new domain without any retraining. Surprisingly, this simple method can lead to significant performance gains, even when the terminology is created automatically.

In future work, we hope to evaluate GBS with models outside of MT, such as automatic summarization, image captioning or dialog generation. We also hope to introduce new constraint-aware models, for example via secondary attention mechanisms over lexical constraints.

Acknowledgments

This project has received funding from Science Foundation Ireland in the ADAPT Centre for Digital Content Technology (www.adaptcentre.ie) at Dublin City University funded under the SFI Research Centres Programme (Grant 13/RC/2106) co-funded under the European Regional Development Fund and the European Union Horizon 2020 research and innovation programme under grant agreement 645452 (QT21). We thank the anonymous reviewers, as well as Iacer Calixto, Peyman Passban, and Henry Elder for helpful feedback on early versions of this work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics* 35(1):3–28. https://doi.org/10.1162/coli.2008.07-055-R2-06-29.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Lisbon, Portugal, pages 1–46. http://aclweb.org/anthology/W15-3001.

Shanbo Cheng, Shujian Huang, Huadong Chen, Xinyu Dai, and Jiajun Chen. 2016. PRIMT: A pick-revise framework for interactive machine translation. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 1240–1249. http://aclweb.org/anthology/N/N16/N16-1148.pdf.

David Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.* 33(2):201–228. https://doi.org/10.1162/coli.2007.33.2.201.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Linguist.* 16(1):22–29. http://dl.acm.org/citation.cfm?id=89086.89095.

Josep Maria Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, Satoshi Enoue, Chiyo Geiss, Joshua Johanson, Ardas Khalsa, Raoum Khiari, Byeongil Ko, Catherine Kobus, Jean Lorieux, Leidiana Martins, Dang-Chuan Nguyen, Alexandra Priori, Thomas Riccardi, Natalia Segal, Christophe Servan, Cyril Tiquet, Bo Wang, Jin Yang, Dakun Zhang, Jing Zhou, and Peter Zoldan. 2016. Systran's pure neural machine translation systems. *CoRR* abs/1610.05540. http://arxiv.org/abs/1610.05540.

Miguel Domingo, Alvaro Peris, and Francisco Casacuberta. 2016. Interactive-predictive translation based on multiple word-segments. *Baltic J. Modern Computing* 4(2):282–291.

George F. Foster. 2002. *Text Prediction for Translators*. Ph.D. thesis, Montreal, P.Q., Canada, Canada. AAINQ72434.

Spence Green. 2014. *Mixed-Initiative Natural Language Translation*. Ph.D. thesis, Stanford, CA, United States.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 329–339. http://aclweb.org/anthology/D/D16/D16-1032.pdf.

Rebecca Knowles and Philipp Koehn. 2016. Neural interactive translation prediction. *AMTA 2016, Vol.* page 107.

Philipp Koehn. 2009. A process study of computer-aided translation. *Machine Translation* 23(4):241–263. https://doi.org/10.1007/s10590-010-9076-3.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics,

Stroudsburg, PA, USA, ACL '07, pages 177–180. http://dl.acm.org/citation.cfm?id=1557769.1557821.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Comput. Linguist.* 30(4):417–449. https://doi.org/10.1162/0891201042544884.

Judea Pearl. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Alexander Rush, Yin-Wen Chang, and Michael Collins. 2013. Optimal beam search for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 210–221. http://www.aclweb.org/anthology/D13-1022.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*. The Association for Computational Linguistics, pages 379–389.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. http://aclweb.org/anthology/P/P16/P16-1162.pdf.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'16, pages 3776–3783. http://dl.acm.org/citation.cfm?id=3016387.3016435.

Jason R. Smith, Herve Saint-amand, Chris Callison-burch, Magdalena Plamada, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the common crawl. In *In Proceedings of the Conference of the Association for Computational Linguistics (ACL*.

Lucia Specia. 2011. Exploiting objective annotations for measuring translation post-editing effort. In *Proceedings of the European Association for Machine Translation*. May.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Toma Erjavec, and Dan Tufi. 2006. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. In *In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006*. pages 2142–2147.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, NIPS'14, pages 3104–3112. http://dl.acm.org/citation.cfm?id=2969033.2969173.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2016. Neural machine translation with reconstruction. *arXiv preprint arXiv:1611.01874* .

Bart van Merrinboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: Frameworks for deep learning. *CoRR* abs/1506.00619.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Joern Wuebker, Spence Green, John DeNero, Sasa Hasan, and Minh-Thang Luong. 2016. Models and inference for prefix-constrained machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 66–75. http://www.aclweb.org/anthology/P16-1007.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. JMLR Workshop and Conference Proceedings, pages 2048–2057. http://jmlr.org/proceedings/papers/v37/xuc15.pdf.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Ventsislav Zhechev. 2012. Machine Translation Infrastructure and Post-editing Performance at Autodesk.

In *AMTA 2012 Workshop on Post-Editing Technology and Practice (WPTP 2012)*. Association for Machine Translation in the Americas (AMTA), San Diego, USA, pages 87–96.

A NMT System Configurations

We train all systems for 500000 iterations, with validation every 5000 steps. The best single model from validation is used in all of the experiments for a language pair. We use ℓ_2 regularization on all parameters with $\alpha = 1e^{-5}$. Dropout is used on the output layers with $p(drop) = 0.5$. We sort mini-batches by source sentence length, and reshuffle training data after each epoch.

All systems use a bidirectional GRUs (Cho et al., 2014) to create the source representation and GRUs for the decoder transition. We use AdaDelta (Zeiler, 2012) to update gradients, and clip large gradients to 1.0.

Training Configurations	
EN-DE	
Embedding Size	300
Recurrent Layers Size	1000
Source Vocab Size	80000
Target Vocab Size	90000
Batch Size	50
EN-FR	
Embedding Size	300
Recurrent Layers Size	1000
Source Vocab Size	66000
Target Vocab Size	74000
Batch Size	40
EN-PT	
Embedding Size	200
Recurrent Layers Size	800
Source Vocab Size	60000
Target Vocab Size	74000
Batch Size	40

A.1 English-German

Our English-German training corpus consists of 4.4 Million segments from the Europarl (Bojar et al., 2015) and CommonCrawl (Smith et al., 2013) corpora.

A.2 English-French

Our English-French training corpus consists of 4.9 Million segments from the Europarl and CommonCrawl corpora.

A.3 English-Portuguese

Our English-Portuguese training corpus consists of 28.5 Million segments from the Europarl, JRC-Aquis (Steinberger et al., 2006) and OpenSubtitles[5] corpora.

[5] http://www.opensubtitles.org/

Combating Human Trafficking with Deep Multimodal Models

Edmund Tong*
Language Technologies Institute
Carnegie Mellon University
edtong@cmu.edu

Amir Zadeh*
Language Technologies Institute
Carnegie Mellon University
abagherz@cs.cmu.edu

Cara Jones
Marinus Analytics, LLC
cara@marinusanalytics.com

Louis-Philippe Morency
Language Technologies Institute
Carnegie Mellon University
morency@cs.cmu.edu

Abstract

Human trafficking is a global epidemic affecting millions of people across the planet. Sex trafficking, the dominant form of human trafficking, has seen a significant rise mostly due to the abundance of escort websites, where human traffickers can openly advertise among at-will escort advertisements. In this paper, we take a major step in the automatic detection of advertisements suspected to pertain to human trafficking. We present a novel dataset called Trafficking-10k, with more than 10,000 advertisements annotated for this task. The dataset contains two sources of information per advertisement: text and images. For the accurate detection of trafficking advertisements, we designed and trained a deep multimodal model called the Human Trafficking Deep Network (HTDN).

1 Introduction

Human trafficking "a crime that shames us all" (UNODC, 2008), has seen a steep rise in the United States since 2012. The number of cases reported rose from 3,279 in 2012 to 7,572 in 2016—more than doubling over the course of five years (Hotline). Sex trafficking is a form of human trafficking, and is a global epidemic affecting millions of people each year (McCarthy, 2014). Victims of sex trafficking are subjected to coercion, force, and control, and are not able to ask for help. Put plainly, sex trafficking is modern-day slavery and is one of the top priorities of law enforcement agencies at all levels.

A major advertising ground for human traffickers is the World Wide Web. The Internet has brought traffickers the ability to advertise online and has fostered the growth of numerous adult escort sites. Each day, there are tens of thousands of Internet advertisements posted in the United States and Canada that market commercial sex. Hiding among the noise of at-will adult escort ads are ads posted by sex traffickers. Often long undetected, trafficking rings and escort websites form a profit cycle that fuels the increase of both trafficking rings and escort websites.

For law enforcement, this presents a significant challenge: how should we identify advertisements that are associated with sex trafficking? Police have limited human and technical resources, and manually sifting through thousands of ads in the hopes of finding something suspicious is a poor use of those resources, even if they know what they are looking for. Leveraging state-of-the-art machine learning approaches in Natural Language Processing and computer vision to detect and report advertisements suspected of trafficking is the main focus of our work. In other words, we strive to find the victims and perpetrators of trafficking who hide in plain sight in the massive amounts of data online. By narrowing down the number of advertisements that law enforcement must sift through, we endeavor to provide a real opportunity for law enforcement to intervene in the lives of victims. However, there are non-trivial challenges facing this line of research:

Adversarial Environment. Human trafficking rings are aware that law enforcement monitors their online activity. Over the years, law enforcement officers have populated lists of keywords that frequently occur in trafficking advertisements. However, these simplistic queries fail when traffickers use complex obfuscation. Traffickers, again aware of this, move to new keywords to blend in with the at-will escort advertisements. This trend creates an adversarial environment for any machine learning

* Authors contributed equally.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1547–1556
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1142

system that attempts to find trafficking rings hiding in plain sight.

Defective Language Compositionality. Online escort advertisements are difficult to analyze, because they lack grammatical structures such as constituency. Therefore, any form of inference must rely more on context than on grammar. This presents a significant challenge to the NLP community. Furthermore, the majority of the ads contain emojis and non-English characters.

Generalizable Language Context. Machine learning techniques can easily learn unreliable cues in training sets such as phone numbers, keywords, and other forms of semantically unreliable discriminators to reduce the training loss. Due to limited similarity between the training and test data due to the large number of ads available online, relying on these cues is futile. Learned discriminative features should be generalizable and model semantics of trafficking.

Multimodal Nature. Escort advertisements are composed of both textual and visual information. Our model should treat these features interdependently. For instance, if the text indicates that the escort is in a hotel room, our model should consider the effect that such knowledge may have on the importance of certain visual features.

We believe that studying human trafficking advertisements can be seen as a fundamental challenge to the NLP, computer vision, and machine learning communities dealing with language and vision problems. In this paper, we present the following contributions to this research direction. First, we study the language and vision modalities of the escort advertisements through deep neural modeling. Second, we take a significant step in automatic detection of advertisements suspected of sex trafficking. While previous methods (Dubrawski et al., 2015) have used simplistic classifiers, we build an end-to-end-trained multimodal deep model called the Human Trafficking Deep Network (HTDN). The HTDN uses information from both text and images to extract cues of human trafficking, and shows outstanding performance compared to previously used models. Third, we present the first rigorously annotated dataset for detection of human trafficking, called Trafficking-10k, which includes more than 10,000 trafficking ads labeled with likelihoods of having been posted by traffickers.[1]

[1]Due to the sensitive nature of this dataset, access can only be granted by emailing Cara Jones. Different levels of access are provided *only* to scientific community.

2 Related Works

Automatic detection of human trafficking has been a relatively unexplored area of machine learning research. Very few machine learning approaches have been proposed to detect signs of human trafficking online. Most of these approaches use simplistic methods such as multimedia matching (Zhou et al., 2016), text-based filtering classifiers such as random forests, logistic regression, and SVMs (Dubrawski et al., 2015), and named-entity recognition to isolate the instances of trafficking (Nagpal et al., 2015). Studies have suggested using statistical methods to find keywords and signs of trafficking from data to help law enforcement agencies (Kennedy, 2012) as well as adult content filtering using textual information (Zhou et al., 2016).

Multimodal approaches have gained popularity over the past few years. These multimodal models have been used for medical purposes, such as detection of suicidal risk, PTSD and depression (Scherer et al., 2016; Venek et al., 2016; Yu et al., 2013; Valstar et al., 2016); sentiment analysis (Zadeh et al., 2016b; Poria et al., 2016; Zadeh et al., 2016a); emotion recognition (Poria et al., 2017); image captioning and media description (You et al., 2016; Donahue et al., 2015); question answering (Antol et al., 2015); and multimodal translation (Specia et al., 2016).

To the best of our knowledge, this paper presents the first multimodal and deep model for detection of human trafficking.

3 Trafficking-10k Dataset

In this section, we present the dataset for our studies. We formalize the problem of recognizing sex trafficking as a machine learning task. The input data is text and images; this is mapped to a measure of how suspicious the advertisement is with regards to human trafficking.

3.1 Data Acquisition and Preprocessing

A subset of 10,000 ads were sampled randomly from a large cache of escort ads for annotation in Trafficking-10k dataset. The distribution of advertisements across the United States and Canada is shown in Figure 1, which indicates the diversity of advertisements in Trafficking-10k. This diversity ensures that models trained on Trafficking-10k can be applicable nationwide. The 10,000 collected ads

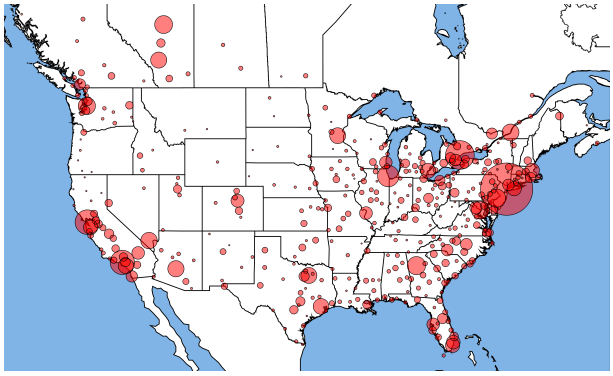


Figure 1: Distribution of advertisements in Trafficking-10k dataset across United States and Canada.

each consist of text and zero or more images. The text in the dataset is in plain text format, derived by stripping the HTML tags from the raw source of the ads. The set of characters in each advertisement is encoded as UTF-8, because there is ample usage of smilies and non-English characters. Advertisements are truncated to the first 184 words, as this covers more than 90% of the ads. Images are resized to 224×224 pixels with RGB channels.

3.2 Trafficking Annotation

Detecting whether or not an advertisement is suspicious requires years of practice and experience in working closely with law enforcement. As a result, annotation is a highly complicated and expensive process, which cannot be scaled using crowdsourcing. In our dataset, annotation is carried out by two expert annotators, each with at least five years of experience, in detection of human trafficking and another annotator with one year of experience. In our dataset, annotations were done by three experts. One expert has over a year of experience, and the other two have over five years of experience in the human trafficking domain. To calculate the inter-annotator agreement, each annotator is given the same set of 1000 ads to annotate and the nominal agreement is found: there was a 83% pairwise agreement (0.62 Krippendorff's alpha). Also, to make sure that annotations are generalizable across the annotators and law enforcement officers, two law enforcement officers annotated, respectively, a subset of 500 and 100 of the advertisements. We found a 62% average pairwise agreement (0.42 Krippendorff's alpha) with our annotators. This gap is reasonable, as law enforcement officers only have experience with local advertisements, while Trafficking-10k annotators have experience with cases across the United States.

Annotators used an annotation interface specifically designed for the Trafficking-10k dataset. In the annotation interface, each advertisement was displayed on a separate webpage. The order of the advertisements is determined uniformly randomly, and annotators were unable to move to the next advertisement without labeling the current one. For each advertisement, the annotator was presented with the question: "In your opinion, would you consider this advertisement suspicious of human trafficking?" The annotator is presented with the following options: "Certainly no," "Likely no," "Weakly no," "Unsure,"[2] "Weakly yes," "Likely yes," and "Certainly yes." Thus, the degree to which advertisements are suspicious is quantized into seven levels.

3.3 Analysis of Language

The language used in these advertisements introduces fundamental challenges to the field of NLP. The nature of the textual content in these advertisements raises the question of how we can make inferences in a linguistic environment with a constantly evolving lexicon. Language used in the Trafficking-10k dataset is highly inconsistent with standard grammar. Often, words are obfuscated by emojis and symbols. The word ordering is inconsistent, and there is rarely any form of constituency. This form of language is completely different from spoken and written English. These attributes make escort advertisements appear somewhat similar to tweets, specifically since these ads are normally short (more than 90% of the ads have at most 184 words). Another point of complexity in these advertisements is the high number of unigrams, due to usage of uncommon words and obfuscation. On top of unigram complexity, advertisers continuously change their writing pattern, making this problem more complex.

3.4 Dataset Statistics

There are 106,954 distinct unigrams, 353,324 distinct bigrams, and 565,403 trigrams in the Trafficking-10k dataset. There are 60,337 images. The total number of distinct characters including whitespace, punctuations, and hex characters is 182. The average length of an ad is 137 words, with a

[2]This option is greyed out for 10 seconds to encourage annotators to make an intuitive decision.

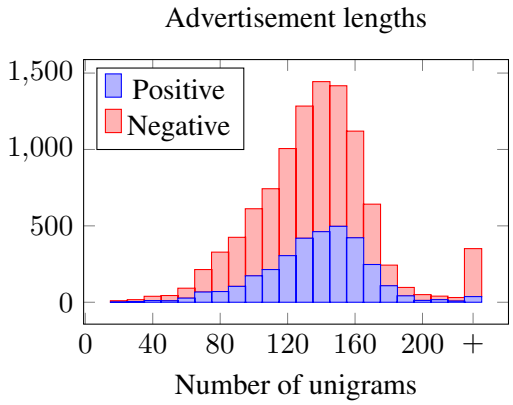


Figure 2: Distribution of the length of advertisements in Trafficking-10k. There is no significant difference between positive and negative cases purely based on length.

standard deviation of 74, median 133. The shortest advertisement has 7 unigrams, and the longest advertisement has 1810 unigrams. There are of 106,954 distinct unigrams, 353,324 distinct bigrams and 565,403 trigrams in the Trafficking-10k dataset. The average number of images in an advertisement is 5.9; the median is 5, the minimum is 0, and the maximum is 90.

The length of *suspected* advertisements is 134 unigrams; the standard deviation is 39, the minimum is 12, and the maximum is 666. The length of *non-suspected* ads is 141; the standard deviation is 85, the minimum is 7, and the maximum is 1810. The total number of suspected ads is 3257; and the total number of non-suspected ads is 6992. Figure 2 shows the histogram of number of ads based on their length. Both the positive and negative distributions are similar. This means that there is no obvious length difference between the two classes. Most of the ads have a length of 80–180 words.

4 Model

In this section, we present our deep multimodal network called the Human Trafficking Deep Network (HTDN). The HTDN is a multimodal network with language and vision components. The input to the HTDN is an ad, text and images. The HTDN is shown in Figure 3. In the remainder of this section, we will outline the different parts of the HTDN, and the input features to each component.

4.1 Trafficking Word Embeddings

Our approach to deal with the adversarial environment of escort ads is to use word vectors, defining words not based on their constituent characters, but rather based on their context. For instance, consider the two unigrams "cash" and "©a$h." While these contain different characters, semantically they are the same, and they occur in the same context. Thus, our expectation is that both the unigrams will be mapped to similar vectors. Word embeddings pretrained on general domains do not cover most of the unigrams in Trafficking-10k. For instance, the GloVe embedding (Pennington et al., 2014) trained on Wikipedia covers only 49.7% of our unigrams. The first step of the HTDN pipeline is to train word vectors (Mikolov et al., 2013) based on the skip-gram model. This is especially suitable for escort ads, because skip-gram models are able to capture context without relying on word order. We train the word embedding using 1,000,000 unlabeled ads from a dataset that does not include the Trafficking-10k data. For each advertisement, the input to the trained embedding is a sequence of words $\hat{\mathbf{w}} = [\hat{w}_1, \ldots, \hat{w}_t]$, and the output is a sequence of 100-dimensional word vectors $\mathbf{w} = [w_1, \ldots, w_t]$, where t is the size of the advertisement and $w_i \in \mathbb{R}^{100}$. Our trained word vectors cover 94.9% of the unigrams in the Trafficking-10k dataset.

4.2 Language Network

Our language network is designed to deal with two challenging aspects of escort advertisements: (1) violation of constituency, and (2) presence of irrelevant information not related to trafficking but present in ads. We address both of these issues by learning a time dependent embedding at word level. This allows the model to not rely on constituency and also remember useful information from the past, should the model get overwhelmed by irrelevant information. Our proposed language network, $\mathcal{F}_l$, takes as input a sequence of word vectors $\mathbf{w} = [w_1, \ldots, w_t]$, and outputs a neural language representation h_l. As a first step, $\mathcal{F}_l$ uses the word embeddings as input to a Long-Short Term Memory (LSTM) network and produces a new supervised context-aware word embedding $\mathbf{u} = [u_1, \ldots, u_t]$ where $u_i \in \mathbb{R}^{300}$ is the output of the LSTM at time i. Then, $\mathbf{u}$ is fed into a fully connected layer with dropout $p = 0.5$ to produce the neural language representation $h_l \in \mathbb{R}^{300}$ according to the following formulas with weights W_l for the LSTM and implicit weights in the fully

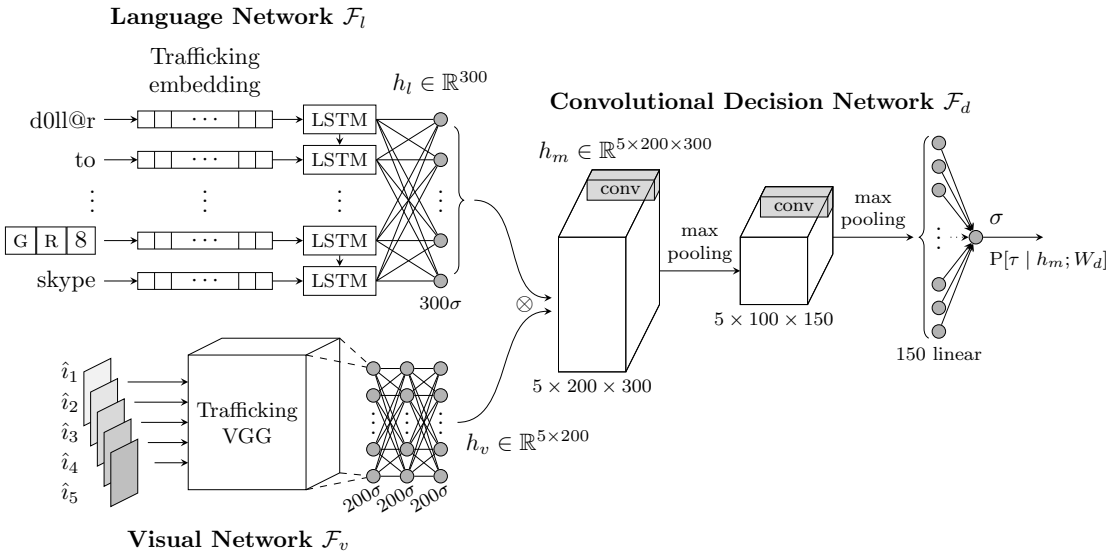


Figure 3: Overview of our proposed Human Trafficking Deep Network (HTDN). The input to HTDN is text and a set of 5 images. The text goes through the Language Network $\mathcal{F}_l$ to get the language representation h_l and the set of 5 images go through the Vision Network $\mathcal{F}_v$ to get the visual representation h_v. h_l and h_v are then fused together to get the multimodal representation h_m. The Convolutional Decision Network $\mathcal{F}_d$ conditioned on the h_m makes inference about whether or not the advertisement is suspected of trafficking

connected layers, which we represent by FC:

$$u_i = LSTM(i, w_i; W_l) \quad (1)$$

$$\mathbf{u} = [u_1, \ldots, u_t] \quad (2)$$

$$h_l = FC(\mathbf{u}). \quad (3)$$

The generated h_l is then used as part of the HTDN pipeline, and is also trained independently to assess the performance of the language-only model. The language network $\mathcal{F}_l$ is the combination of the LSTM and the fully-connected network.

4.3 Vision Network

Parallel to the language network, the vision network $\mathcal{F}_v$ takes as input advertisement images and extracts visual representations h_v. The vision network takes at most five images; the median number of images per advertisement in Trafficking-10k is 5. To learn contextual and abstract information from images, we use a deep convolutional neural network called Trafficking-VGG (T-VGG), a fine-tuned instance of the well-known VGG network (Simonyan and Zisserman, 2014). T-VGG is a deep model with 13 consecutive convolutional layers followed by 2 fully connected layers; it does not include the softmax layer of VGG. The procedure for fine-tuning T-VGG maps each individual image to a label that comes from the advertisement, and then performs end-to-end training. For example, if there are five images in an advertisement with positive label, all five images are mapped to positive label. After fine-tuning, three fully connected layers of 200 neurons with dropout $p = 0.5$ are added to the network. The combination of T-VGG and the fully connected layers is the vision network $\mathcal{F}_l$. We consider five images $\hat{\boldsymbol{i}} = \{\hat{i}_1, \ldots, \hat{i}_5\}$ from each input advertisement. If the advertisement has fewer than five images, zero-filled images are added. For each image, the output of $\mathcal{F}_v$ is a representation of five images $\mathbf{i} = \{i_1, \ldots, i_5\}$. The visual representation $h_v \in \mathbb{R}^{5\times200}$ is a matrix with a size-200 representation of each of the 5 images:

$$h_v = \mathcal{F}_v(\hat{\boldsymbol{i}}; W_v). \quad (4)$$

4.4 Multimodal Fusion

Escort advertisements have complex dynamics between text and images. Often, neither linguistic nor visual cues alone can suffice to classify whether an ad is suspicious. Interactions between linguistic and visual cues can be non-trivial, so this requires an explicit joint representation for each neuron in the linguistic and visual representations. In our multimodal fusion approach we address this by calculating an outer product between language and visual representations h_l and h_v to build the full space of possible outcomes:

$$h_m = h_l \otimes h_v, \quad (5)$$

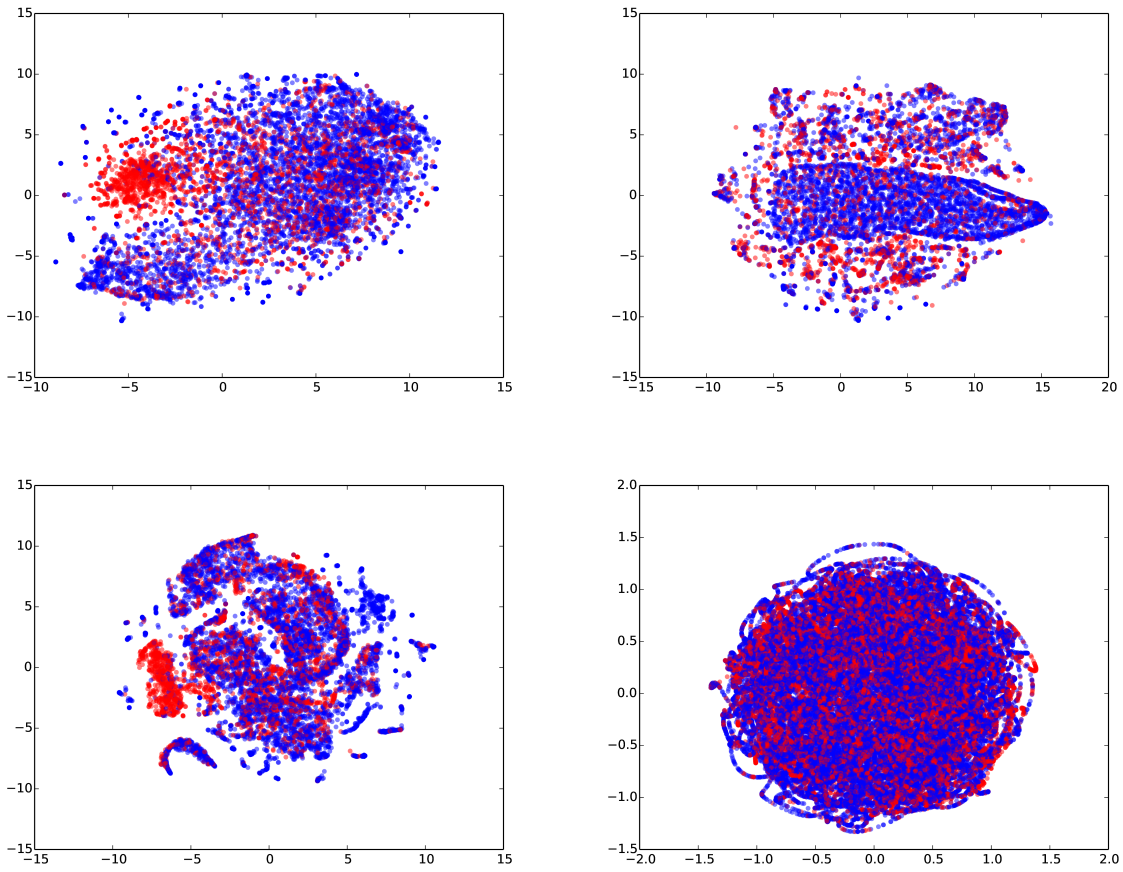


Figure 4: 2D t-SNE representation of different input features for baseline models. Clockwise from top left: one hot vectors with expert data, one hot vectors without expert data, visual features from Vision Network $\mathcal{F}_v$, and average word vectors. These representations show that inference is not trivial in Trafficking-10k dataset.

where $\otimes$ is an outer product of the two representations. This creates a joint multimodal tensor called h_m for language and visual modalities. In this tensor, every neuron in the language representation is multiplied by every neuron in vision representation, thus creating a new representation containing the information of both of them. Thus, the final fusion tensor $h_m \in \mathbb{R}^{5\times200\times300}$ contains information from the joint interaction of the language and visual modalities.

4.5 Convolutional Decision Network

The multimodal representation h_m is used as the input to the convolutional decision network $\mathcal{F}_d$. $\mathcal{F}_d$ has two layers of convolution and max pooling with a dropout rate of $p = 0.5$, followed by a fully connected layer of 150 neurons with a dropout rate of $p = 0.5$. Performing convolutions in this space enables the model to attend to small areas of linguistic and visual cues. It can thus find correspondences between specific combinations of the linguistic and visual representations. The final decision is made by a single sigmoid neuron.

5 Experiments

In our experiments, we compare the HTDN with previously used approaches for detection of trafficking suspicious ads. Furthermore, we compare the HTDN to the performance of its unimodal components. In all our experiments we perform binary classification of whether the advertisement is suspected of being related to trafficking. The main comparison method that we use is the weighted accuracy and F1-score (due to imbalance it dataset). The formulation for weighted accuracy is as follows:

$$\text{Wt. Acc.} = \frac{\text{TP} \times \text{N/P} + \text{TN}}{2\text{N}} \tag{6}$$

Model	Wt. Acc. (%)	F1 (%)	Acc. (%)	Precision (%)	Recall (%)
Random	50.0	-	68.2	-	-
Keywords					
Random Forest	67.0	55.2	78.1	78.2	42.6
Logistic Regression	69.9	57.8	78.4	75.5	46.8
Linear SVM	69.5	57.0	78.6	78.0	44.9
Average Trafficking Vectors					
Random Forest	67.3	54.1	78.0	79.3	41.1
Logistic Regression	72.2	61.7	80.2	79.2	50.6
Linear SVM	70.3	57.7	79.2	80.7	44.9
108 One-Hot					
Random Forest	62.4	60.7	72.6	61.5	60.0
Logistic Regression	62.5	45.1	72.2	60.0	36.1
Linear SVM	61.7	45.1	71.8	58.6	36.7
Bag of Words					
Random Forest	57.6	24.5	70.4	63.2	15.2
Logistic Regression	71.1	24.5	70.4	63.2	15.2
Linear SVM	71.2	24.5	70.4	63.2	15.2
HTDN Unimodal					
$\mathcal{F}_l$	74.5	65.8	78.8	69.8	62.3
$\mathcal{F}_v$ [VGG]	69.1	58.4	74.2	66.7	52.0
$\mathcal{F}_v$ [T-VGG]	70.4	59.5	77.3	78.3	48.0
HTDN	**75.3**	**66.5**	80.0	71.4	62.2
Human	83.7	73.7	84.0	76.7	70.9

Table 1: Results of our experiments. We compare our HTDN model to various baselines using different inputs. HTDN ourperforms other baselines in both weighted accuracy and F-score.

where TP (resp. TN) is true positive (resp. true negative) predictions, and P (resp. N) is the total number of positive (resp. negative) examples.

5.1 Baselines

We compare the performance of the HTDN network with baseline models divided in 4 major categories

Bag-of-Words Baselines. This set of baselines is designed to assess performance of off-the-shelf basic classifiers and basic language features. We train random forest, logistic regression and linear SVMs to show the performance of simple language-only models.

Keyword Baselines. These demonstrate the performance of models that use a set of 108 keywords, all highly related to trafficking, provided by law enforcement officers.[3] A binary one-hot vector representing these keywords is used to train the random forest, logistic regression, and linear SVM models.

108 One-Hot Baselines. Similar to Keywords Baseline, we use feature selection technique to filter the most informative 108 words for detection of trafficking. We compare the performance of this baseline to Keywords baseline to evaluate the usefulness of expert knowledge in keywords selection vs automatic data-driven keyword selection.

Average Trafficking Vectors Baselines. We assess the magnitude of success for the trafficking word embeddings for different classifiers. For the random forest, logistic regression, and linear SVM models, the average word vector is calculated and used as input.

HTDN Unimodal. These baselines show the performance of unimodal components of HTDN. For language we only use $\mathcal{F}_l$ component of the pipeline and for visual we use $\mathcal{F}_v$, using both pre-trained a VGG and finetuned T-VGG.

[3]Not presented in this paper due to sensitive nature of these keywords.

Random and Human. Random is based on assigning the more frequent class in training set to all the test data, and can be considered a lower bound for our model. Human performance metrics are upper bounds for this task's metrics.

We visualize the different inputs to our baseline models to show the complexity of the dataset when using different feature sets. Figure 4 shows the 2D t-SNE (Maaten and Hinton, 2008) representation of the training data in our dataset according to the Bag-of-Words (top right) models, expert keywords (top left), average word vectors (bottom right), and the visual representation h_v bottom left. The distribution of points suggests that none of the feature representations make the classification task trivial.

5.2 Training Parameters

All the models in our experiments are trained on the Trafficking-10k designated training set and tested on the designated test set. Hyperparameter evaluation is performed using a subset of training set as validation set. The HTDN model is trained using the Adam optimizer (Kingma and Ba, 2014). The neural weights were initialized randomly using Xavier initialization technique (Glorot and Bengio, 2010). The random forest model uses 10 estimators, with no maximum depth, and minimum-samples-per-split value of 2. The linear SVM model uses an ℓ_2-penalty and a square hinge loss with $C = 1$.

6 Results and Discussion

The results of our experiments are shown in Table 1. We report the results on three metrics: F1-score, weighted accuracy, and accuracy. Due to the imbalance between the numbers of positive and negative samples, weighted accuracy is more informative than unweighted accuracy, so we focus on the former.

HTDN. The first observation from Table 1 is that the HTDN model outperforms all the proposed baselines. There is a significant gap between the HTDN (and variants) and other non-neural approaches. This better performance is an indicator of complex interactions in detecting dynamics of human trafficking, which is captured by the HTDN.

Both Modalities are Helpful. Both modalities are helpful in predicting signs of trafficking ($\mathcal{F}_l$ and $\mathcal{F}_v$ [T-VGG]). Fine-tuning VGG network parameters shows improvement over pre-trained VGG parameters.

Language is More Important. Since $\mathcal{F}_l$ shows better performance than $\mathcal{F}_v$ [T-VGG], the language modality appears to be the more informative modality for detecting trafficking suspicious ads.

7 Conclusion and Future Work

In this paper, we took a major step in multimodal modeling of suspected online trafficking advertisements. We presented a novel dataset, Trafficking-10k, with more than 10,000 advertisements annotated for this task. The dataset contains two modalities of information per advertisement: text and images. We designed a deep multimodal model called the Human Trafficking Deep Network (HTDN). We compared the performance of the HTDN to various models that use language and vision alone. The HTDN outperformed all of these, indicating that using information from both sources may be more helpful than using just one.

Exploring language through character modeling. In order to eliminate the need for retraining the word vectors as the language of the domain evolves, we plan to use character models to learn a better language model for trafficking. As new obfuscated words are introduced in escort advertisements, our hope is that character models will stay invariant to these obfuscations.

Understanding images. While CNNs have proven to be useful for many different computer vision tasks, we seek to improve the learning capability of the visual network. Future direction involves using graphical modeling to understand interactions in the scene. Another direction involves working to understand text in images, which can provide more information about the subjects of the images.

Given that the current state of the art in this area generally does not use deep models, this may be a major opportunity for improvement. To this end, we encourage the research community to reach out to Cara Jones, an author of this paper, to obtain a copy of Trafficking-10k and other training data.

Acknowledgements

We would like to thank William Chargin for creating figures and revising this paper. We would also like to thank Torsten Wörtwein for his assistance in visualizing our data. Furthermore, we would like to thank our anonymous reviewers for their valuable feedback. Finally, we would like to acknowledge collaborators from Marinus Analytics for the time and effort that they put into annotating advertise-

ments for the dataset, and for allowing us to use their advertisement data.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2425–2433.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 2625–2634.

Artur Dubrawski, Kyle Miller, Matthew Barnes, Benedikt Boecking, and Emily Kennedy. 2015. Leveraging publicly available data to discern patterns of human-trafficking activity. *Journal of Human Trafficking* 1(1):65–85.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*. volume 9, pages 249–256.

National Human Trafficking Hotline. ???? Hotline statistics.

Emily Kennedy. 2012. Predictive patterns of sex trafficking online. *Dietrich College Honors Theses* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Lauren A McCarthy. 2014. Human trafficking and the new slavery. *Annual Review of Law and Social Science* 10:221–242.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Chirag Nagpal, Kyle Miller, Benedikt Boecking, and Artur Dubrawski. 2015. An entity resolution approach to isolate instances of human trafficking online. *arXiv preprint arXiv:1509.06659* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Soujanya Poria, Erik Cambria, Rajiv Bajpai, and Amir Hussain. 2017. A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion* 1:34.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, pages 439–448.

Stefan Scherer, Gale M Lucas, Jonathan Gratch, Albert Skip Rizzo, and Louis-Philippe Morency. 2016. Self-reported symptoms of depression and ptsd are associated with reduced vowel space in screening interviews. *IEEE Transactions on Affective Computing* 7(1):59–73.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation, Berlin, Germany. Association for Computational Linguistics*.

UNODC. 2008. Human trafficking: An overview. Web, New York. http://www.ungift.org/doc/knowledgehub/resource-centre/GIFT˙Human˙Trafficking˙An˙Overview˙2008.pdf.

Michel Valstar, Jonathan Gratch, Björn Schuller, Fabien Ringeval, Dennis Lalanne, Mercedes Torres Torres, Stefan Scherer, Giota Stratou, Roddy Cowie, and Maja Pantic. 2016. Avec 2016: Depression, mood, and emotion recognition workshop and challenge. In *Proceedings of the 6th International Workshop on Audio/Visual Emotion Challenge*. ACM, pages 3–10.

Verena Venek, Stefan Scherer, Louis-Philippe Morency, Albert Rizzo, and John Pestian. 2016. Adolescent suicidal risk assessment in clinician-patient interaction. *IEEE Transactions on Affective Computing* .

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 4651–4659.

Zhou Yu, Stefen Scherer, David Devault, Jonathan Gratch, Giota Stratou, Louis-Philippe Morency, and Justine Cassell. 2013. Multimodal prediction of psychological disorders: Learning verbal and nonverbal commonalities in adjacency pairs. In *Semdial 2013 DialDam: Proceedings of the 17th Workshop on the Semantics and Pragmatics of Dialogue*. pages 160–169.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016a. Mosi: Multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *arXiv preprint arXiv:1606.06259* .

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016b. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems* 31(6):82–88.

Andrew Jie Zhou, Jiyun Luo, and Lewis John McGibbney. 2016. Multimedia metadata-based forensics in human trafficking web data. *Vanessa Murdock, Charles LA Clarke, Jaap* page 10.

MalwareTextDB: A Database for Annotated Malware Articles

Swee Kiat Lim and **Aldrian Obaja Muis** and **Wei Lu**
Singapore University of Technology and Design
8 Somapah Road, Singapore, 487372
limsweekiat@gmail.com, {aldrian_muis,luwei}@sutd.edu.sg

Chen Hui Ong
DSO National Laboratories
20 Science Park Drive, Singapore, 118230
ochenhui@dso.org.sg

Abstract

Cybersecurity risks and malware threats are becoming increasingly dangerous and common. Despite the severity of the problem, there has been few NLP efforts focused on tackling cybersecurity.

In this paper, we discuss the construction of a new database for annotated malware texts. An annotation framework is introduced based around the MAEC vocabulary for defining malware characteristics, along with a database consisting of 39 annotated APT reports with a total of 6,819 sentences. We also use the database to construct models that can potentially help cybersecurity researchers in their data collection and analytics efforts.

1 Introduction

In 2010, the malware known as Stuxnet physically damaged centrifuges in Iranian nuclear facilities (Langner, 2011). More recently in 2016, a botnet known as Mirai used infected Internet of Things (IoT) devices to conduct large-scale Distributed Denial of Service (DDoS) attacks and disabled Internet access for millions of users in the US West Coast (US-CERT, 2016). These are only two cases in a long list ranging from ransomeware on personal laptops (Andronio et al., 2015) to taking over control of moving cars (Checkoway et al., 2011). Attacks such as these are likely to become increasingly frequent and dangerous as more devices and facilities become connected and digitized.

Recently, *cybersecurity defense* has also been recognized as one of the "*problem areas likely to be important both for advancing AI and for its long-run impact on society*" (Sutskever et al., 2016). In particular, we feel that natural language processing (NLP) has the potential for substantial contribution in cybersecurity and that this is a critical research area given the urgency and risks involved.

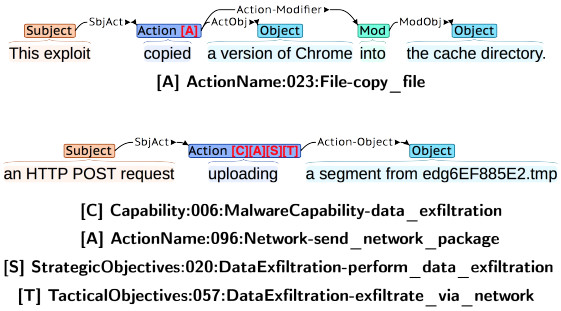


Figure 1: Annotated sentence and sentence fragment from MalwareTextDB. Such annotations provide semantic-level information to the text.

There exists a large repository of malware-related texts online, such as detailed malware reports by various cybersecurity agencies such as Symantec (DiMaggio, 2015) and Cylance (Gross, 2016) and in various blog posts. Cybersecurity researchers often consume such texts in the process of data collection. However, the sheer volume and diversity of these texts make it difficult for researchers to quickly obtain useful information. A potential application of NLP can be to quickly highlight critical information from these texts, such as the specific actions taken by a certain malware. This can help researchers quickly understand the capabilities of a specific malware and search in other texts for malware with similar capabilities.

An immediate problem preventing application of NLP techniques to malware texts is that such

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1557–1567
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1143

texts are mostly unannotated. This severely limits their use in supervised learning techniques.

In light of that, we introduce a database of annotated malware reports for facilitating future NLP work in cybersecurity. To the best of our knowledge, this is the first database consisting of annotated malware reports. It is intended for public release, where we hope to inspire contributions from other research groups and individuals.

The main contributions of this paper are:

- We initiate a framework for annotating malware reports and annotate 39 Advanced Persistent Threat (APT) reports (containing 6,819 sentences) with attribute labels from the Malware Attribute Enumeration and Characterization (MAEC) vocabulary (Kirillov et al., 2010).
- We propose the following tasks, construct models for tackling them, and discuss the challenges:
  - Classify if a sentence is useful for inferring malware actions and capabilities,
  - Predict token, relation and attribute labels for a given malware-related text, as defined by the earlier framework, and
  - Predict a malware's signatures based only on text describing the malware.

2 Background

2.1 APTnotes

The 39 APT reports in this database are sourced from APTnotes, a GitHub repository of publicly-released reports related to APT groups (Blanda, 2016). The repository is constantly updated, which means it is a constant source of reports for annotations. While the repository consists of 384 reports (as of writing), we have chosen 39 reports from the year 2014 to initialize the database.

2.2 MAEC

The MAEC vocabulary was devised by The MITRE Corporation as a standardized language for describing malware (Kirillov et al., 2010). The MAEC vocabulary is used as a source of labels for our annotations. This will facilitate cross-applications in other projects and ensure relevance in the cybersecurity community.

2.3 Related Work

There are datasets available, which are used for more general tasks such as content extraction (Walker et al., 2006) or keyword extraction (Kim et al., 2010). These may appear similar to our dataset. However, a big difference is that we are not performing general-purpose annotation and not all tokens are annotated. Instead, we only annotated tokens relevant to malware capabilities and provide more valuable information by annotating the type of malware capability or action implied. These are important differentiating factors, specifically catered to the cybersecurity domain.

While we are not aware of any database catering specifically to malware reports, there are various databases in the cybersecurity domain that provide malware hashes, such as the National Software Reference Library (NSRL) (NIST, 2017; Mead, 2006) and the File Hash Repository (FHR) by the Open Web Application Security Project (OWASP, 2015).

Most work on classifying and detecting malware has also been focusing on detecting system calls (Alazab et al., 2010; Briones and Gomez, 2008; Willems et al., 2007; Qiao et al., 2013). More recently, Rieck et al. (2011) has incorporated machine learning techniques for detecting malware, again through system calls. To the best of our knowledge, we are not aware of any work on classifying malware based on analysis of malware reports. By building a model that learns to highlight critical information on malware capabilities, we feel that malware-related texts can become a more accessible source of information and provide a richer form of malware characterization beyond detecting file hashes and system calls.

3 Data Collection

We worked together with cybersecurity researchers while choosing the preliminary dataset, to ensure that it is relevant for the cybersecurity community. The factors considered when selecting the dataset include the mention of most current malware threats, the range of author sources, with blog posts and technical security reports, and the range of actor attributions, from several suspected state actors to smaller APT groups.

3.1 Preprocessing

After the APT reports have been downloaded in PDF format, the PDFMiner tool (Shinyama, 2004) is used to convert the PDF files into plaintext format. The reports often contain non-sentences, such as the cover page or document header and

footer. We went through these non-sentences manually and subsequently removed them before the annotation. Hence only complete sentences are considered for subsequent steps.

3.2 Annotation

The Brat Rapid Annotation Tool (Stenetorp et al., 2012) is used to annotate the reports. The main aim of the annotation is to map important word phrases that describe malware actions and behaviors to the relevant MAEC vocabulary, such as the ones shown in Figure 1. We first extract and enumerate the labels from the MAEC vocabulary, which we call attribute labels. This gives us a total of 444 attribute labels, consisting of 211 ActionName labels, 20 Capability labels, 65 StrategicObjectives labels and 148 TacticalObjectives labels. These labels are elaborated in Section 3.5.

There are three main stages to the annotation process. These are cumulative and eventually build up to the annotation of the attribute labels.

3.3 Stage 1 - Token Labels

The first stage involves annotating the text with the following *token labels*, illustrated in Figure 2:

Action This refers to an event, such as "*registers*", "*provides*" and "*is written*".

Subject This refers to the initiator of the Action such as "*The dropper*" and "*This module*".

Object This refers to the recipient of the Action such as "*itself*", "*remote persistent access*" and "*The ransom note*"; it also refers to word phrases that provide elaboration on the Action such as "*a service*", "*the attacker*" and "*disk*".

Modifier This refers to tokens that link to other word phrases that provide elaboration on the Action such as "*as*" and "*to*".

This stage helps to identify word phrases that are relevant to the MAEC vocabulary. Notice that for the last sentence in Figure 2, "*The ransom note*" is tagged as an Object instead of a Subject. This is because the Action "*is written*" is not being initiated by "*The ransom note*". Instead, the Subject is absent in this sentence.

3.4 Stage 2 - Relation Labels

The second stage involves annotating the text with the following *relation labels*:

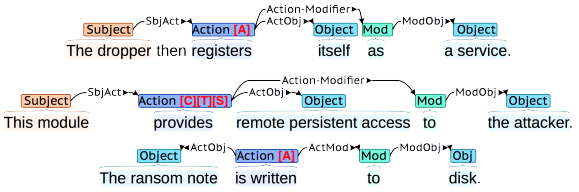


Figure 2: Examples of annotated sentences.

The regime's greatest fear is internal dissent and resulting destabilization.

Songun is North Korea's "military first" doctrine.

Figure 3: Examples of irrelevant sentences.

SubjAction This links an Action with its relevant Subject.

ActionObj This links an Action with its relevant Object.

ActionMod This links an Action with its relevant Modifier.

ModObj This links a Modifier with the Object that provides elaboration.

This stage helps to make the links between the labelled tokens explicit, which is important in cases where a single Action has multiple Subjects, Objects or Modifiers. Figure 2 demonstrates how the relation labels are used to link the token labels.

3.5 Stage 3 - Attribute Labels

The third stage involves annotating the text with the *attribute labels* extracted from the MAEC vocabulary. Since the Action is the main indicator of a malware's action or capability, the attribute labels are annotated onto the Actions tagged in Stage 1. Each Action should have one or more attribute labels.

There are four classes of attribute labels: ActionName, Capability, StrategicObjectives and TacticalObjectives. These labels describe different actions and capabilities of the malware. Refer to Appendix A for examples and elaboration.

3.6 Summary

The above stages complete the annotation process and is done for each document. There are also sentences that are not annotated at all since they do not provide any indication of malware actions or capabilities, such as the sentences in Figure 3. We call these sentences *irrelevant sentences*.

At the time of writing, the database consists of 39 annotated APT reports with a combined total of 6,819 sentences. Out of the 6,819 sentences,

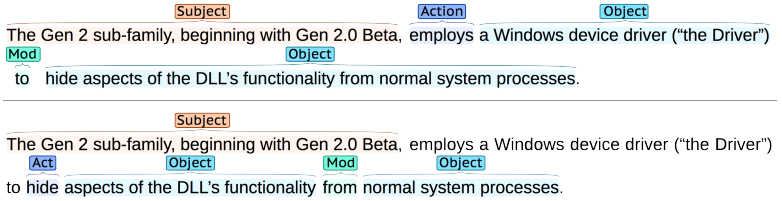


Figure 4: Two different ways for annotating a sentence, where both seem to be equally satisfactory to a human annotator. In this case, both serve to highlight the malware's ability to hide its DLL's functionality.

Token Labels (by label)		**Relation Labels** (by label)		**Attribute Labels** (by class)	
Subj	1,778	SubjAction	2,343	ActionName	982
Obj	4,411	ActionObj	2,713	Capability	2,524
Act	2,975	ActionMod	1,841	StratObj	2,004
Mod	1,819	ModObj	1,808	TactObj	1,592
Total	**10,983**	**Total**	**8,705**	**Total**	**7,102**

Table 1: Breakdown of annotation statistics.

2,080 sentences are annotated. Table 1 shows the breakdown of the annotation statistics.

3.7 Annotators' Challenges

We can calculate the Cohen's Kappa (Cohen, 1960) to quantify the agreement between annotators and to give an estimation of the difficulty of this task for human annotators. Using annotations from pairs of annotators, the Cohen's Kappa was calculated to be 0.36 for annotation of the Token labels. This relatively low agreement between annotators suggests that this is a rather difficult task. In the following subsections, we discuss some possible reasons that make this annotation task difficult.

3.7.1 Complex Sentence Structures

In many cases, there may be no definite way to label the tokens. Figure 4 shows two ways to annotate the same sentence. Both annotations essentially serve to highlight the Gen 2 sub-family's capability of hiding the DLL's functionality. The first annotation highlights the method used by the malware to hide the library, i.e., employing the Driver. The second annotation focuses on the malware hiding the library and does not include the method. Also notice that the Modifiers highlighted are different in the two cases, since this depends on the Action highlighted and are hence mutually exclusive. Such cases occur more commonly when the sentences contain complex noun- and verb-phrases that can be decomposed in several ways. Repercussions surface later in the experiments described in Section 5.2, specifically in the second point under Discussion.

3.7.2 Large Quantity of Labels

Due to the large number (444) of attribute labels, it is challenging for annotators to remember all of the attribute labels. Moreover, some of the attribute labels are subject to interpretation. For instance, should *Capability: 005: MalwareCapability-command_and_control* be tagged for sentences that mention the location or IP addresses of command and control servers, even though such sentences may not be relevant to the capabilities of the malware?

3.7.3 Specialized Domain Knowledge Required

Finally, this task requires specialized cybersecurity domain knowledge from the annotator and the ability to apply such knowledge in a natural language context. For example, given the phrase "*load the DLL into memory*", the annotator has to realize that this phrase matches the attribute label *ActionName: 119: ProcessMemory-map_library_into_process*. The abundance of labels with the many ways that each label can be expressed in natural language makes this task extremely challenging.

4 Proposed Tasks

The main goal of creating this database is to aid cybersecurity researchers in parsing malware-related texts for important information. To this end, we propose several tasks that build up to this main goal.

Task 1 Classify if a sentence is relevant for inferring malware actions and capabilities

Task 2 Predict token labels for a given malware-related text

Task 3 Predict relation labels for a given malware-related text

Task 4 Predict attribute labels for a given malware-related text

Task 5 Predict a malware's signatures based on the text describing the malware and the text's annotations

Task 1 arose from discussions with domain experts where we found that a main challenge for cybersecurity researchers is having to sift out critical sentences from lengthy malware reports and articles. Figure 3 shows sentences describing the political and military background of North Korea in the APT report *HPSR SecurityBriefing_Episode16_NorthKorea*. Such information is essentially useless for cybersecurity researchers focused on malware actions and capabilities. It will be helpful to build a model that can filter relevant sentences that pertain to malware.

Tasks 2 to 4 serve to automate the laborious annotation procedure as described earlier. With sufficient data, we hope that it becomes possible to build an effective model for annotating malware-related texts, using the framework and labels we defined earlier. Such a model will help to quickly increase the size of the database, which in turn facilitate other supervised learning tasks.

Task 5 explores the possibility of using malware texts and annotations to predict a malware's signatures. While conventional malware analyzers generate a list of malware signatures based on the malware's activities in a sandbox, such analysis is often difficult due to restricted distribution of malware samples. In contrast, numerous malware reports are freely available and it will be helpful for cybersecurity researchers if such texts can be used to predict malware signatures instead of having to rely on a limited supply of malware samples.

In the following experiments, we construct models for tackling each of these tasks and discuss the performance of our models.

5 Experiments and Results

Since the focus of this paper is on the introduction of a new framework and database for annotating malware-related texts, we only use simple algorithms for building the models and leave more complex models for future work.

For the following experiments, we use linear support vector machine (SVM) and multinomial Naive Bayes (NB) implementations in the scikit-learn library (Pedregosa et al., 2011). The regularization parameter in SVM and smoothing parameter in NB were tuned (with the values 10^{-3} to 10^{3} in logarithmic increments) by taking the value that gave the best performance in development set.

	P	R	F_1
SVM	69.7	54.0	60.5
NB	59.5	68.5	63.2

Table 2: Task 1 scores: classifying relevant sentences.

For experiments where Conditional Random Field (CRF) (Lafferty et al., 2001) is used, we utilized the CRF++ implementation (Kudo, 2005).

For scoring the predictions, unless otherwise stated, we use the metrics module in scikit-learn for SVM and NB, as well as the CoNLL2000 conlleval Perl script for CRF[1].

Also, unless otherwise mentioned, we make use of all 39 annotated documents in the database. The experiments are conducted with a 60%/20%/20% training/development/test split, resulting in 23, 8 and 8 documents in the respective datasets. Each experiment is conducted 5 times with a different random allocation of the dataset splits and we report averaged scores[2].

Since we focus on building a database, we weigh recall and precision as equally important in the following experiments and hence focus on the F_1 score metric. The relative importance of recall against precision will ultimately depend on the downstream tasks.

5.1 Task 1 - Classify sentences relevant to malware

We make use of the annotations in our database for this supervised learning task and consider a sentence to be relevant as long as it has an annotated token label. For example, the sentences in Figure 2 will be labeled relevant whereas the sentences in Figure 3 will be labeled irrelevant.

A simple bag-of-words model is used to represent each sentence. We then build two models – SVM and NB – for tackling this task.

Results: Table 2 shows that while the NB model outperforms the SVM model in terms of F_1 score, the performance of both models are still rather low with F_1 scores below 70 points. We proceed to discuss possible sources of errors for the models.

[1] www.cnts.ua.ac.be/conll2000/chunking/output.html

[2] Note that therefore the averaged F_1 may not be the harmonic mean of averaged P and R in the result tables.

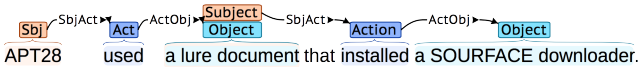


Figure 5: An example of a token ("*a lure document*") labelled as both Subject and Object. In the first case, it is the recipient of the Action "*used*", while in the latter case, it is the initiator of the Action "*installed*".

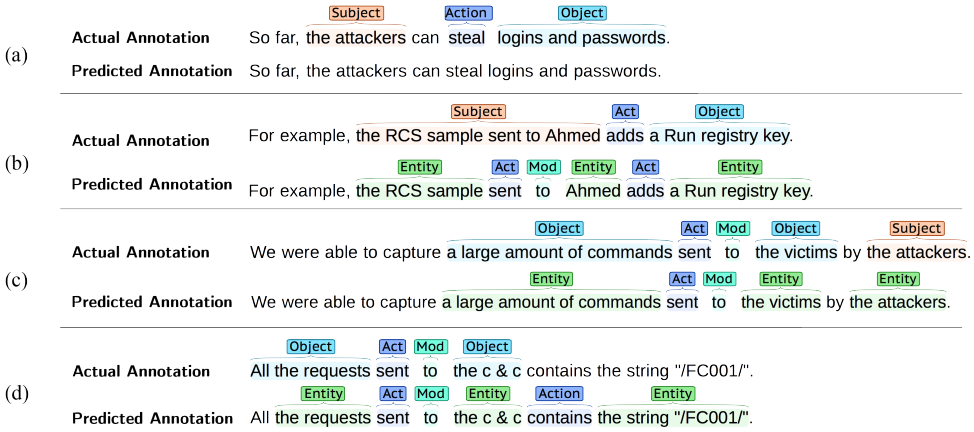


Figure 6: Actual and predicted annotations. For predicted annotations, the Entity label replaces the Subject and Object labels.

Discussion: We find that there are two main types of misclassified sentences.

1. Sentences describing malware without implying specific actions

These sentences often contain malware-specific terms, such as "*payload*" and "*malware*" in the following sentence.

This file is the main payload of the malware.

These sentences are often classified as relevant, probably due to the presence of malware-specific terms. However, such sentences are actually irrelevant because they merely describe the malware but do not indicate specific malware actions or capabilities.

2. Sentences describing attacker actions

Such sentences mostly contain the term "*attacker*" or names of attackers. For instance, the following sentence is incorrectly classified as irrelevant.

This is another remote administration tool often used by the Pitty Tiger crew.

Such sentences involving the attacker are often irrelevant since the annotations focus on the malware and not the attacker. However, the above sentence implies that the malware is a remote administration tool and hence is a relevant sentence that implies malware capability.

5.2 Task 2 - Predict token labels

Task 2 concerns automating Stage 1 for the annotation process described in Section 3.3. Within the annotated database, we find several cases where a single word-phrase may be annotated with both Subject and Object labels (see Figure 5). In order to simplify the model for prediction, we redefine Task 2 as predicting Entity, Action and Modifier labels for word-phrases. The single Entity label is used to replace both Subject and Object labels. Since the labels may extend beyond a single word token, we use the BIO format for indicating the span of the labels (Sang and Veenstra, 1999). We use two approaches for tackling this task: a) CRF is used to train a model for directly predicting token labels, b) A pipeline approach where the NB model from Task 1 is used to filter relevant sentences. A CRF model is then trained to predict token labels for relevant sentences.

The CRF model in Approach 1 is trained on the entire training set, whereas the CRF model in Approach 2 is trained only on the gold relevant sentences in the training set.

For features in both approaches, we use unigrams and bigrams, part-of-Speech labels from the Stanford POStagger (Toutanova et al., 2003), and Brown clustering features after optimizing the cluster size (Brown et al., 1992). A C++ implementation of the Brown clustering algorithm is

	Approach 1			Approach 2		
Token Label	**P**	**R**	**F_1**	**P**	**R**	**F_1**
Entity	48.8	25.1	32.9	42.8	33.8	37.6
Action	55.2	30.3	38.9	50.8	41.1	45.2
Modifier	55.7	28.4	37.3	48.9	37.4	42.1
Average	51.7	27.0	35.2	45.9	36.3	40.3

Table 3: Task 2 scores: predicting token labels.

	Approach 1			Approach 2		
Token Label	**P**	**R**	**F_1**	**P**	**R**	**F_1**
Entity	63.6	32.1	42.3	56.5	46.3	50.6
Action	60.2	31.4	41.0	54.6	42.8	47.7
Modifier	56.4	28.1	37.1	50.1	37.1	42.3
Average	62.7	31.8	41.9	55.9	45.3	49.8

Table 4: Task 2 relaxed/token-level scores.

Relation Label	**P**	**R**	**F_1**
SubjAction	86.3	82.3	84.2
ActionObj	91.6	86.2	88.8
ActionMod	98.5	96.4	97.4
ModObj	98.0	96.7	97.4
Average	89.2	89.4	89.3

Table 5: Task 3 scores: predicting relation labels.

used (Liang, 2005). The Brown cluster was trained on a larger corpus of APT reports, consisting of 103 APT reports not in the annotated database and the 23 APT reports from the training set. We group together low-frequency words that appear 4 or less times in the set of 126 APT reports into one cluster and during testing we assign new words into this cluster.

Results: Table 3 demonstrates that Approach 2 outperforms Approach 1 on most scores. Nevertheless, both approaches still give low performance for tackling Task 2 with F_1-scores below 50 points.

Discussion: There seem to be three main categories of wrong predictions:

1. Sentences describing attacker actions

Such sentences are also a main source of prediction errors in Task 1. Again, most sentences describing attackers are deemed irrelevant and left unannotated because we focus on malware actions rather than human attacker actions. However, these sentences may be annotated in cases where the attacker's actions imply a malware action or capability.

For example, the Figure 6a describes the attackers stealing credentials. This implies that the malware used is capable of stealing and exfiltrating credentials. It may be challenging for the model to distinguish whether such sentences describing attackers should be annotated since a level of inference is required.

2. Sentences containing noun-phrases made up of participial phrases and/or prepositional phrases

These sentences contain complex noun-phrases with multiple verbs and prepositions, such as in Figures 6b and 6c. In Figure 6b, "*the RCS sample sent to Ahmed*" is a noun-phrase annotated as a single Subject/Entity. However, the model decomposes the noun-phrase into the subsidiary noun "*the RCS sample*" and participial phrase "*sent to Ahmed*" and further decompose the participial phrase into the constituent words, predicting Action, Modifier and Entity labels for "*sent*", "*to*" and "*Ahmed*" respectively. There are cases where such decomposition of noun-phrases is correct, such as in Figure 6c.

As mentioned in Section 3.7, this is also a challenge for human annotators because there may be several ways to decompose the sentence, many of which serve equally well to highlight certain malware aspects (see Figure 4).

Whether such decomposition is correct depends on the information that can be extracted from the decomposition. For instance, the decomposition in Figure 6c implies that the malware can receive remote commands from attackers. In contrast, the decomposition predicted by the model in Figure 6b does not offer any insight into the malware. This is a difficult task that requires recognition of the phrase spans and the ability to decide which level of decomposition is appropriate.

3. Sentences containing noun-phrases made up of determiners and adjectives

These sentences contain noun-phrases with determiners and adjectives such as "*All the requests*" in Figure 6d. In such cases, the model may only predict the Entity label for part of the noun-phrase. This is shown in Figure 6d, where the model predicts the Entity label for "*the requests*" instead of "*All the requests*".

Thus, we also consider a relaxed scoring scheme where predictions are scored in token level instead of phrase level (see Table 4). The aim of the relaxed score is to give credit to the model when the span for a predicted label intersects with the span for the actual label, as in Figure 6d.

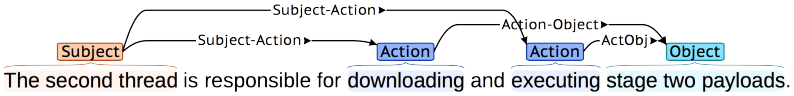


Figure 7: An example of an entity with multiple parents. In this case, *stage two payloads* has two parents by ActionObject relations - *downloading* and *executing*.

5.3 Task 3 - Predict relation labels

Following the prediction of token labels in Task 2, we move on to Task 3 for building a model for predicting relation labels. Due to the low performance of the earlier models for predicting token labels, for this experiment we decided to use the gold token labels as input into the model for predicting relation labels. Nevertheless, the models can still be chained in a pipeline context.

The task initially appeared to be similar to a dependency parsing task where the model predicts dependencies between the entities demarcated by the token labels. However, on further inspection, we realized that there are several entities which have more than one parent entity (see Figure 7). As such, we treat the task as a binary classification task, by enumerating all possible pairs of entities and predicting whether there is a relation between each pair.

Predicting the relation labels from the token labels seem to be a relatively straightforward task and hence we design a simple rule-based model for the predictions. We tuned the rule-based model on one of the documents (*AdversaryIntelligenceReport_DeepPanda_0 (1)*) and tested it on the remaining 38 documents. The rules are documented in Appendix B.

Results: Table 5 shows the scores from testing the model on the remaining 38 documents.

The results from the rule-based model are better than expected, with the average F_1-scores exceeding 84 points for all the labels. This shows that the relation labels can be reliably predicted given good predictions of the preceding token labels.

Discussion: The excellent performance from the rule-based model suggests that there is a well-defined structure in the relations between the entities. It may be possible to make use of this inherent structure to help improve the results for predicting the token labels.

Also, notice that by predicting the SubjAction, ActionObj and ActionMod relations, we are simultaneously classifying the ambiguous Entity labels into specific Subject and Object labels. For instance, Rule 1 predicts a ModObj relation between a Modifier and an Entity, implying that the Entity is an Object, whereas Rule 3 predicts a SubjAction relation between an Entity and an Action, implying that the Entity is a Subject.

Attribute Category	NB			SVM		
	P	R	F_1	P	R	F_1
ActionName	35.2	23.9	28.0	43.9	27.9	33.9
Capability	41.5	39.8	40.6	42.5	41.1	41.8
StrategicObjectives	33.7	24.4	28.3	32.2	23.5	27.2
TacticalObjectives	27.6	17.4	21.1	30.2	18.4	22.7

Table 6: Task 4 scores: predicting attribute labels.

5.4 Task 4 - Predict attribute labels

A significant obstacle in the prediction of attribute labels is the large number of attribute labels available. More precisely, we discover that many of these attribute labels occur rarely, if not never, in the annotated reports. This results in a severely sparse dataset for training a model.

Due to the lack of substantial data, we decide to use *token groups* instead of entire sentences for predicting attribute labels. Token groups are the set of tokens that are linked to each other via relation labels. We extract the token groups from the gold annotations and then build a model for predicting the attribute labels for each token group. Again, we use a bag-of-words model to represent the token groups while SVM and NB are each used to build a model for predicting attribute labels.

Results: Table 6 shows the average scores over 5 runs for the four separate attribute categories. For this task, SVM appears to perform generally better than NB, although much more data seems to be required in order to train a reliable model for predicting attribute labels. The Capability category shows the best performance, which is to be expected, since the Capability attributes occur the most frequently.

Discussion: The main challenge for this task is the sparse data and the abundant attribute labels available. In fact, out of the 444 attribute labels, 190 labels do not appear in the database. For the remaining 254 attribute labels that do occur in the database, 92 labels occur less than five times and 50 labels occur only once. With the sparse data

Features Used	NB			SVM		
	P	R	F_1	P	R	F_1
Text only	58.8	50.8	53.5	49.3	47.0	47.2
Ann. only	64.7	55.0	58.0	62.6	57.2	59.2
Text and Ann.	59.3	50.7	53.6	54.3	51.1	51.6

Table 7: Task 5 scores: predicting malware signatures using text and annotations.

available, particularly for rare attribute labels, effective one-shot learning models might have to be designed to tackle this difficult task.

5.5 Task 5 - Predict malware signatures using text and annotations

Conventional malware analyzers, such as malwr.com, generate a list of signatures based on the malware's activities in a sandbox. Examples of such signatures include *antisandbox_sleep*, which indicates anti-sandbox capabilities or *persistence_autorun*, which indicates persistence capabilities.

If it is possible to build an effective model to predict malware signatures based on natural language texts about the malware, this can help cybersecurity researchers predict signatures of malware samples that are difficult to obtain, using the malware reports freely available online.

By analyzing the hashes listed in each APT report, we obtain a list of signatures for the malware discussed in the report. However, we are unable to obtain the signatures for several hashes due to restricted distribution of malware samples. There are 8 APT reports without any obtained signatures, which are subsequently discarded for the following experiments. This leaves us with 31 out of 39 APT reports.

The current list of malware signatures from Cuckoo Sandbox[3] consists of 378 signature types. However, only 68 signature types have been identified for the malware discussed in the 31 documents. Furthermore, out of these 68 signature types, 57 signature types appear less than 10 times, which we exclude from the experiments. The experiments that follow will focus on predicting the remaining 11 signature types using the 31 documents.

The OneVsRestClassifier implementation in scikit-learn is used in the following experiments, since this is a multilabel classification problem. We also use SVM and NB to build two types of models for comparison.

Three separate methods are used to generate features for the task: a) the whole text in each APT report is used as features via a bag-of-words representation, without annotations, b) the gold labels from the annotations are used as features, without the text, and c) both the text and the gold annotations are used, via a concatenation of the two feature vectors.

Results: Comparing the first two rows in Table 7, we can see that using the annotations as features significantly improve the results, especially the precision. SVM model also seems to benefit more from the annotations, even outperforming NB in one case.

Discussion: The significant increase in precision suggests that the annotations provide a condensed source of features for predicting malware signatures, improving the models' confidence. We also observe that some signatures seem to benefit more from the annotations, such as *persistence_autorun* and *has_pdb*. In particular, *persistence_autorun* has a direct parallel in attribute labels, which is *MalwareCapability-persistence*, showing that using MAEC vocabulary as attribute labels is appropriate.

6 Conclusion

In this paper, we presented a framework for annotating malware reports. We also introduced a database with 39 annotated APT reports and proposed several new tasks and built models for extracting information from the reports. Finally, we discuss several factors that make these tasks extremely challenging given currently available models. We hope that this paper and the accompanying database serve as a first step towards NLP being applied in cybersecurity and that other researchers will be inspired to contribute to the database and to construct their own datasets and implementations. More details about this database can be found at `http://statnlp.org/research/re/`.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work is supported by ST Electronics – SUTD Cyber Security Laboratory Project 1 Big Data Security Analytics, and is partly supported by MOE Tier 1 grant SUTDT12015008.

[3] https://cuckoosandbox.org/

References

Mamoun Alazab, Sitalakshmi Venkataraman, and Paul Watters. 2010. Towards Understanding Malware Behaviour by the Extraction of API Calls. In *2010 Second Cybercrime and Trustworthy Computing Workshop*. IEEE, November 2009, pages 52–59. https://doi.org/10.1109/CTC.2010.8.

Nicoló Andronio, Stefano Zanero, and Federico Maggi. 2015. *HelDroid: Dissecting and Detecting Mobile Ransomware*, Springer International Publishing, Cham, chapter 18, pages 382–404. https://doi.org/10.1007/978-3-319-26362-5.

Kiran Blanda. 2016. APTnotes. https://github.com/aptnotes/.

Ismael Briones and Aitor Gomez. 2008. Graphs, entropy and grid computing: Automatic comparison of malware. In *Virus bulletin conference*. pages 1–12.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based N-gram Models of Natural Language. *Comput. Linguist.* 18(4):467–479. http://dl.acm.org/citation.cfm?id=176313.176316.

Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proceedings of the 20th USENIX Conference on Security*. USENIX Association, Berkeley, CA, USA, SEC'11, pages 6–6. http://dl.acm.org/citation.cfm?id=2028067.2028073.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20(1):37–46. https://doi.org/10.1177/001316446002000104.

Jon DiMaggio. 2015. The Black Vine cyberespionage group. Technical report, Symantec. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-black-vine-cyberespionage-group.pdf.

Jon Gross. 2016. Operation Dust Storm. Technical report, Cylance. https://www.cylance.com/hubfs/2015_cylance_website/assets/operation-dust-storm/Op_Dust_Storm_Report.pdf.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Stroudsburg, PA, USA, SemEval '10, pages 21–26. http://dl.acm.org/citation.cfm?id=1859664.1859668.

Ivan Kirillov, Desiree Beck, Penny Chase, and Robert Martin. 2010. Malware Attribute Enumeration and Characterization. *The MITRE Corporation, Tech. Rep* .

Taku Kudo. 2005. CRF++. https://taku910.github.io/crfpp/.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *International Conference on Machine Learning (ICML 2001)*. pages 282–289. http://dl.acm.org/citation.cfm?id=655813.

R. Langner. 2011. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Security Privacy* 9(3):49–51. https://doi.org/10.1109/MSP.2011.67.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Master's thesis, Massachusetts Institute of Technology. https://doi.org/1721.1/33296.

Steve Mead. 2006. Unique file identification in the National Software Reference Library. *Digital Investigation* 3(3):138 – 150. https://doi.org/10.1016/j.diin.2006.08.010.

NIST. 2017. National Software Reference Library. http://www.nsrl.nist.gov/.

OWASP. 2015. OWASP File Hash Repository. https://www.owasp.org/index.php/OWASP_File_Hash_Repository.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12:2825–2830. http://dl.acm.org/citation.cfm?id=2078195.

Yong Qiao, Jie He, Yuexiang Yang, and Lin Ji. 2013. Analyzing Malware by Abstracting the Frequent Itemsets in API Call Sequences. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, pages 265–270. https://doi.org/10.1109/TrustCom.2013.36.

Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. 2011. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security* 19(4):639–668. https://doi.org/10.3233/JCS-2010-0410.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing Text Chunks. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, USA, page 173. https://doi.org/10.3115/977035.977059.

Yusuke Shinyama. 2004. PDFMiner. https://euske.github.io/pdfminer/.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: A Web-based Tool for NLP-assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, EACL '12, pages 102–107. http://dl.acm.org/citation.cfm?id=2380921.2380942.

Ilya Sutskever, Dario Amodei, and Sam Altman. 2016. Special projects. Technical report, OpenAI, https://openai.com/blog/special-projects/. https://openai.com/blog/special-projects/.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '03, pages 173–180. https://doi.org/10.3115/1073445.1073478.

US-CERT. 2016. Heightened DDoS Threat Posed by Mirai and Other Botnets. Technical report, United States Computer Emergency Readiness Team. https://www.us-cert.gov/ncas/alerts/TA16-288A.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia* 57.

Carsten Willems, Thorsten Holz, and Felix Freiling. 2007. Toward Automated Dynamic Malware Analysis Using CWSandbox. *IEEE Security and Privacy Magazine* 5(2):32–39. https://doi.org/10.1109/MSP.2007.45.

A Attribute Labels

The following elaborates on the types of malware actions described by each class of attribute labels and gives specific examples.

A.1 ActionName

The ActionName labels describe specific actions taken by the malware, such as downloading a file *ActionName: 090: Network- download_ file* or creating a registry key *ActionName: 135: Registry-create_registry_key*.

A.2 Capability

The Capability labels describe general capabilities of the malware, such as exfiltrating stolen data *Capability: 006: MalwareCapability-data_exfiltration* or spying on the victim *Capability: 019: MalwareCapability-spying*.

A.3 StrategicObjectives

The StrategicObjectives labels elaborate on the Capability labels and provide more details on the capabilities of the malware, such as preparing stolen data for exfiltration *StrategicObjectives: 021: DataExfiltration-stage_data_for_exfiltration* or capturing information from input devices connected to the victim's machine *StrategicObjectives: 061: Spying-capture_system_input_peripheral_data*.

Each StrategicObjectives label belongs to a Capability label.

A.4 TacticalObjectives

The TacticalObjectives labels provide third level of details on the malware's capability, such as encrypting stolen data for exfiltration *TacticalObjectives: 053: DataExfiltration-encrypt_data* or an ability to perform key-logging *TacticalObjectives: 140: Spying-capture_keyboard_input*.

Again, each TacticalObjectives label belongs to a Capability label.

B Rules for Rule-based Model in Task 3

The following are the rules used in the rule-based model described in Section 5.3.

1. If a Modifier is followed by an Entity, a ModObj relation is predicted between the Modifier and the Entity
2. If an Action is followed by an Entity, an ActionObj relation is predicted between the Action and the Entity
3. If an Entity is followed by an Action of token-length 1, a SubjAction relation is predicted between the Entity and the Action
4. An ActionObj relation is predicted between any Action that begins with *be* and the most recent previous Entity
5. An ActionObj relation is predicted between any Action that begins with *is*, *was*, *are* or *were* and ends with *-ing* and the most recent previous Entity
6. An ActionMod relation is predicted between all Modifiers and the most recent previous Action

A Corpus of Annotated Revisions for Studying Argumentative Writing

Fan Zhang **Homa B. Hashemi** **Rebecca Hwa** **Diane Litman**
University of Pittsburgh
Pittsburgh, PA, 15260
{zhangfan,hashemi,hwa,litman}@cs.pitt.edu

Abstract

This paper presents ArgRewrite, a corpus of between-draft revisions of argumentative essays. Drafts are manually aligned at the sentence level, and the writer's purpose for each revision is annotated with categories analogous to those used in argument mining and discourse analysis. The corpus should enable advanced research in writing comparison and revision analysis, as demonstrated via our own studies of student revision behavior and of automatic revision purpose prediction.

1 Introduction

Most writing-related natural language processing (NLP) research focuses on the analysis of single drafts. Examples include document-level quality assessment (Attali and Burstein, 2006; Burstein and Chodorow, 1999), discourse-level analysis and mining (Burstein et al., 2003; Falakmasir et al., 2014; Persing and Ng, 2016), and fine-grained error detection (Leacock et al., 2010; Grammarly, 2016). Less studied is the analysis of changes *between drafts* – a comparison of revisions and the properties of the differences. Research on this topic can support applications involving revision analysis (Zhang and Litman, 2015), paraphrase (Malakasiotis and Androutsopoulos, 2011) and correction detection (Swanson and Yamangil, 2012; Xue and Hwa, 2014).

Although there are some corpora resources for NLP research on writing comparisons, most tend to be between individual sentences/phrases for tasks such as paraphrase comparison (Dolan and Brockett, 2005; Tan and Lee, 2014) or grammar error correction (Dahlmeier et al., 2013; Yannakoudakis et al., 2011). In terms of revision analysis, the most relevant work analyzes Wikipedia revisions (Daxenberger and Gurevych, 2013; Bronner and Monz, 2012); however, the domain of Wikipedia is so specialized that the properties of Wikipedia revisions do not correspond well with other kinds of texts.

This work presents the ArgRewrite corpus[1] to facilitate revision analysis research for argumentative essays. The corpus consists of a collection of three drafts of essays written by university students and employees; the drafts are manually aligned at the sentence level, then the purpose of each revision is manually coded using a revision schema closely related to argument mining/discourse analysis. Within the domain of argumentative essays, the corpus will be useful for supporting research in argumentative revision analysis and the application of argument mining techniques. The corpus may also be useful for research on paraphrase comparisons, grammar error correction, and computational stylistics (Popescu and Dinu, 2008; Flekova et al., 2016). In this paper, we present two example uses of our corpus: 1) rewriting behavior data analysis, and 2) automatic revision purpose classification.

2 Corpus Design Decisions

Consider this scenario: Alice begins her social science argumentative essay with the sentence "Electronic communication allows people to make connections beyond physical limits."

An analytical system might (rightly) identify the sentence as the thesis of her essay, and an evaluative system might give the essay a low score due to this sentence's vagueness and a later lack of evidence (though Alice may not know why she received that score).

Now suppose in a revised draft, Alice expanded

[1]The corpus is based on the ArgRewrite system developed in our prior work (Zhang et al., 2016).

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1568–1578
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1144

the sentence: "Electronic communication allows people to make connections beyond physical ~~lim-its~~ *location and enriches connections that would have been impossible to make otherwise.*"

An analytical system would still identify the sentence as the thesis, and an evaluative system might raise the overall score a little higher. Alice may become satisfied with the increase and move on. However, there is an opportunity lost – neither the analytical nor the evaluative system addressed the quality of her revision.

A revision analysis system might be helpful for Alice because it would link "limits" to "location and ..." and identify the reason why she made the change – perhaps *adding precision.* If Alice had intended her change as a way to add evidential support for her thesis, she would see that her attempt was not as successful as she hoped.

The above scenario highlights the application of a revision analysis system. This paper is about creating a corpus to enable the development of such systems. Because this is a relatively new problem, there are many possible ways for us to design the corpus. Here we discuss some of our decisions.

First, we need to define the unit of revision. The example above illustrates a phrase-aligned revision. While this offers a fairly precise definition of the scope of a revision, it may be difficult to achieve consistent annotations. For example, the changes may not adhere to any syntactic linguistic unit. For this first corpus, we define our unit of revision to be at the sentence level. In other words, even if a pair of sentences contains multiple edits, the entire sentence pair will be annotated as one sentence revision.

Second, we need to define the quality we want to observe about the revision sentence pair. For this first corpus, we focus on recognizing the purpose of the revision, as in the example above. It is a useful property, and it has previously been studied by others in the literature. People have considered both binary purpose categories such as Content vs. Surface (Faigley and Witte, 1981) or Factual vs. Fluency (Bronner and Monz, 2012) as well as more fine-grained categories (Pfeil et al., 2006; Jones, 2008; Liu and Ram, 2009; Daxenberger and Gurevych, 2012; Zhang and Litman, 2015). Our corpus follows the two-tiered schema used by (Zhang and Litman, 2015) (see Section 3.2).

Third, we not only have to decide on the annotation format, we also need to decide how to obtain

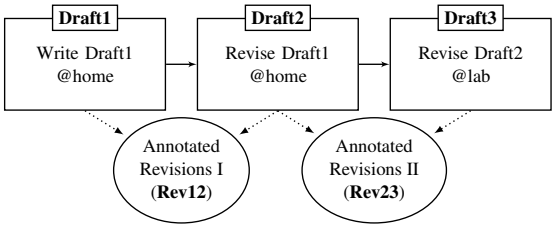


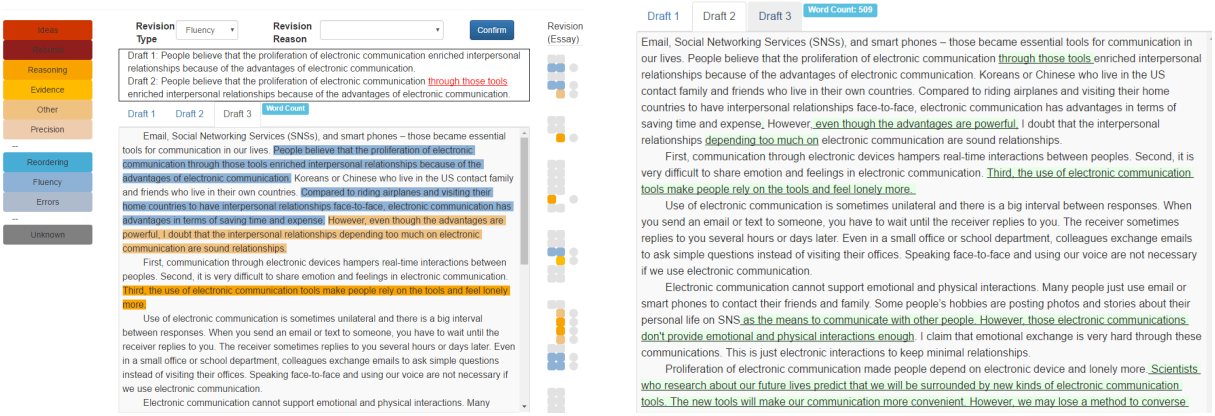
Figure 1: Our collected corpus contains five components: three drafts of an essay and two annotated revisions between drafts.

the raw text: argumentative essays with multiple drafts. We decided to sample from a population of predominantly college students, inclusive of both native and proficient non-native (aka L2) speakers. Comparing to high school students, college students are expected to produce essays having a better organization of the argument elements. Including native and L2 speakers allows for the exploration of possible rewriting differences between writers of varying backgrounds. We decided to give all subjects the same writing prompt and collect three drafts. The identical prompt minimizes the impact of topic difference for argumentation-related study. The collection of three drafts allows for a comparison of revision differences at different stages of rewriting.

Finally, we need a method for eliciting two revised drafts from each writer. Ideally, an instructor would give formative feedback after each draft for each student, but we do not have the resources to carry out such an expensive project. We simulate instructor feedback by asking students to add more examples after the first draft. To elicit a second revised draft, we use two different systems. First, we utilize an idealized[2] version of the ArgRewrite revision analysis system (Zhang et al., 2016). ArgRewrite highlights the locations of revisions at the sentence level and colors the revisions differently according to the revision purpose types. Our second system shows a character-based comparison between subsequent essay drafts[3]. This system is designed to have a similar look as ArgRewrite by highlighting the location of revisions. However, the type of revisions are not provided.

[2]All automatic revision feedback was manually examined/corrected to guarantee correctness.

[3]Code derived from https://code.google.com/p/google-diff-match-patch/ which implements Myers' algorithm (Myers, 1986).



(a) Interface A. (b) Interface B.

Figure 2: Screenshot of the interfaces. (a) *Interface A* with the annotated revision purposes, (b) *Interface B* with a streamlined character-based diff.

3 The ArgRewrite Corpus

Based on the above design decisions, we have developed a corpus of argumentative essays with three drafts and detailed annotations for sentence-aligned revisions between each consecutive pair of drafts. The main corpus has five elements, with the relationships between them shown in Figure 1; Section 3.1 describes the procedure for obtaining them. Section 3.2 briefly describes the revision schema we used and reports the inter-annotator agreement. Additionally, we have collected meta-data from the participants who contributed to the corpus (discussed in Section 3.3); these data may be useful for user behavior analysis.

3.1 Corpus Development Procedure

We have recruited 60 participants aged 18 years and older, among whom 40 were English native speakers and 20 were non-native speakers with sufficient English proficiency.[4] The study to collect the corpus is carried out in three 40-60 minute sessions over the duration of two weeks.

Draft1 Each participant begins by completing a pre-study questionnaire (Section 3.3) and writing a short essay online. Participants are instructed to keep the essay around 400 words, making a single main point with two supporting examples. They are given the following prompt:

> *"Suppose you've been asked to contribute a short op-ed piece for The New York Times. Argue whether the proliferation of electronic communications (e.g., email, text or other social media) enriches or hinders the development of interpersonal relationships."*

Draft2 A few days later, participants are asked to revise their first draft online based on the following feedback: *Strengthen the essay by adding one more example or reasoning for the claim; then add a rebuttal to an opposing idea; keep the essay at 400 words.* With this feedback we try to push participants to make revisions for later processing by the two interfaces used to create Draft3.

Annotated Revisions I (Rev12) The two drafts are semi-manually aligned at the sentence level.[5] Then, the purpose of each pair of sentence revision is manually coded by a trained annotator, following the annotation guideline (see Section 3.2).

Draft3 Participants write their third draft in a lab environment. This time, they are not given additional instructional feedback. Instead, participants are shown a computer interface that highlights the differences between their first and second drafts. They are asked to revise and create a third draft to improve the general quality of their essay. We experimented with two variations of revision elicitation. Chosen at random, half of the participants (10 L2 participants and 20 Native participants) are shown Interface A, the interface based on the ArgRewrite system (Zhang et al., 2016), which highlights the annotated differences between the drafts (Figure 2(a)); half of the participants are shown In-

[4] i.e., with a TOEFL score higher than 100.

[5] Sentences are first automatically aligned (Zhang and Litman, 2014), then manually corrected by human.

Draft1	Revision Purpose	Draft2	Revision Purpose	Draft3
This world has no restriction on who one can talk to.	Conventions/ Grammar/ Spelling	This world has no restrictions on whom one can talk to.		This world has no restrictions on whom one can talk to.
			Rebuttal/ Reservation	Unfortunately, the younger users of digital communication cannot be entirely protected from the rhetoric of any outsider.
			Warrant/ Reasoning/ Backing	Modern society is now faced with the issue of cyber bullying as a result.
The only aspects of communication that this new development improves are internet navigation and faux internet relatability.	Word-Usage/ Clarity	The only aspects of digital communication that this new development improves are internet navigation and faux internet relatability.	Word-Usage/ Clarity	The only aspects of digital communication that this new development improves are internet navigation and faux internet relationships.
	Claims/ Ideas	Being immersed in the sphere of new technologies can allow for complete isolation from the active, non-digital world.		Being immersed in the sphere of new technologies can allow for complete isolation from the active, non-digital world.

Table 1: Examples from the annotated corpus. The sentences were aligned across the drafts and the revision purposes were labeled on the aligned sentence pairs. From Draft1 to Draft2, there are two *Modify* revisions (*Spelling* and *Clarity*) and one *Add* revision. From Draft2 to Draft3, there are two *Add* revisions (*Rebuttal* and *Reasoning*) and one *Modify* revision (*Clarity*).

terface B, a streamlined character-based diff (Figure 2(b)). In Interface A, some purposes were renamed from the original annotation categories to help the participants better understand the system (as detailed in Table 2)[6]. Both interface groups are asked to read a tutorial about their respective interfaces before beginning to revise. Participants in group A are also asked to verify the manually annotated revision purposes between their first and second drafts. This information is collected to investigate the impact of the difference between the system's recognized and the participant's intended purpose. After completing the final revision, all participants are given a post-study survey about their experiences (Section 3.3). Additionally, participants in group A are asked to verify the automatically predicted revision purposes between their second and third drafts (Section 4.2).

Annotated Revisions II (Rev23) Regardless of which interface the participants used, the second and third draft are compared and annotated by the trained annotator in the same process as before.

[6]Figure 2(a) has two additional categories. Precision was intended to represent revisions that make a sentence more precise. Unknown was intended to represent revisions that cannot be categorized to existing categories. These two categories were not used during annotation as they were reported to be confusing in our pilot studies.

3.2 Revision Annotation Guidelines

Following our prior corpus annotations (Zhang and Litman, 2015), sentence revisions are first coarsely categorized as *Surface* or *Content* changes (Faigley and Witte, 1981), depending on whether any informational content was modified; within each coarse category, we distinguish between several finer categories based on the argumentative and discourse writing literature (Kneupper, 1978; Faigley and Witte, 1981; Burstein et al., 2003). Our adapted schema has three *Surface* categories (Organization, Word Usage/Clarity, and Conventions/Grammar/Spelling) and five *Content* categories (Claim/Ideas, Warrant/Reasoning/Backing, Rebuttal/Reservation, Evidence, and General Content Development). Table 1 shows example aligned sentences in three collected drafts and their annotated revision categories. The edit types of revisions (Add, Delete and Modify) are decided according to the alignment of sentences.

Two annotators (one is experienced, and the other is newly trained) participated in data annotation. The annotators first went through a "training" phase where both annotators annotated 5 files and discussed their disagreements to resolve misunderstandings. Then, both annotators separately annotated 10 new files and Kappa was calculated

Name in Schema	Name in System	Definition
Content	Content	revisions that changed the information of essay
Claims/Ideas	Ideas	revisions that aimed to change the thesis of essay
Warrant/Reasoning/Backing	Reasoning	revisions that aimed to change the reasoning of thesis
Rebuttal/Reservation	Rebuttal	revisions that aimed to change the rebuttal of thesis
Evidence	Evidence	revisions that aimed to change the evidence support for thesis
General Content	Other	other types of content revisions
Surface	Surface	revisions that did not change the information of essay
Organization	Reordering	revisions that switched the order of sentences
Word Usage/Clarity	Fluency	revisions that aimed to make the essay more fluent
Conventions/Grammar/Spelling	Errors	revisions that aimed to fix the spelling/grammar mistakes

Table 2: Definition of category names in Interface A.

L2 (20)	Draft1	Draft2	Draft3
Avg #Words	379.1	412.8	484.7
Avg #Sentences	18.6	20.2	23.7
Avg #Paragraphs	3.9	4.5	4.8
Native (40)	Draft1	Draft2	Draft3
Avg #Words	372.4	394.7	531.6
Avg #Sentences	18.8	20.4	25.8
Avg #Paragraphs	4.0	4.7	5.1

Table 3: Descriptive statistics of the ArgRewrite Corpus, including average number of words, sentences and paragraphs per essay draft.

on the annotation of these 10 new files. The Kappa on this held-out data is 0.84 on the two coarse categories of *Surface* vs. *Content* and 0.71 on the eight fine-grained categories that appear in Table 2. The disagreements between annotators were removed after discussion and the final labels were used as the gold standard annotation.

3.3 Meta-Data

In addition to the raw text and annotations, the corpus release includes participant meta-data from both a pre-study and a post-study survey.

Pre-Study Survey The pre-study survey asks for participants' demographic information as well as their self-reported writing background, such as participants' confidence in their writing ability, the number of drafts they typically make, etc. The questions are listed in Appendix A.

Post-Study Survey The post-study survey contains questions about the participants' in-lab revision experience, such as whether they found the computer interface helpful. All questions are answered on a scale of 1 to 5, ranging from "strongly disagree" to "strongly agree". Details of questions are shown in Appendix B.

3.4 Descriptive Statistics

Table 3 indicates the average number of words/sentences/paragraphs per essay draft. The corpus includes 180 essays: 120 (Draft1 and Draft2) with an average of about 400 words and 60 (Draft3) with an average of around 500 words.

Among the 40 native speakers, there were 29 (72.5%) undergraduates, 6 (15%) graduate students, and 5 (12.5%) non-students (post-docs and lecturers). Among the 20 L2 speakers, there were 4 (20%) undergraduates, and 16 (80%) graduate students; there were 9 Chinese, 2 Bengali, 2 Marathi, 2 Persian, 1 Arabic, 1 Korean, 1 Portuguese, 1 Spanish, and 1 Tamil. In terms of discipline, 33 participants (55%) were from the natural sciences, 24 (40%) from the social sciences, and 2 (3.3%) from the humanities. 1 participant (1.7%) did not reveal his/her discipline.

3.5 Public Release

The corpus is freely available for research usage[7]. The first release includes the raw text plus the revision annotations and the meta-data. The revision annotations are stored as .xlsx files. There are 60 spreadsheet files for revisions from Draft1 to Draft2 and 60 more spreadsheet files for revisions from Draft2 to Draft3. Each spreadsheet file contains two sheets: Old Draft and New Draft. Each row in the sheet represents one sentence in the corresponding draft. The index of the aligned sentence row in the other draft and the type of the revision on the sentence are recorded. The meta-data are in .log text files. Information in the text files are stored using the JSON data format.

4 Example Uses of the Corpus

While the development of a full fledged revision analysis system is outside the scope of this work, we demonstrate potential uses of our corpus with two examples. We first perform statistical analyses on the collected revision data and meta-data

[7] http://argrewrite.cs.pitt.edu

	Content		*Surface*	
	Rev12	Rev23	Rev12	Rev23
L2 (20)	172	78	163	176
Interface A	91	37	71	85
Interface B	81	41	92	91
Native (40)	334	285	303	246
Interface A	177	154	149	111
Interface B	157	131	154	135

Table 4: Number of revisions, by participant groups (language, interface), coarse-grain purposes, and revision drafts (Rev12 is between Draft1-Draft2; Rev23 is between Draft2-Draft3).

to understand aspects of participant behavior. We also use the corpus to train a supervised classifier to automatically predict revision purposes.

4.1 Student Revision Behavior Analysis

While it is well-established that thoughtful revisions improve one's writing, and while many college-level courses require students to submit multiple drafts of writing assignments (Addison and McGee, 2010), instructors rarely monitor and provide feedback to students while they revise. This is partly due to instructors' time constraints and partly due to their uncertainty about how to support students' revisions (Cole, 2014; Melzer, 2014). There is much we do not know about how to stimulate students to self-reflect and revise.

4.1.1 Hypotheses

Using the ArgRewrite Corpus, we can begin to ask and address some questions about revision habits and behaviors. Our first question is: How do different types of revision feedback impact student revision? And relatedly: Does student background (e.g., native vs. L2) make a difference? We thus mine the corpus to test the following hypotheses:

H1. There is a difference in participants' revising behaviors depending on which interface is used to elicit the third draft.

H2. For participants who used Interface A, if the recognized revision purpose differs from the participants' intended revision purpose, participants will further modify their revision.

H3. L2 and native speakers have different behaviors in making revisions.

H1 and **H2** address the first question; **H3** addresses the second.

4.1.2 Methodology

To test the hypotheses, we will use both subjective and objective measures. Subjective measures are based on participant post-study survey answers. Ideally, objective measures should be based on an assessment of improvements in the revised drafts; since we do not have evaluative data at this time, we approximate the degree of improvement using the number of revisions, since these two quantities were demonstrated to be positively correlated (Zhang and Litman, 2015). The objective measures are computed from Tables 4 and 5.

To compare differences between specific subgroups on the subjective and objective measures, we conduct ANOVA tests with two factors. There are multiple factors that can influence the users' rewriting behaviors such as the user's native language, education level and previous revision behaviors, etc. In our study, we try to explore the difference between interface groups considering one of the most salient confounding factors: language. We use one factor as the participant's native language (whether the participant is native or L2) and the other factor as the interface used. To determine correlation between quantitative measures, we conduct Spearman (ρ) and Pearson (r) correlation tests.

4.1.3 Results and Discussion

Testing for H1 Comparing Group A and Group B participants, we observe some differences. First, we detect that Group A agrees with the statement "The system helps me to recognize the weakness of my essay" more so than Group B (Group A has a mean rating of 3.97 ("Agree") while Group B's is 3.17 ("Neutral"), $p < .003$). Second, in Group A, there is a trending positive correlation between the number of revisions[8] from Draft2 to Draft3 and the ratings for the statement "The system encourages me to make more revisions than I usually make" (ρ=.33 and $p < .07$); whereas there is no such correlation for Group B. Additional information about revision purposes may elicit a stronger self-reflection response in Group A participants. In contrast, in Group B, there is a significant negative correlation between the number of Rev12 and ratings for the statement "it is convenient to view my previous revisions with the system" (ρ=-.36 and $p < .05$). This suggests that the character-based interface is ineffective when participants have to reflect on many changes.

[8]The results reported are the normalized numbers $\frac{\#revisions}{\#sentences}$, where #sentences represents the number of sentences in the draft before revision. The absolute numbers were also tested and similar findings were observed.

Revision Purpose	Draft1 to Draft2			Draft2 to Draft3			Totals
	#Add	#Delete	#Modify	#Add	#Delete	#Modify	
Content	294	179	33	320	27	16	869
Claims/Ideas	25	8	4	5	0	0	42
Warrant/Reasoning/Backing	166	83	7	191	13	3	463
Rebuttal/Reservation	23	1	0	13	0	0	37
General Content	50	80	18	86	13	13	260
Evidence	30	7	4	25	1	0	67
Surface	0	0	466	0	0	422	888
Word Usage/Clarity	0	0	362	0	0	357	719
Conventions/Grammar/Spelling	0	0	75	0	0	52	127
Organization	0	0	29	0	0	13	42

Table 5: Number of revisions, by fine-grain revision purposes and edit types (add, delete, modify).

On the other hand, when comparing the number of revisions made by Group A and Group B on Rev23 (controlling for their Rev12 numbers), we did not find a significant difference.

As we did not observe a significant difference in the number of revisions made by the two interface groups, we cannot verify that **H1** is true; possibly a larger pool of participants is needed, or possibly the writing assignment is not extensive enough (in length and in the number of drafts). Another possible explanation is that the system might only motivate the users to make more revisions when the feedback is different from the user's intention. To further verify the correctness of H1, we plan to have the essays graded by experts. The graded scores could allow us to analyze whether essays improved more when Interface A was used.

Testing for H2 Focusing on the 30 participants from Group A, we check the impact of the feedback regarding Rev12 on how they subsequently revise (Rev23). We counted the *Add* and *Modify* revisions where the participant disagrees with the revision purpose assigned by the annotator in Rev12. Of those, we then count the number of times the corresponding sentences were further revised[9]. Of the 53 sentences where the participants disagreed with the annotator, 45 were further revised in the third draft. The ratio is 0.849, much higher than the overall ratio of general Rev12 revisions being further revised in Rev23 (161/394 = 0.409) and the ratio of the agreed Rev12 revisions being revised in Rev23 (67/341 = 0.196). In further analysis, a Pearson correlation test is conducted to check the correlation between the number of Rev23 and the number of disagreements for different types of agreement/disagreements, controlling for the number of Rev12. We find a negative correlation between Rev23 and the number of cases (r=-0.41, $p < .03$) in which the revisions annotated as *Content* are verified by the participants; we also find a positive correlation between Rev23 and the number of cases (r=0.36, $p <= .05$) in which the revisions annotated as *Surface* are intended to be *Content* revisions by the participants. Both findings are consistent with **H2**, suggesting that participants will revise further if they perceive that their intended revisions were not recognized.

Testing for H3 We observe that native and L2 speakers exhibit different behaviors. First, we tested the difference in Content23 and Surface23[10] between these speaker groups with ANOVA. We observe significant difference in the number of content ($p < .02$) and surface ($p < .03$) revisions made by L2 and native speakers. More specifically, our native speakers make more *Content* changes while the L2 speakers make more *Surface* changes. Second, with ANOVA we found a significant interaction effect of the two factors (Group and users' L2 or native status) ($p < .021$) on their ratings for the statement "the system helps me to recognize the weakness of my essay" with L2 speakers having a stronger Interface A preference. Third, we observe a significant positive correlation in the native group between the number of content revisions in Rev23 and the ratings of the statement "the system encourages me to make more revisions than I usually make" (ρ=.4 and $p < .009$). This suggests that giving feedback (from either interface) encourages native speakers to make more content revisions. Finally, in the L2 group, there is a significant negative correlation between the number of surface revisions in Rev12 and the ratings for the statement "the system helps me to recognize the weakness of my es-

[9] *Delete* revisions were ignored as the deleted sentences are not traceable in Draft3

[10] content/surface revisions from Draft2 to Draft3

say" (ρ=-.57 and $p < .008$). This shows that giving feedback to L2 speakers is less helpful when they make more surface revisions. These results are consistent with **H3**.

Summary Our findings suggest that feedback on revisions do impact how students review and rewrite their drafts. However, there are many factors at play, including the interface design and the students' linguistic backgrounds.

4.2 Automatic Revision Identification

Another use of the corpus is to serve as a gold standard for training and testing a revision purpose prediction component for use in an automatic revision analysis system. In the version of ArgRewrite evaluated earlier (Interface A), the manual annotation of revision purposes enabled the system to provide revision feedback to users, which motivated them to improve their writing (**H2**). Automatic argumentative revision purpose prediction has been previously investigated by Zhang and Litman (2015). They developed and reported the performance of a binary classifier for each individual revision category (1 for revisions of the category and 0 for the rest of all revisions) using features from prior research. The availability of our corpus makes it possible for researchers to replicate such methods and conduct further studies.

4.2.1 Hypotheses

In this paper, we repeat the experiment of Zhang and Litman (2015) under different settings to investigate three new hypotheses that can now be investigated given the features of our corpus:

H4. The method used in Zhang and Litman (2015) for high school writings is also useful for the writings of college students.

H5. The same revision classification method works differently for first revision attempts and second revision attempts.

H6. The revision classification model trained on L2 essays has a different preference from the model trained on native essays.

4.2.2 Methodology

We followed the work of (Zhang and Litman, 2015), where unigram features (words) were used as the baseline and the SVM classifier was used. Besides unigrams, three groups of features used in revision analysis, argument mining and discourse analysis research were extracted (*Location*, *Textual* and *Language*) as in Table 6 (Bronner and Monz, 2012; Daxenberger and Gurevych, 2013; Burstein et al., 2001; Falakmasir et al., 2014).

For **H4**, 10-fold (participant) cross-validation is conducted on all the essays in the corpus. Unweighted average F-score for each revision category is reported, using unigram features versus using all features. Zhang and Litman (2015) observed a significant improvement over the unigram baseline using all the features. If **H4** is true, we should expect a similar improvement over the unigram baseline using our corpus.

For **H5**, 10-fold cross-validation was conducted for the revisions from Draft1 to Draft2 and revisions from Draft2 to Draft3 separately. We compared the improvement ratio brought by the advanced features over the unigram baseline.

For **H6**, we trained two classifiers separately with L2 and native essays with all the features. 20 native participants were first randomly selected as the test data. Afterwards classifiers were trained separately using the 20 L2 participants' essays and the remaining 20 native participants' essays. We would expect that the performance of the two trained classifiers is different on the same test data.

4.2.3 Results and Discussion

The first two rows of Table 7 support **H4**. We observe that the method (SVM + all features) used in Zhang and Litman (2015) significantly improves performance (compared to a unigram baseline) for half of the classification tasks, which is similar to Zhang and Litman's results on high school (primarily L1) writing. In our corpus, performance on *Claim*, *Evidence*, *Rebuttal* and *Organization* was not significantly better than the baseline, possibly due to the limited number of positive training samples for these categories (Table 5). For example, one reason that the performance in Table 7 for *Evidence* might be low is that there are less than 100 Evidence instances in Table 5.

For **H5**, the four rows in the middle of Table 7 show the difference of the cross-validation results on first attempt revisions and second attempt revisions. The earlier results using all the revisions, versus now just using only Rev12 or Rev23 revisions are similar, which provides little support for **H5**. With one exception, the features proposed in Zhang and Litman (2015) could again significantly improve the performance over the unigram baseline, for the same set of categories as when using all the revisions. However, for the *Conventions/Grammar/Spelling* category, we did not ob-

Group	Illustration
Location	The location of revised sentences in the paragraph/essay (e.g., whether the sentence is the first or last sentence of the paragraph/essay, the index of the sentence in the paragraph)
Textual	The textual features of revised sentences (e.g., whether the sentence contains a named entity, certain discourse markers ("because", "due to", etc), sentence difference (edit distance, difference in punctuations, etc.) and edit types (Add, Delete or Modify))
Language	The language features of revised sentences (e.g., difference in POS tags, spelling/grammar mistakes)

Table 6: Illustration of features used in the revision classification study.

Experiments	Text-based					Surface		
	Claim	Warrant	General	Evidence	Rebuttal	Org.	Word	Conv
10fold + All Revs + Unigram	0.49	0.58	0.48	0.49	0.49	0.49	0.73	0.49
10fold + All Revs + All features	0.49	0.77∗	0.55∗	0.50	0.49	0.49	0.86∗	0.62∗
10fold + Rev12 + Unigram	0.50	0.58	0.47	0.50	0.50	0.50	0.57	0.62
10fold + Rev12 + All features	0.50	0.77∗	0.56∗	0.50	0.50	0.50	0.72∗	0.72∗
10fold + Rev23 + Unigram	0.50	0.46	0.53	0.49	0.50	0.50	0.58	0.46
10fold + Rev23 + All features	0.50	0.60∗	0.65∗	0.49	0.50	0.50	0.78∗	0.50
20 L2 (train) + 20 Native (test)	0.50	0.72	0.48	0.49	0.50	0.50	0.83	**0.63**
20 Native (train) + 20 Native (test)	0.50	**0.76**	**0.52**	0.49	0.50	0.50	**0.89**	0.54

Table 7: Average unweighted F-score for each binary classification task. The first 6 rows show the average value of 10-fold cross-validation. $*$ indicates significantly better than unigram baseline ($p < .05$). The last 2 rows show the F-value for training on L2/Native data and testing on Native data. **Bold** indicates larger than the number in the other row.

serve a significant improvement for revisions from Draft2 and Draft3. A possible explanation is that there is a bigger difference in the writers' rewriting behavior from Draft2 to Draft3, which increases the difficulty of prediction.

The last two rows of Table 7 support **H6**. Interestingly, we observe a better performance on *Warrant*, *General* and *Word Usage/Clarity* with a classifier trained and tested using native essays. Perhaps essays of native speakers are more similar to each other when revised along these dimensions. For *Conventions/Grammar/Spelling*, in contrast, the classifier trained on L2 data yields a better performance on the same native test set. This may be because the L2 revisions usually include more spelling/grammar corrections.

5 Conclusion and Future Work

We have presented a new corpus for writing comparison research. Currently the corpus focuses on essay revisions made by both native and L2 college students. In addition to three drafts of essays, we have analyzed the drafts to align semantically similar sentences and to assign revision purposes for each revised aligned sentence pair. We have also conducted two studies to demonstrate the use of the corpus for revision behavior analysis and for automatic revision purpose classification.

While in this paper we explored language as one factor influencing rewriting behavior, our corpus also contains information about other potential factors such as gender and education level which we plan to investigate in the future. We also plan to augment the corpus to support additional types of research on revision analysis. Some potential augmentations include more fine-grained revision categories, revision properties such as statement strength (Tan and Lee, 2014) and quality evaluations, and sub-sentential revision scopes.

Acknowledgments

We want to thank Amanda Godley, Geeta Kothari, and the members of the ArgRewrite group (Reed Armstrong, Nicolo Manfredi and Tazin Afrin) for their helpful feedback and the anonymous reviewers for their suggestions. We also want to thank Adam Hobaugh, Dennis Wakefield and Anthony M Taliani for their assistance in the set up of study environments. This material is based upon work supported by the National Science Foundation under Grant No. 1550635. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This research is also funded by the Learning Research and Development Center of the University of Pittsburgh.

References

Joanne Addison and Sharon James McGee. 2010. Writing in high school/writing in college: Research trends and future directions. *College Composition and Communication* pages 147–179.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment* 4(3).

Amit Bronner and Christof Monz. 2012. User edits classification using document revision histories. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. pages 356–366.

Jill Burstein and Martin Chodorow. 1999. Automated essay scoring for nonnative English speakers. In *Proceedings of a Symposium on Computer Mediated Language Assessment and Evaluation in Natural Language Processing*. pages 68–75.

Jill Burstein, Daniel Marcu, Slava Andreyev, and Martin Chodorow. 2001. Towards automatic classification of discourse elements in essays. In *Proceedings of the 39th annual Meeting on Association for Computational Linguistics*. pages 98–105.

Jill Burstein, Daniel Marcu, and Kevin Knight. 2003. Finding the WRITE stuff: Automatic identification of discourse structure in student essays. *IEEE Intelligent Systems* 18(1):32–39.

Daniel Cole. 2014. What if the earth is flat? working with, not against, faculty concerns about grammar in student writing. *The WAC Journal* 25:7–35.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. pages 22–31.

Johannes Daxenberger and Iryna Gurevych. 2012. A corpus-based study of edit categories in featured and non-featured Wikipedia articles. In *COLING*. pages 711–726.

Johannes Daxenberger and Iryna Gurevych. 2013. Automatically classifying edit categories in Wikipedia revisions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 578–589.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*.

Lester Faigley and Stephen Witte. 1981. Analyzing revision. *College composition and communication* pages 400–414.

Mohammad Hassan Falakmasir, Kevin D Ashley, Christian D Schunn, and Diane J Litman. 2014. Identifying thesis and conclusion statements in student essays to scaffold peer review. In *International Conference on Intelligent Tutoring Systems*. pages 254–259.

Lucie Flekova, Daniel Preoţiuc-Pietro, and Lyle Ungar. 2016. Exploring stylistic variation with age and income on Twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Short Papers)*. pages 313–319.

Grammarly. 2016. Grammarly. http://www.grammarly.com. [Online; accessed 04-10-2017].

John Jones. 2008. Patterns of revision in online writing a study of Wikipedia's featured articles. *Written Communication* 25(2):262–289.

Charles W Kneupper. 1978. Teaching argument: An introduction to the Toulmin model. *College Composition and Communication* 29(3):237–241.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. Automated grammatical error detection for language learners. *Synthesis lectures on human language technologies* 3(1):1–134.

Jun Liu and Sudha Ram. 2009. Who does what: Collaboration patterns in the Wikipedia and their impact on data quality. In *19th Workshop on Information Technologies and Systems*. pages 175–180.

Prodromos Malakasiotis and Ion Androutsopoulos. 2011. A generate and rank approach to sentence paraphrasing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 96–106.

Dan Melzer. 2014. The connected curriculum: Designing a vertical transfer writing curriculum. *The WAC Journal* 25:78–91.

Eugene W Myers. 1986. An O(ND) difference algorithm and its variations. *Algorithmica* 1(1-4):251–266.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *Proceedings of NAACL-HLT*. pages 1384–1394.

Ulrike Pfeil, Panayiotis Zaphiris, and Chee Siang Ang. 2006. Cultural differences in collaborative authoring of Wikipedia. *Journal of Computer-Mediated Communication* 12(1):88–113.

Marius Popescu and Liviu P. Dinu. 2008. Rank distance as a stylistic similarity. In *Coling 2008*. pages 91–94.

Ben Swanson and Elif Yamangil. 2012. Correction detection and error type selection as an ESL educational aid. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 357–361.

Chenhao Tan and Lillian Lee. 2014. A corpus of sentence-level revisions in academic writing: A step towards understanding statement strength in communication. In *Proceedings of ACL (short paper)*.

Huichao Xue and Rebecca Hwa. 2014. Improved correction detection in revised ESL sentences. In *ACL (2)*. pages 599–604.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. pages 180–189.

Fan Zhang, Rebecca Hwa, Diane Litman, and Homa B Hashemi. 2016. Argrewrite: A web-based revision assistant for argumentative writings. *NAACL HLT 2016* page 37.

Fan Zhang and Diane Litman. 2014. Sentence-level rewriting detection. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications*. pages 149–154.

Fan Zhang and Diane Litman. 2015. Annotation and classification of argumentative writing revisions. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*. pages 133–143.

A Questions of the pre-study survey

1. Is English your native language?
2. (only L2 participants) What is your native language?
3. What is your major? Please select the closest discipline to your major.
    - Natural sciences
    - Social sciences
    - Humanities
4. Are you an undergraduate or graduate student?
5. What is your current year of study?
6. When writing a paper for a class, how many drafts of major revisions do you typically make?
7. Overall, how confident are you with your writings? (Not at all confident, Not very confident, Somewhat confident, confident, Extremely confident)
8. (only L2 participants) Please tell us how comfortable you feel about writing in the English language versus writing in your primary language. (Not at all comfortable, Not very comfortable, Somewhat comfortable, comfortable, Extremely comfortable)
9. What are some of your recent classes that have an intensive writing component to them? How did you do in these classes?
10. What aspects of writing do you think you are good at? e.g. vocabulary choice, clear sentences, writing organization.
11. What aspects of writing do you think you can improve?

B Questions of the post-study survey

1. The system allows me to have a better understanding of my previous revision efforts.
2. It is convenient to view my previous revisions with the system.
3. The system helps me to recognize the weakness of my essay.
4. The system encourages me to make more revisions than I usually make.
5. The system encourages me to think more about making more meaningful changes.
6. Overall the system is helpful to my writing.

Watset: Automatic Induction of Synsets from a Graph of Synonyms

Dmitry Ustalov[†*], **Alexander Panchenko**[‡], **and Chris Biemann**[‡]

[†]Institute of Natural Sciences and Mathematics, Ural Federal University, Russia
[*]Krasovskii Institute of Mathematics and Mechanics, Russia
[‡]Language Technology Group, Department of Informatics, Universität Hamburg, Germany
dmitry.ustalov@urfu.ru
{panchenko,biemann}@informatik.uni-hamburg.de

Abstract

This paper presents a new graph-based approach that induces synsets using synonymy dictionaries and word embeddings. First, we build a weighted graph of synonyms extracted from commonly available resources, such as Wiktionary. Second, we apply word sense induction to deal with ambiguous words. Finally, we cluster the disambiguated version of the ambiguous input graph into synsets. Our meta-clustering approach lets us use an efficient hard clustering algorithm to perform a fuzzy clustering of the graph. Despite its simplicity, our approach shows excellent results, outperforming five competitive state-of-the-art methods in terms of F-score on three gold standard datasets for English and Russian derived from large-scale manually constructed lexical resources.

1 Introduction

A *synset* is a set of mutual synonyms, which can be represented as a clique graph where nodes are words and edges are synonymy relations. Synsets represent word senses and are building blocks of WordNet (Miller, 1995) and similar resources such as thesauri and lexical ontologies. These resources are crucial for many natural language processing applications that require common sense reasoning, such as information retrieval (Gong et al., 2005) and question answering (Kwok et al., 2001; Zhou et al., 2013). However, for most languages, no manually-constructed resource is available that is comparable to the English WordNet in terms of coverage and quality. For instance, Kiselev et al. (2015) present a comparative analysis of lexical resources available for the Russian language concluding that there is no resource compared to WordNet in terms of coverage and quality for Russian. This lack of linguistic resources for many languages urges the development of new methods for automatic construction of WordNet-like resources. The automatic methods foster construction and use of the new lexical resources.

Wikipedia[1], Wiktionary[2], OmegaWiki[3] and other collaboratively-created resources contain a large amount of lexical semantic information—yet designed to be human-readable and not formally structured. While semantic relations can be automatically extracted using tools such as DKPro JWKTL[4] and Wikokit[5], words in these relations are not disambiguated. For instance, the synonymy pairs (*bank*, *streambank*) and (*bank*, *banking company*) will be connected via the word "bank", while they refer to the different senses. This problem stems from the fact that articles in Wiktionary and similar resources list undisambiguated synonyms. They are easy to disambiguate for humans while reading a dictionary article, but can be a source of errors for language processing systems.

The contribution of this paper is a novel approach that resolves ambiguities in the input graph to perform fuzzy clustering. The method takes as an input synonymy relations between potentially ambiguous terms available in human-readable dictionaries and transforms them into a machine readable representation in the form of disambiguated synsets. Our method, called WATSET, is based on a new local-global meta-algorithm for fuzzy graph clustering. The underlying principle is to discover the word senses based on a *local* graph cluster-

[1] http://www.wikipedia.org
[2] http://www.wiktionary.org
[3] http://www.omegawiki.org
[4] https://dkpro.github.io/dkpro-jwktl
[5] https://github.com/componavt/wikokit

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1579–1590
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1145

ing, and then to induce synsets using *global* sense clustering. We show that our method outperforms other methods for synset induction. The induced resource eliminates the need in manual synset construction and can be used to build WordNet-like semantic networks for under-resourced languages. An implementation of our method along with induced lexical resources is available online.[6]

2 Related Work

Methods based on resource linking surveyed by Gurevych et al. (2016) gather various existing lexical resources and perform their linking to obtain a machine-readable repository of lexical semantic knowledge. For instance, BabelNet (Navigli and Ponzetto, 2012) relies in its core on a linking of WordNet and Wikipedia. UBY (Gurevych et al., 2012) is a general-purpose specification for the representation of lexical-semantic resources and links between them. The main advantage of our approach compared to the lexical resources is that no manual synset encoding is required.

Methods based on word sense induction try to induce sense representations without the need for any initial lexical resource by extracting semantic relations from text. In particular, word sense induction (WSI) based on word ego networks clusters graphs of semantically related words (Lin, 1998; Pantel and Lin, 2002; Dorow and Widdows, 2003; Véronis, 2004; Hope and Keller, 2013; Pelevina et al., 2016; Panchenko et al., 2017a), where each cluster corresponds to a word sense. An ego network consists of a single node (ego) together with the nodes they are connected to (alters) and all the edges among those alters (Everett and Borgatti, 2005). In the case of WSI, such a network is a local neighborhood of one word. Nodes of the ego network are the words which are semantically similar to the target word.

Such approaches are able to discover homonymous senses of words, e.g., "bank" as slope versus "bank" as organisation (Di Marco and Navigli, 2012). However, as the graphs are usually composed of semantically related words obtained using distributional methods (Baroni and Lenci, 2010; Biemann and Riedl, 2013), the resulting clusters by no means can be considered synsets. Namely, (1) they contain words related not only via synonymy relation, but via a mixture of relations such as synonymy, hypernymy, co-hyponymy, antonymy, etc. (Heylen et al., 2008; Panchenko, 2011); (2) clusters are not unique, i.e., one word can occur in clusters of different ego networks referring to the same sense, while in WordNet a word sense occurs only in a single synset.

In our synset induction method, we use word ego network clustering similarly as in word sense induction approaches, but apply them to a graph of semantically clean synonyms.

Methods based on clustering of synonyms, such as our approach, induce the resource from an ambiguous graph of synonyms where edges a extracted from manually-created resources. According to the best of our knowledge, most experiments either employed graph-based word sense induction applied to text-derived graphs or relied on a linking-based method that already assumes availability of a WordNet-like resource. A notable exception is the ECO approach by Gonçalo Oliveira and Gomes (2014), which was applied to induce a WordNet of the Portuguese language called Onto.PT.[7] We compare to this approach and to five other state-of-the-art graph clustering algorithms as the baselines.

ECO (Gonçalo Oliveira and Gomes, 2014) is a *fuzzy* clustering algorithm that was used to induce synsets for a Portuguese WordNet from several available synonymy dictionaries. The algorithm starts by adding random noise to edge weights. Then, the approach applies Markov Clustering (see below) of this graph several times to estimate the probability of each word pair being in the same synset. Finally, candidate pairs over a certain threshold are added to output synsets.

MaxMax (Hope and Keller, 2013) is a *fuzzy* clustering algorithm particularly designed for the word sense induction task. In a nutshell, pairs of nodes are grouped if they have a maximal mutual affinity. The algorithm starts by converting the undirected input graph into a directed graph by keeping the maximal affinity nodes of each node. Next, all nodes are marked as root nodes. Finally, for each root node, the following procedure is repeated: all transitive children of this root form a cluster and the root are marked as non-root nodes; a root node together with all its transitive children form a fuzzy cluster.

Markov Clustering (MCL) (van Dongen, 2000) is a *hard* clustering algorithm for graphs based on simulation of stochastic flow in graphs.

[6] https://github.com/dustalov/watset

[7] http://ontopt.dei.uc.pt

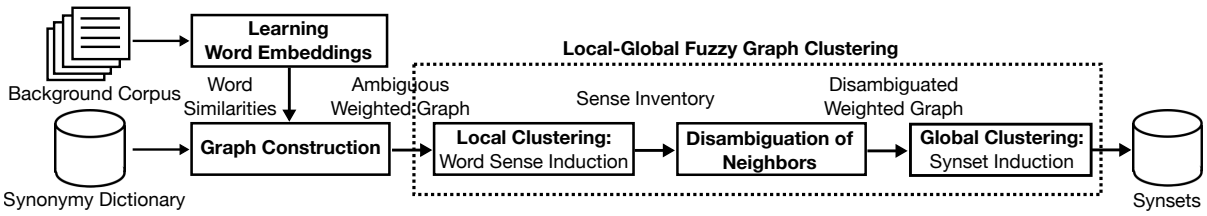


Figure 1: Outline of the WATSET method for synset induction.

MCL simulates random walks within a graph by alternation of two operators called expansion and inflation, which recompute the class labels. Notably, it has been successfully used for the word sense induction task (Dorow and Widdows, 2003).

Chinese Whispers (CW) (Biemann, 2006) is a *hard* clustering algorithm for weighted graphs that can be considered as a special case of MCL with a simplified class update step. At each iteration, the labels of all the nodes are updated according to the majority labels among the neighboring nodes. The algorithm has a meta-parameter that controls graph weights that can be set to three values: (1) *top* sums over the neighborhood's classes; (2) *nolog* downgrades the influence of a neighboring node by its degree or by (3) *log* of its degree.

Clique Percolation Method (CPM) (Palla et al., 2005) is a *fuzzy* clustering algorithm for unweighted graphs that builds up clusters from k-cliques corresponding to fully connected subgraphs of k nodes. While this method is only commonly used in social network analysis, we decided to add it to the comparison as synsets are essentially cliques of synonyms, which makes it natural to apply an algorithm based on clique detection.

3 The WATSET Method

The goal of our method is to induce a set of unambiguous synsets by grouping individual ambiguous synonyms. An outline of the proposed approach is depicted in Figure 1. The method takes a dictionary of ambiguous synonymy relations and a text corpus as an input and outputs synsets. Note that the method can be used without a background corpus, yet as our experiments will show, corpus-based information improves the results when utilizing it for weighting the word graph's edges.

A synonymy dictionary can be perceived as a graph, where the nodes correspond to lexical entries (words) and the edges connect pairs of the nodes when the synonymy relation between them holds. The cliques in such a graph naturally form densely connected sets of synonyms corresponding to concepts (Gfeller et al., 2005). Given the fact that solving the clique problem exactly in a graph is NP-complete (Bomze et al., 1999) and that these graphs typically contain tens of thousands of nodes, it is reasonable to use efficient hard graph clustering algorithms, like MCL and CW, for finding a global segmentation of the graph. However, the hard clustering property of these algorithm does not handle polysemy: while one word could have several senses, it will be assigned to only one cluster. To deal with this limitation, a word sense induction procedure is used to induce senses for all words, one at the time, to produce a disambiguated version of the graph where a word is now represented with one or many word senses. The concept of a disambiguated graph is described in (Biemann, 2012). Finally, the disambiguated word sense graph is clustered globally to induce synsets, which are hard clusters of word senses.

More specifically, the method consists of five steps presented in Figure 1: (1) learning word embeddings; (2) constructing the ambiguous weighted graph of synonyms G; (3) inducing the word senses; (4) constructing the disambiguated weighted graph G' by disambiguating of neighbors with respect to the induced word senses; (5) global clustering of the graph G'.

3.1 Learning Word Embeddings

Since different graph clustering algorithms are sensitive to edge weighting, we consider distributional semantic similarity based on word embeddings as a possible edge weighting approach for our synonymy graph. As we show further, this approach improves over unweighted versions and yields the best overall results.

3.2 Construction of a Synonymy Graph

We construct the synonymy graph $G = (V, E)$ as follows. The set of nodes V includes every lexeme appearing in the input synonymy dictionaries. The set of undirected edges E is composed of all edges

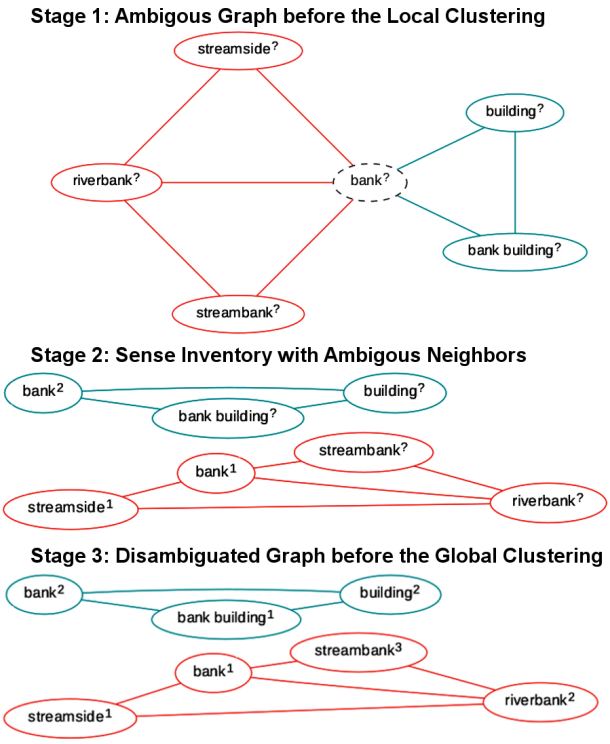


Figure 2: Disambiguation of an ambiguous input graph using local clustering (WSI) to facilitate global clustering of words into synsets.

$(u, v) \in V \times V$ retrieved from one of the input synonymy dictionaries. We consider three edge weight representations:

- **ones** that assigns every edge the constant weight of 1;
- **count** that weights the edge (u, v) as the number of times the synonymy pair appeared in the input dictionaries;
- **sim** that assigns every edge (u, v) a weight equal to the cosine similarity of skip-gram word vectors (Mikolov et al., 2013).

As the graph G is likely to have polysemous words, the goal is to separate individual word senses using graph-based word sense induction.

3.3 Local Clustering: Word Sense Induction

In order to facilitate global fuzzy clustering of the graph, we perform disambiguation of its ambiguous nodes as illustrated in Figure 2. First, we use a graph-based word sense induction method that is similar to the curvature-based approach of Dorow and Widdows (2003). In particular, removal of the nodes participating in many triangles tends to separate the original graph into several connected components. Thus, given a word u, we extract a network of its nearest neighbors from the synonymy graph G. Then, we remove the original word u from this network and run a hard graph clustering algorithm that assigns one node to one and only one cluster. In our experiments, we test Chinese Whispers and Markov Clustering. The expected result of this is that each cluster represents a different sense of the word u, e.g.:

$bank^1$ $\{streambank, riverbank, \dots\}$
$bank^2$ $\{bank\ company, \dots\}$
$bank^3$ $\{bank\ building, building, \dots\}$
$bank^4$ $\{coin\ bank, penny\ bank, \dots\}$

We denote, e.g., bank^1, bank^2 and other items as word senses referred to as $\text{senses}(\text{bank})$. We denote as $\text{ctx}(s)$ a cluster corresponding to the word sense s. Note that the context words have no sense labels. They are recovered by the disambiguation approach described next.

3.4 Disambiguation of Neighbors

Next, we disambiguate the neighbors of each induced sense. The previous step results in splitting word nodes into (one or more) sense nodes. However, nearest neighbors of each sense node are still ambiguous, e.g., ($bank^3$, $building^?$). To recover these sense labels of the neighboring words, we employ the following sense disambiguation approach proposed by Faralli et al. (2016). For each word u in the context $\text{ctx}(s)$ of the sense s, we find the most similar sense of that word $\hat{u}$ to the context. We use the cosine similarity measure between the context of the sense s and the context of each candidate sense u' of the word u:

$$\hat{u} = \arg\max_{u' \in \text{senses}(u)} \cos(\text{ctx}(s), \text{ctx}(u')).$$

A context $\text{ctx}(\cdot)$ is represented by a sparse vector in a vector space of all ambiguous words of all contexts. The result is a disambiguated context $\widehat{\text{ctx}}(s)$ in a space of *disambiguated* words derived from its *ambiguous* version $\text{ctx}(s)$:

$$\widehat{\text{ctx}}(s) = \{\hat{u} : u \in \text{ctx}(s)\}.$$

3.5 Global Clustering: Synset Induction

Finally, we construct the word sense graph $G' = (V', E')$ using the disambiguated senses instead of the original words and establishing the edges between these disambiguated senses:

$$V' = \bigcup_{u \in V} \text{senses}(u), \quad E' = \bigcup_{s \in V'} \{s\} \times \widehat{\text{ctx}}(s).$$

Running a hard clustering algorithm on G' produces the desired set of synsets as our final result. Figure 2 illustrates the process of disambiguation of an input ambiguous graph on the example of the word "bank". As one may observe, disambiguation of the nearest neighbors is a necessity to be able to construct a global version of the sense-aware graph. Note that current approaches to WSI, e.g., (Véronis, 2004; Biemann, 2006; Hope and Keller, 2013), do not perform this step, but perform only local clustering of the graph since they do not aim at a global representation of synsets.

3.6 Local-Global Fuzzy Graph Clustering

While we use our approach to synset induction in this work, the core of our method is the "local-global" fuzzy graph clustering algorithm, which can be applied to arbitrary graphs (see Figure 1). This method, summarized in Algorithm 1, takes an undirected graph $G = (V, E)$ as the input and outputs a set of fuzzy clusters of its nodes V. This is a meta-algorithm as it operates on top of two hard clustering algorithms denoted as $\text{Cluster}_{\text{local}}$ and $\text{Cluster}_{\text{global}}$, such as CW or MCL. At the first phase of the algorithm, for each node its senses are induced via ego network clustering (lines 1–7). Next, the disambiguation of each ego network is performed (lines 8–15). Finally, the fuzzy clusters are obtained by applying the hard clustering algorithm to the disambiguated graph (line 16). As a post-processing step, the sense labels can be removed to make the cluster elements subsets of V.

Algorithm 1 WATSET fuzzy graph clustering

Input: a set of nodes V and a set of edges E.
Output: a set of fuzzy clusters of V.

1: **for all** $u \in V$ **do**
2: $\quad C \leftarrow \text{Cluster}_{\text{local}}(\text{Ego}(u))$ // $C = \{C_1, ...\}$
3: $\quad$ **for** $i \leftarrow 1 \dots |C|$ **do**
4: $\quad\quad \text{ctx}(u^i) \leftarrow C_i$
5: $\quad\quad \text{senses}(u) \leftarrow \text{senses}(u) \cup \{u^i\}$
6: $\quad$ **end for**
7: **end for**
8: $V' \leftarrow \bigcup_{u \in V} \text{senses}(u)$
9: **for all** $s \in V'$ **do**
10: $\quad$ **for all** $u \in \text{ctx}(s)$ **do**
11: $\quad\quad \hat{u} \leftarrow \underset{u' \in \text{senses}(u)}{\arg\max} \cos(\text{ctx}(s), \text{ctx}(u'))$
12: $\quad$ **end for**
13: $\quad \widehat{\text{ctx}}(s) \leftarrow \{\hat{u} : u \in \text{ctx}(s)\}$
14: **end for**
15: $E' \leftarrow \bigcup_{s \in V'} \{s\} \times \widehat{\text{ctx}}(s)$
16: **return** $\text{Cluster}_{\text{global}}(V', E')$

4 Evaluation

We conduct our experiments on resources from two different languages. We evaluate our approach on two datasets for English to demonstrate its performance on a resource-rich language. Additionally, we evaluate it on two Russian datasets since Russian is a good example of an under-resourced language with a clear need for synset induction.

4.1 Gold Standard Datasets

For each language, we used two differently constructed lexical semantic resources listed in Table 1 to obtain gold standard synsets.

English. We use **WordNet**[8], a popular English lexical database constructed by expert lexicographers. WordNet contains general vocabulary and appears to be *de facto* gold standard in similar tasks (Hope and Keller, 2013). We used WordNet 3.1 to derive the synonymy pairs from synsets. Additionally, we use **BabelNet**[9], a large-scale multilingual semantic network constructed automatically using WordNet, Wikipedia and other resources. We retrieved all the synonymy pairs from the BabelNet 3.7 synsets marked as English.

Russian. As a lexical ontology for Russian, we use **RuWordNet**[10] (Loukachevitch et al., 2016), containing both general vocabulary and domain-specific synsets related to sport, finance, economics, etc. Up to a half of the words in this resource are multi-word expressions (Kiselev et al., 2015), which is due to the coverage of domain-specific vocabulary. RuWordNet is a WordNet-like version of the RuThes thesaurus that is constructed in the traditional way, namely by a small group of expert lexicographers (Loukachevitch, 2011). In addition, we use **Yet Another RussNet**[11] (**YARN**) by Braslavski et al. (2016) as another gold standard for Russian. The resource is constructed using crowdsourcing and mostly covers general vocabulary. Particularly, non-expert users are allowed to edit synsets in a collaborative way loosely supervised by a team of project curators. Due to the ongoing development of the re-

[8] https://wordnet.princeton.edu

[9] http://www.babelnet.org

[10] http://ruwordnet.ru/en

[11] https://russianword.net/en

source, we selected as the gold standard only those synsets that were edited at least eight times in order to filter out noisy incomplete synsets.

Resource		# words	# synsets	# synonyms
WordNet	En	148 730	117 659	152 254
BabelNet	En	11 710 137	6 667 855	28 822 400
RuWordNet	Ru	110 242	49 492	278 381
YARN	Ru	9 141	2 210	48 291

Table 1: Statistics of the gold standard datasets.

4.2 Evaluation Metrics

To evaluate the quality of the induced synsets, we transformed them into binary synonymy relations and computed precision, recall, and F-score on the basis of the overlap of these binary relations with the binary relations from the gold standard datasets. Given a synset containing n words, we generate a set of $\frac{n(n-1)}{2}$ pairs of synonyms. The F-score calculated this way is known as *Paired F-score* (Manandhar et al., 2010; Hope and Keller, 2013). The advantage of this measure compared to other cluster evaluation measures, such as *Fuzzy B-Cubed* (Jurgens and Klapaftis, 2013), is its straightforward interpretability.

4.3 Word Embeddings

English. We use the standard 300-dimensional word embeddings trained on the 100 billion tokens Google News corpus (Mikolov et al., 2013).[12]

Russian. We use the 500-dimensional word embeddings trained using the skip-gram model with negative sampling (Mikolov et al., 2013) using a context window size of 10 with the minimal word frequency of 5 on a 12.9 billion tokens corpus of books. These embeddings were shown to produce state-of-the-art results in the RUSSE shared task[13] and are part of the Russian Distributional Thesaurus (RDT) (Panchenko et al., 2017b).[14]

4.4 Input Dictionary of Synonyms

For each language, we constructed a synonymy graph using openly available language resources. The statistics of the graphs used as the input in the further experiments are shown in Table 2.

English. Synonyms were extracted from the English Wiktionary[15], which is the largest Wiktionary at the present moment in terms of the lexical coverage, using the DKPro JWKTL tool by Zesch et al. (2008). English words have been extracted from the dump.

Russian. Synonyms from three sources were combined to improve lexical coverage of the input dictionary and to enforce confidence in jointly observed synonyms: (1) synonyms listed in the Russian Wiktionary extracted using the Wikokit tool by Krizhanovsky and Smirnov (2013); (2) the dictionary of Abramov (1999); and (3) the Universal Dictionary of Concepts (Dikonov, 2013). While the two latter resources are specific to Russian, Wiktionary is available for most languages. Note that the same input synonymy dictionaries were used by authors of YARN to construct synsets using crowdsourcing. The results on the YARN dataset show how close an automatic synset induction method can approximate manually created synsets provided the same starting material.[16]

Language	# words	# synonyms
English	243 840	212 163
Russian	83 092	211 986

Table 2: Statistics of the input datasets.

5 Results

We compare WATSET with five state-of-the art graph clustering methods presented in Section 2: Chinese Whispers (CW), Markov Clustering (MCL), MaxMax, ECO clustering, and the clique percolation method (CPM). The first two algorithms perform hard clustering, while the last three are fuzzy clustering methods just like our method. While the hard clustering algorithms are able to discover clusters which correspond to synsets composed of unambigous words, they can produce wrong results in the presence of lexical ambiguity (one node belongs to several synsets). In our experiments, we rely on our own implementation of MaxMax and ECO as reference implementations are not available. For CW[17], MCL[18]

[12] https://code.google.com/p/word2vec
[13] http://www.dialog-21.ru/en/evaluation/2015/semantic_similarity
[14] http://russe.nlpub.ru/downloads
[15] We used the Wiktionary dumps of February 1, 2017.
[16] We used the YARN dumps of February 7, 2017.
[17] https://www.github.com/uhh-lt/chinese-whispers
[18] http://java-ml.sourceforge.net

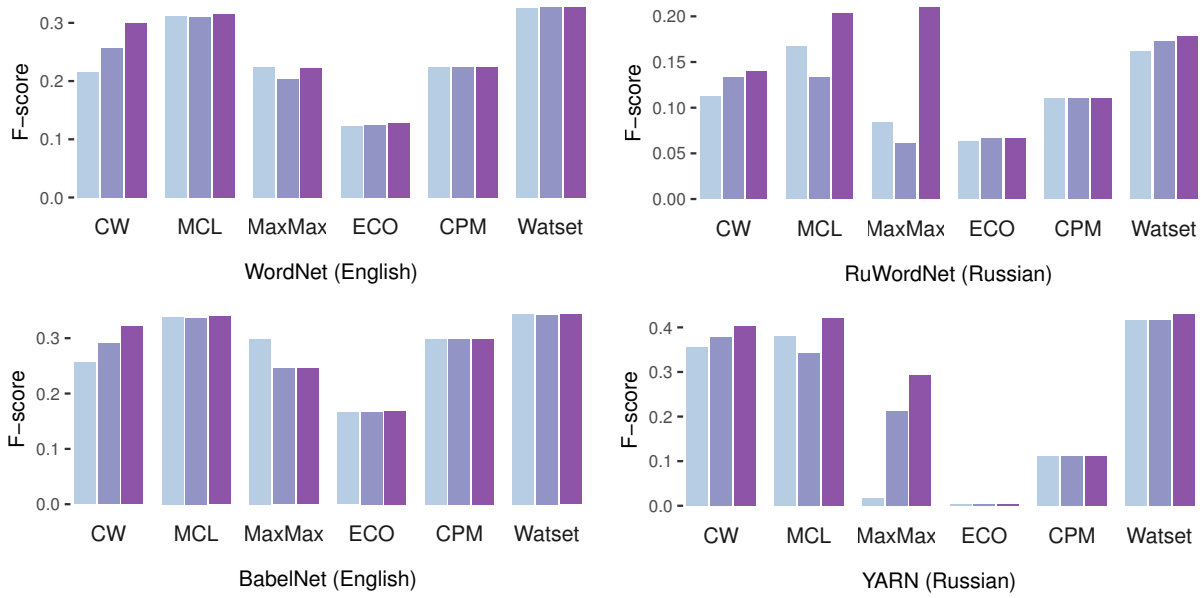


Figure 3: Impact of the different graph weighting schemas on the performance of synset induction: ones, count, sim. Each bar corresponds to the top performance of a method in Tables 3 and 4.

and CPM[19], available implementations have been used. During the evaluation, we delete clusters equal or larger than the threshold of 150 words as they hardly can represent any meaningful synset. The notation WATSET[MCL, CW_{top}] means using MCL for local clustering and Chinese Whispers in the *top* mode for global clustering.

5.1 Impact of Graph Weighting Schema

Figure 3 presents an overview of the evaluation results on both datasets. The first step, common for all of the tested synset induction methods, is graph construction. Thus, we started with an analysis of three ways to weight edges of the graph introduced in Section 3.2: binary scores (*ones*), frequencies (*count*), and semantic similarity scores (*sim*) based on word vector similarity. Results across various configurations and methods indicate that using the weights based on the similarity scores provided by word embeddings is the best strategy for all methods except MaxMax on the English datasets. However, its performance using the *ones* weighting does not exceed the other methods using the *sim* weighting. Therefore, we report all further results on the basis of the *sim* weights. The edge weighting scheme impacts Russian more for most algorithms. The CW algorithm however remains sensitive to the weighting also for the English dataset due to its randomized nature.

[19] https://networkx.github.io

5.2 Comparative Analysis

Table 3 and 4 present evaluation results for both languages. For each method, we show the best configurations in terms of F-score. One may note that the granularity of the resulting synsets, especially for Russian, is very different, ranging from 4 000 synsets for the $CPM_{k=3}$ method to 67 645 induced by the ECO method. Both tables report the number of words, synsets and synonyms after pruning huge clusters larger than 150 words. Without this pruning, the MaxMax and CPM methods tend to discover giant components obtaining almost zero precision as we generate all possible pairs of nodes in such clusters. The other methods did not show such behavior.

WATSET robustly outperforms all other methods according to F-score on both English datasets (Table 3) and on the YARN dataset for Russian (Table 4). Also, it outperforms all other methods according to recall on both Russian datasets. The disambiguation of the input graph performed by the WATSET method splits nodes belonging to several local communities to several nodes, significantly facilitating the clustering task otherwise complicated by the presence of the hubs that wrongly link semantically unrelated nodes.

Interestingly, in all the cases, the toughest competitor was a hard clustering algorithm—MCL (van Dongen, 2000). We observed that the "plain" MCL successfully groups monosemous words, but

Method	# words	# synsets	# synonyms	WordNet			BabelNet		
				P	R	F1	P	R	F1
WATSET[MCL, MCL]	243 840	112 267	345 883	0.345	**0.308**	**0.325**	0.400	0.301	**0.343**
MCL	243 840	84 679	387 315	0.342	0.291	0.314	0.390	0.300	**0.339**
WATSET[MCL, CW_{log}]	243 840	105 631	431 085	0.314	**0.325**	**0.319**	0.359	**0.312**	**0.334**
CW_{top}	243 840	77 879	539 753	0.285	**0.317**	0.300	0.326	**0.317**	0.321
WATSET[CW_{log}, MCL]	243 840	164 689	227 906	**0.394**	0.280	**0.327**	**0.439**	0.245	0.314
WATSET[CW_{log}, CW_{log}]	243 840	164 667	228 523	0.392	0.280	**0.327**	**0.439**	0.245	0.314
$CPM_{k=2}$	186 896	67 109	317 293	**0.561**	0.141	0.225	**0.492**	0.214	0.299
MaxMax	219 892	73 929	797 743	0.176	0.300	0.222	0.202	**0.313**	0.245
ECO	243 840	171 773	84 372	**0.784**	0.069	0.128	**0.699**	0.096	0.169

Table 3: Comparison of the synset induction methods on datasets for English. All methods rely on the similarity edge weighting (*sim*); best configurations of each method in terms of F-scores are shown for each dataset. Results are sorted by F-score on BabelNet, top three values of each metric are boldfaced.

Method	# words	# synsets	# synonyms	RuWordNet			YARN		
				P	R	F1	P	R	F1
WATSET[CW_{nolog}, MCL]	83 092	55 369	332 727	0.120	**0.349**	**0.178**	0.402	**0.463**	**0.430**
WATSET[MCL, MCL]	83 092	36 217	403 068	0.111	0.341	0.168	0.405	0.455	**0.428**
WATSET[CW_{top}, CW_{log}]	83 092	55 319	341 043	0.116	**0.351**	0.174	0.386	**0.474**	**0.425**
MCL	83 092	21 973	353 848	0.155	0.291	**0.203**	0.550	0.340	0.420
WATSET[MCL, CW_{top}]	83 092	34 702	473 135	0.097	**0.361**	0.153	0.351	**0.496**	0.411
CW_{nolog}	83 092	19 124	672 076	0.087	0.342	0.139	0.364	0.451	0.403
MaxMax	83 092	27 011	461 748	**0.176**	0.261	**0.210**	**0.582**	0.195	0.292
$CPM_{k=3}$	15 555	4 000	45 231	**0.234**	0.072	0.111	**0.626**	0.060	0.110
ECO	83 092	67 645	18 362	**0.724**	0.034	0.066	**0.904**	0.002	0.004

Table 4: Results on Russian sorted by F-score on YARN, top three values of each metric are boldfaced.

isolates the neighborhood of polysemous words, which results in the recall drop in comparison to WATSET. CW operates faster due to a simplified update step. On the same graph, CW tends to produce larger clusters than MCL. This leads to a higher recall of "plain" CW as compared to the "plain" MCL, at the cost of lower precision.

Using MCL instead of CW for sense induction in WATSET expectedly produces more fine-grained senses. However, at the global clustering step, these senses erroneously tend to form coarse-grained synsets connecting unrelated senses of the ambiguous words. This explains the generally higher recall of WATSET[MCL, ·]. Despite the randomized nature of CW, variance across runs do not affect the overall ranking: The rank of different versions of CW (*log, nolog, top*) can change, while the rank of the best CW configuration compared to other methods remains the same.

The MaxMax algorithm shows mixed results. On the one hand, it outputs large clusters uniting more than hundred nodes. This inevitably leads to a high recall, as it is clearly seen in the results for Russian because such synsets still pass under our cluster size threshold of 150 words. Its synsets on English datasets are even larger and get pruned, which results in low recall. On the other hand, smaller synsets having at most 10–15 words were identified correctly. MaxMax appears to be extremely sensible to edge weighting, which also complicates its practical use.

The CPM algorithm showed unsatisfactory results, emitting giant components encompassing thousands of words. Such clusters were automatically pruned, but the remaining clusters are relatively correctly built synsets, which is confirmed by the high values of precision. When increasing the minimal number of elements in the clique k, recall improves, but at the cost of a dramatic precision drop. We suppose that the network structure assumptions exploited by CPM do not accurately model the structure of our synonymy graphs.

Finally, the ECO method yielded the worst results because the most cluster candidates failed to pass through the constant threshold used for estimating whether a pair of words should be included in the same cluster. Most synsets produced by this method were trivial, i.e., containing only a single

Resource		P	R	F1
BabelNet on WordNet	En	0.729	0.998	0.843
WordNet on BabelNet	En	0.998	0.699	0.822
YARN on RuWordNet	Ru	0.164	0.162	0.163
BabelNet on RuWordNet	Ru	0.348	0.409	0.376
RuWordNet on YARN	Ru	0.670	0.121	0.205
BabelNet on YARN	Ru	0.515	0.109	0.180

Table 5: Performance of lexical resources cross-evaluated against each other.

word. The remaining synsets for both languages have at most three words that have been connected by a chance due to the edge noising procedure used in this method resulting in low recall.

6 Discussion

On the absolute scores. The results obtained on all gold standards (Figure 3) show similar trends in terms of relative ranking of the methods. Yet absolute scores of YARN and RuWordNet are substantially different due to the inherent difference of these datasets. RuWordNet is more domain-specific in terms of vocabulary, so our input set of generic synonymy dictionaries has a limited coverage on this dataset. On the other hand, recall calculated on YARN is substantially higher as this resource was manually built on the basis of synonymy dictionaries used in our experiments.

The reason for low absolute numbers in evaluations is due to an inherent vocabulary mismatch between the input dictionaries of synonyms and the gold datasets. To validate this hypothesis, we performed a cross-resource evaluation presented in Table 5. The low performance of the cross-evaluation of the two resources supports the hypothesis: no single resource for Russian can obtain high recall scores on another one. Surprisingly, even BabelNet, which integrates most of available lexical resources, still does not reach a recall substantially larger than 0.5.[20] Note that the results of this cross-dataset evaluation are not directly comparable to results in Table 4 since in our experiments we use much smaller input dictionaries than those used by BabelNet.

On sparseness of the input dictionary. Table 6 presents some examples of the obtained synsets of various sizes for the top WATSET configuration on both languages. As one might observe, the quality of the results is highly plausible. However, one limitation of all approaches considered in this paper is the dependence on the completeness of the input dictionary of synonyms. In some parts of the input synonymy graph, important bridges between words can be missing, leading to smaller-than-desired synsets. A promising extension of the present methodology is using distributional models to enhance connectivity of the graph by cautiously adding extra relations.

Size	Synset
2	{*decimal point*, *dot*}
3	{*gullet*, *throat*, *food pipe*}
4	{*microwave meal*, *ready meal*, *TV dinner*, *frozen dinner*}
5	{*objective case*, *accusative case*, *oblique case*, *object case*, *accusative*}
6	{*radio theater*, *dramatized audiobook*, *audio theater*, *radio play*, *radio drama*, *audio play*}

Table 6: Sample synsets induced by the WATSET[MCL, MCL] method for English.

7 Conclusion

We presented a new robust approach to fuzzy graph clustering that relies on hard graph clustering. Using ego network clustering, the nodes belonging to several local communities are split into several nodes each belonging to one community. The transformed "disambiguated" graph is then clustered using an efficient hard graph clustering algorithm, obtaining a fuzzy clustering as the result. The disambiguated graph facilitates clustering as it contains fewer hubs connecting unrelated nodes from different communities. We apply this meta clustering algorithm to the task of synset induction on two languages, obtaining the best results on three datasets and competitive results on one dataset in terms of F-score as compared to five state-of-the-art graph clustering methods.

Acknowledgments

We acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG) foundation under the "JOIN-T" project, the DAAD, the RFBR under the project no. 16-37-00354 mol_a, and the RFH under the project no. 16-04-12019. We also thank three anonymous reviewers for their helpful comments, Andrew Krizhanovsky for providing a parsed Wiktionary, Natalia Loukachevitch for the provided RuWordNet dataset, and Denis Shirgin who suggested the WATSET name.

[20] We used BabelNet 3.7 extracting all 3 497 327 synsets that were marked as Russian.

References

Nikolay Abramov. 1999. *The dictionary of Russian synonyms and semantically related expressions [Slovar' russkikh sinonimov i skhodnykh po smyslu vyrazhenii]*. Russian Dictionaries [Russkie slovari], Moscow, Russia, 7th edition. In Russian.

Marco Baroni and Alessandro Lenci. 2010. Distributional Memory: A General Framework for Corpus-based Semantics. *Computational Linguistics* 36(4):673–721. https://doi.org/10.1162/coli_a_00016.

Chris Biemann. 2006. Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*. Association for Computational Linguistics, New York City, NY, USA, TextGraphs-1, pages 73–80. http://dl.acm.org/citation.cfm?id=1654774.

Chris Biemann. 2012. *Structure Discovery in Natural Language*. Theory and Applications of Natural Language Processing. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-25923-4.

Chris Biemann and Martin Riedl. 2013. Text: now in 2D! A framework for lexical expansion with contextual similarity. *Journal of Language Modelling* 1(1):55–95. https://doi.org/10.15398/jlm.v1i1.60.

Immanuel M. Bomze, Marco Budinich, Panos M. Pardalos, and Marcello Pelillo. 1999. The maximum clique problem. In *Handbook of Combinatorial Optimization*, Springer US, pages 1–74. https://doi.org/10.1007/978-1-4757-3023-4_1.

Pavel Braslavski, Dmitry Ustalov, Mukhin Mukhin, and Yuri Kiselev. 2016. YARN: Spinning-in-Progress. In *Proceedings of the 8th Global WordNet Conference*. Global WordNet Association, Bucharest, Romania, GWC 2016, pages 58–65. http://gwc2016.racai.ro/procedings.pdf.

Antonio Di Marco and Roberto Navigli. 2012. Clustering and Diversifying Web Search Results with Graph-Based Word Sense Induction. *Computational Linguistics* 39(3):709–754. https://doi.org/10.1162/COLI_a_00148.

Vyachelav G. Dikonov. 2013. Development of lexical basis for the Universal Dictionary of UNL Concepts. In *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*. RGGU, Moscow, volume 12 (19), pages 212–221. http://www.dialog-21.ru/media/1238/dikonovv.pdf.

Beate Dorow and Dominic Widdows. 2003. Discovering Corpus-Specific Word Senses. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 2*. Association for Computational Linguistics, Budapest, Hungary, EACL '03, pages 79–82. https://doi.org/10.3115/1067737.1067753.

Martin Everett and Stephen P. Borgatti. 2005. Ego network betweenness. *Social Networks* 27(1):31–38. https://doi.org/10.1016/j.socnet.2004.11.007.

Stefano Faralli, Alexander Panchenko, Chris Biemann, and Simone P. Ponzetto. 2016. Linked Disambiguated Distributional Semantic Networks. In *The Semantic Web – ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II*. Springer International Publishing, Cham, pages 56–64. https://doi.org/10.1007/978-3-319-46547-0_7.

David Gfeller, Jean-Cédric Chappelier, and Paulo De Los Rios. 2005. Synonym Dictionary Improvement through Markov Clustering and Clustering Stability. In *Proceedings of the International Symposium on Applied Stochastic Models and Data Analysis*. pages 106–113. https://conferences.telecom-bretagne.eu/asmda2005/IMG/pdf/proceedings/106.pdf.

Hugo Gonçalo Oliveira and Paolo Gomes. 2014. ECO and Onto.PT: a flexible approach for creating a Portuguese wordnet automatically. *Language Resources and Evaluation* 48(2):373–393. https://doi.org/10.1007/s10579-013-9249-9.

Zhiguo Gong, Chan Wa Cheang, and U. Leong Hou. 2005. Web Query Expansion by WordNet. In *Proceedings of the 16th International Conference on Database and Expert Systems Applications - DEXA '05*, Springer Berlin Heidelberg, Copenhagen, Denmark, pages 166–175. https://doi.org/10.1007/11546924_17.

Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer, and Christian Wirth. 2012. UBY – A Large-Scale Unified Lexical-Semantic Resource Based on LMF. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Avignon, France, EACL '12, pages 580–590. http://www.aclweb.org/anthology/E12-1059.

Iryna Gurevych, Judith Eckle-Kohler, and Michael Matuschek. 2016. *Linked Lexical Knowledge Bases: Foundations and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Kris Heylen, Yves Peirsman, Dirk Geeraerts, and Dirk Speelman. 2008. Modelling Word Similarity: an Evaluation of Automatic Synonymy Extraction Algorithms. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*. European Language Resources Association, Marrakech, Morocco, LREC 2008, pages 3243–3249. http://www.lrec-conf.org/proceedings/lrec2008/pdf/818_paper.pdf.

David Hope and Bill Keller. 2013. MaxMax: A Graph-Based Soft Clustering Algorithm Applied to Word Sense Induction. In *Computational Linguistics*

and Intelligent Text Processing: 14th International Conference, CICLing 2013, Samos, Greece, March 24-30, 2013, Proceedings, Part I, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 368–381. https://doi.org/10.1007/978-3-642-37247-6_30.

David Jurgens and Ioannis Klapaftis. 2013. SemEval-2013 Task 13: Word Sense Induction for Graded and Non-Graded Senses. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, GA, USA, pages 290–299. http://www.aclweb.org/anthology/S13-2049.

Yuri Kiselev, Sergey V. Porshnev, and Mikhail Mukhin. 2015. Current Status of Russian Electronic Thesauri: Quality, Completeness and Availability [Sovremennoe sostoyanie elektronnykh tezaurusov russkogo yazyka: kachestvo, polnota i dostupnost']. *Programmnaya Ingeneria* 6:34–40. In Russian. http://novtex.ru/prin/full/06_2015.pdf.

Andrew A. Krizhanovsky and Alexander V. Smirnov. 2013. An approach to automated construction of a general-purpose lexical ontology based on Wiktionary. *Journal of Computer and Systems Sciences International* 52(2):215–225. https://doi.org/10.1134/S1064230713020068.

Cody Kwok, Oren Etzioni, and Daniel S. Weld. 2001. Scaling Question Answering to the Web. *ACM Transactions on Information Systems* 19(3):242–262. https://doi.org/10.1145/502115.502117.

Dekang Lin. 1998. An Information-Theoretic Definition of Similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., Madison, WI, USA, ICML '98, pages 296–304. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.1832&rep=rep1&type=pdf.

Natalia Loukachevitch. 2011. *Thesauri in information retrieval tasks [Tezaurusy v zadachakh informatsionnogo poiska]*. Moscow University Press [Izd-vo MGU], Moscow, Russia. In Russian.

Natalia V. Loukachevitch, German Lashevich, Anastasia A. Gerasimova, Vladimir V. Ivanov, and Boris V. Dobrov. 2016. Creating Russian WordNet by Conversion. In *Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue"*. RSUH, Moscow, Russia, pages 405–415. http://www.dialog-21.ru/media/3409/loukachevitchnvetal.pdf.

Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach, and Sameer Pradhan. 2010. SemEval-2010 Task 14: Word Sense Induction & Disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Uppsala, Sweden, pages 63–68. http://www.aclweb.org/anthology/S10-1011.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., Harrahs and Harveys, NV, USA, pages 3111–3119. https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* 38(11):39–41. https://doi.org/10.1145/219717.219748.

Roberto Navigli and Simone P. Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193:217–250. https://doi.org/10.1016/j.artint.2012.07.001.

Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435:814–818. https://doi.org/10.1038/nature03607.

Alexander Panchenko. 2011. Comparison of the Baseline Knowledge-, Corpus-, and Web-based Similarity Measures for Semantic Relations Extraction. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. Association for Computational Linguistics, Edinburgh, UK, pages 11–21. http://www.aclweb.org/anthology/W11-2502.

Alexander Panchenko, Eugen Ruppert, Stefano Faralli, Simone P. Ponzetto, and Chris Biemann. 2017a. Unsupervised Does Not Mean Uninterpretable: The Case for Word Sense Induction and Disambiguation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 86–98. http://www.aclweb.org/anthology/E17-1009.

Alexander Panchenko, Dmitry Ustalov, Nikolay Arefyev, Denis Paperno, Natalia Konstantinova, Natalia Loukachevitch, and Chris Biemann. 2017b. Human and Machine Judgements for Russian Semantic Relatedness. In *Analysis of Images, Social Networks and Texts: 5th International Conference, AIST 2016, Yekaterinburg, Russia, April 7-9, 2016, Revised Selected Papers*. Springer International Publishing, Yekaterinburg, Russia, pages 221–235. https://doi.org/10.1007/978-3-319-52920-2_21.

Patrick Pantel and Dekang Lin. 2002. Discovering Word Senses from Text. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Edmonton, Alberta, Canada, KDD '02, pages 613–619. https://doi.org/10.1145/775047.775138.

Maria Pelevina, Nikolay Arefiev, Chris Biemann, and Alexander Panchenko. 2016. Making Sense of Word Embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Berlin, Germany, pages 174–183. http://anthology.aclweb.org/W16-1620.

Stijn van Dongen. 2000. *Graph Clustering by Flow Simulation*. Ph.D. thesis, University of Utrecht.

Jean Véronis. 2004. HyperLex: lexical cartography for information retrieval. *Computer Speech & Language* 18(3):223–252. https://doi.org/10.1016/j.csl.2004.05.002.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*. European Language Resources Association, Marrakech, Morocco, pages 1646–1652. http://www.lrec-conf.org/proceedings/lrec2008/pdf/420_paper.pdf.

Guangyou Zhou, Yang Liu, Fang Liu, Daojian Zeng, and Jun Zhao. 2013. Improving Question Retrieval in Community Question Answering Using World Knowledge. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. AAAI Press, Beijing, China, IJCAI '13, pages 2239–2245. https://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/download/6581/7029.

Neural Modeling of Multi-Predicate Interactions for Japanese Predicate Argument Structure Analysis

Hiroki Ouchi[1,2] **Hiroyuki Shindo**[1,2] **Yuji Matsumoto**[1,2]
[1] Nara Institute of Science and Technology
[2] RIKEN Center for Advanced Intelligence Project (AIP)
{ ouchi.hiroki.nt6, shindo, matsu }@is.naist.jp

Abstract

The performance of Japanese predicate argument structure (PAS) analysis has improved in recent years thanks to the joint modeling of interactions between multiple predicates. However, this approach relies heavily on syntactic information predicted by parsers, and suffers from error propagation. To remedy this problem, we introduce a model that uses *grid-type recurrent neural networks*. The proposed model automatically induces features sensitive to *multi-predicate interactions* from the word sequence information of a sentence. Experiments on the NAIST Text Corpus demonstrate that without syntactic information, our model outperforms previous syntax-dependent models.

1 Introduction

Predicate argument structure (PAS) analysis is a basic semantic analysis task, in which systems are required to identify the semantic units of a sentence, such as *who did what to whom*. In pro-drop languages such as Japanese, Chinese and Italian, arguments are often omitted in text, and such *argument omission* is regarded as one of the most problematic issues facing PAS analysis (Iida and Poesio, 2011; Sasano and Kurohashi, 2011; Hangyo et al., 2013).

In response to the argument omission problem, in Japanese PAS analysis, a joint model of the interactions between multiple predicates has been gaining popularity and achieved the state-of-the-art results (Ouchi et al., 2015; Shibata et al., 2016). This approach is based on the linguistic intuition that the predicates in a sentence are semantically related to each other, and capturing this relation can be useful for PAS analysis. In the example sentence in Figure 1, the word "男 $_i$ (man$_i$)" is the accusative argument of the predicate "逮捕した (arrested)" and is shared by the other predicate "逃走した (escaped)" as its nominative argument. Considering the semantic relation between "逮捕した (arrested)" and "逃走した (escaped)", we intuitively know that the person arrested by someone is likely to be the escaper. That is, information about one predicate-argument relation could help to identify another predicate-argument relation.

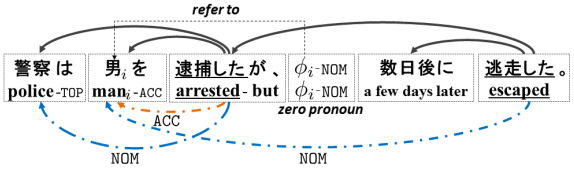


Figure 1: Example of Japanese PAS. The upper edges denote dependency relations, and the lower edges denote case arguments. "NOM" and "ACC" denote the nominative and accusative arguments, respectively. "ϕ_i" is a *zero pronoun*, referring to the *antecedent* "男 $_i$ (man$_i$)".

However, to model such *multi-predicate interactions*, the joint approach in the previous studies relies heavily on syntactic information, such as part-of-speech (POS) tags and dependency relations predicted by POS taggers and syntactic parsers. Consequently, it suffers from error propagation caused by pipeline processing.

To remedy this problem, we propose a neural model which automatically induces features sensitive to multi-predicate interactions exclusively from the word sequence information of a sentence. The proposed model takes as input all predicates and their argument candidates in a sentence at a time, and captures the interactions using grid-type recurrent neural networks (Grid-RNN) without syntactic information.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1591–1600
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1146

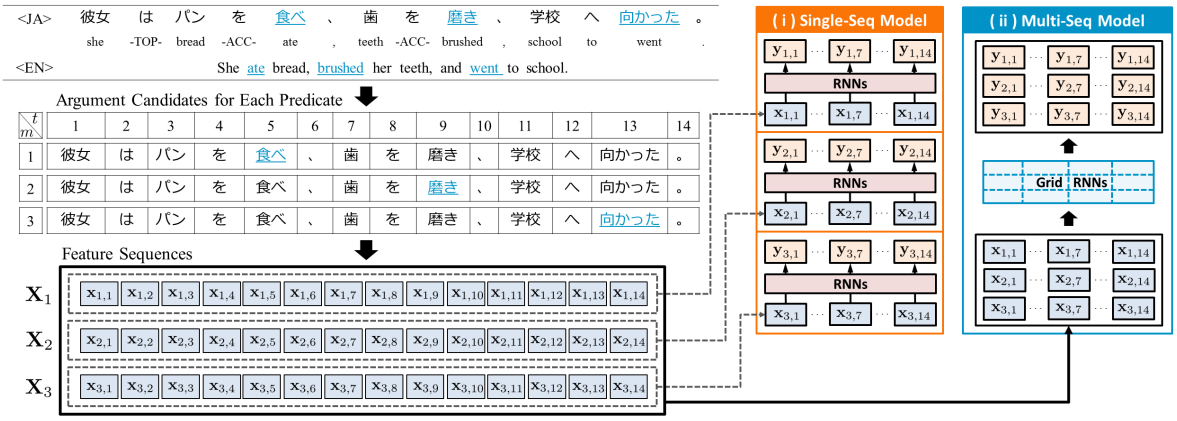


Figure 2: Overview of neural models: (i) *single-sequence* and (ii) *multi-sequence* models.

In this paper, we first introduce a basic model that uses RNNs. This model independently estimates the arguments of each predicate without considering multi-predicate interactions (Sec. 3). Then, extending this model, we propose a neural model that uses Grid-RNNs (Sec. 4).

Performing experiments on the NAIST Text Corpus (Iida et al., 2007), we demonstrate that even without syntactic information, our neural models outperform previous syntax-dependent models (Imamura et al., 2009; Ouchi et al., 2015). In particular, the neural model using Grid-RNNs achieved the best result. This suggests that the proposed grid-type neural architecture effectively captures multi-predicate interactions and contributes to performance improvements. [1]

2 Japanese Predicate Argument Structure Analysis

2.1 Task Description

In Japanese PAS analysis, arguments are identified that each fulfills one of the three major case roles, *nominative* (NOM), *accusative* (ACC) and *dative* (DAT) cases, for each predicate. Arguments can be divided into the following three categories according to the positions relative to their predicates (Hayashibe et al., 2011; Ouchi et al., 2015):

Dep: Arguments that have direct syntactic dependency on the predicate.

Zero: Arguments referred to by zero pronouns within the same sentence that have no direct syntactic dependency on the predicate.

Inter-Zero: Arguments referred to by zero pronouns outside of the same sentence.

For example, in Figure 1, the nominative argument "警察 (police)" for the predicate "逮捕した (arrested)" is regarded as a *Dep* argument, because the argument has a direct syntactic dependency on the predicate. By contrast, the nominative argument "男 $_i$ (man$_i$)" for the predicate "逃走した (escaped)" is regarded as a *Zero* argument, because the argument has no direct syntactic dependency on the predicate.

In this paper, we focus on the analysis for these intra-sentential arguments, i.e., *Dep* and *Zero*. In order to identify inter-sentential arguments (*Inter-Zero*), a much broader space must be searched (e.g., the whole document), resulting in a much more complicated analysis than intra-sentential arguments.[2] Owing to this complication, Ouchi et al. (2015) and Shibata et al. (2016) focused exclusively on intra-sentential argument analysis. Following this trend, we also restrict our focus to intra-sentential argument analysis.

2.2 Challenging Problem

Arguments are often omitted in Japanese sentences. In Figure 1, ϕ_i represents the omitted argument, called the *zero pronoun*. This zero pronoun ϕ_i refers to "男 $_i$ (man$_i$)". In Japanese PAS analysis, when an argument of the target predicate is omitted, we have to identify the antecedent of the omitted argument (i.e., the *Zero* argument).

The analysis of such *Zero* arguments is much more difficult than that for *Dep* arguments, owing to the lack of direct syntactic dependencies. For *Dep* arguments, the syntactic dependency between an argument and its predicate is a strong clue. In the sentence in Figure 1, for the predi-

[1] Our source code is publicly available at https://github.com/hiroki13/neural-pasa-system

[2] The F-measure remains 10-20% (Taira et al., 2008; Imamura et al., 2009; Sasano and Kurohashi, 2011).

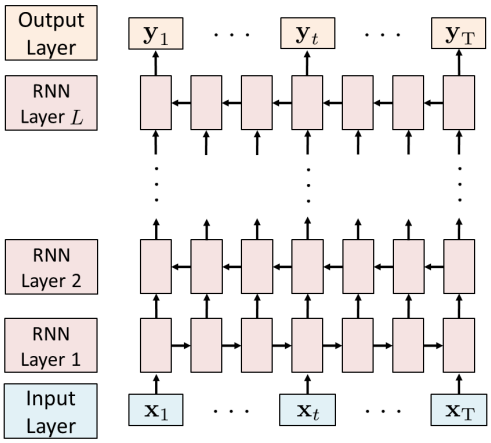


Figure 3: Overall architecture of the single-sequence model. This model consists of three components: (i) Input Layer, (ii) RNN Layer and (iii) Output Layer.

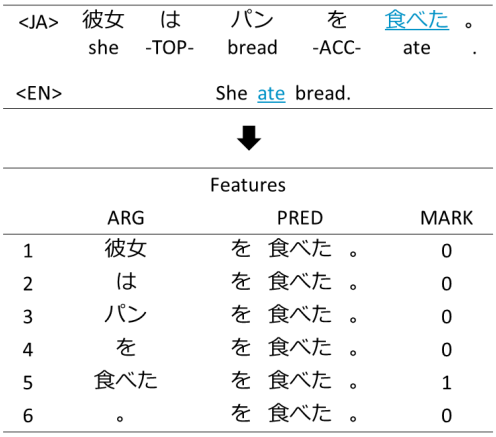


Figure 4: Example of feature extraction. The underlined word is the target predicate. From the sentence "彼女はパンを食べた。(She ate bread.)", three types of features are extracted for the target predicate "食べた (ate)".

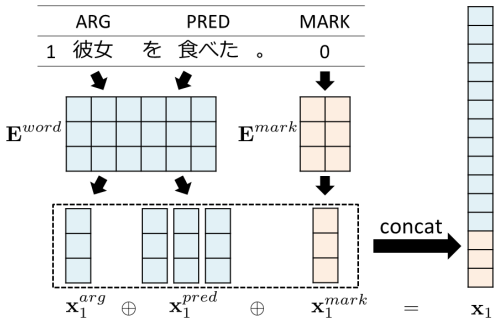


Figure 5: Example of the process of creating a feature vector. The extracted features are mapped to each vector, and all the vectors are concatenated into one feature vector.

cate "逮捕した (arrested)", the nominative argument is "警察 (police)". This argument is easily identified by relying on the syntactic dependency. By contrast, because the nominative argument "男$_i$ (man$_i$)" has no syntactic dependency on its predicate "逃走した (escaped)", we must rely on other information to identify the zero argument.

As a solution to this problem, we exploit two kinds of information: (i) the context of the entire sentence, and (ii) multi-predicate interactions. For the former, we introduce *single-sequence model* that induces context-sensitive representations from a sequence of argument candidates of a predicate. For the latter, we introduce *multi-sequence model* that induces predicate-sensitive representations from multiple sequences of argument candidates of all predicates in a sentence (shown in Figure 2).

3 Single-Sequence Model

The single-sequence model exploits stacked bidirectional RNNs (Bi-RNN) (Schuster and Paliwal, 1997; Graves et al., 2005, 2013; Zhou and Xu, 2015). Figure 3 shows the overall architecture, which consists of the following three components:

Input Layer: Map each word to a feature vector representation.

RNN Layer: Produce high-level feature vectors using Bi-RNNs.

Output Layer: Compute the probability of each case label for each word using the softmax function.

In the following subsections, we describe each of these three components in detail.

3.1 Input Layer

Given an input sentence $w_{1:T} = (w_1, \cdots, w_T)$ and a predicate p, each word w_t is mapped to a feature representation $\mathbf{x}_t$, which is the concatenation ($\oplus$) of three types of vectors:

$$\mathbf{x}_t = \mathbf{x}_t^{arg} \oplus \mathbf{x}_t^{pred} \oplus \mathbf{x}_t^{mark} \quad (1)$$

where each vector is based on the following atomic features inspired by Zhou and Xu (2015):

`ARG`**:** Word index of each word.

`PRED`**:** Word index of the target predicate and the words around the predicate.

`MARK`**:** Binary index that represents whether or not the word is the predicate.

Figure 4 presents an example of the atomic features. For the ARG feature, we extract a word index $x^{word} \in \mathcal{V}$ for each word. Similarly, for the PRED feature, we extract each word index x^{word} for the C words taking the target predicate at the center, where C denotes the window size. The MARK feature $x^{mark} \in \{0, 1\}$ is a binary value that represents whether or not the word is the predicate.

Then, using feature indices, we extract feature vector representations from each embedding matrix. Figure 5 shows the process of creating the feature vector $\mathbf{x}_1$ for the word w_1 "彼女 (she)". We set two embedding matrices: (i) a word embedding matrix $\mathbf{E}^{word} \in \mathbb{R}^{d_{word} \times |\mathcal{V}|}$, and (ii) a mark embedding matrix $\mathbf{E}^{mark} \in \mathbb{R}^{d_{mark} \times 2}$. From each embedding matrix, we extract the corresponding column vectors and concatenate them as a feature vector $\mathbf{x}_t$ based on Eq. 1.

Each feature vector $\mathbf{x}_t$ is multiplied with a parameter matrix $\mathbf{W}_x$:

$$\mathbf{h}_t^{(0)} = \mathbf{W}_x \, \mathbf{x}_t \tag{2}$$

The vector $\mathbf{h}_t^{(0)}$ is given to the first RNN layer as input.

3.2 RNN Layer

In the RNN layers, feature vectors are updated recurrently using Bi-RNNs. Bi-RNNs process an input sequence in a left-to-right manner for odd-numbered layers and in a right-to-left manner for even-numbered layers. By stacking these layers, we can construct the deeper network structures.

Stacked Bi-RNNs consist of L layers, and the hidden state in the layer $\ell \in (1, \cdots, L)$ is calculated as follows:

$$\mathbf{h}_t^{(\ell)} = \begin{cases} g^{(\ell)}(\mathbf{h}_t^{(\ell-1)}, \, \mathbf{h}_{t-1}^{(\ell)}) & (\ell = \text{odd}) \\ g^{(\ell)}(\mathbf{h}_t^{(\ell-1)}, \, \mathbf{h}_{t+1}^{(\ell)}) & (\ell = \text{even}) \end{cases} \tag{3}$$

Both of the odd- and even-numbered layers receive $\mathbf{h}_t^{(\ell-1)}$, the t-th hidden state of the $\ell-1$ layer, as the first input of the function $g^{(\ell)}$, which is an arbitrary function [3]. For the second input of $g^{(\ell)}$, odd-numbered layers receive $\mathbf{h}_{t-1}^{(\ell)}$, whereas even-numbered layers receive $\mathbf{h}_{t+1}^{(\ell)}$. By calculating the hidden states until the L-th layer, we obtain a hidden state sequence $\mathbf{h}_{1:\mathrm{T}}^{(L)} = (\mathbf{h}_1^{(L)}, \cdots, \mathbf{h}_{\mathrm{T}}^{(L)})$. Using each vector $\mathbf{h}_t^{(L)}$, we calculate the probability of case labels for each word in the output layer.

[3] In this work, we used the Gated Recurrent Unit (GRU) (Cho et al., 2014) as the function $g^{(\ell)}$.

3.3 Output Layer

For the output layer, multi-class classification is performed using the softmax function:

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}_y \, \mathbf{h}_t^{(L)})$$

where $\mathbf{h}_t^{(L)}$ denotes a vector representation propagated from the last RNN layer (Fig 3). Each element of $\mathbf{y}_t$ is a probability value corresponding to each label. The label with the maximum probability among them is output as a result. In this work, we set five labels: NOM, ACC, DAT, PRED, null. PRED is the label for the predicate, and null denotes a word that does not fulfill any case role.

4 Multi-Sequence Model

Whereas the single-sequence model assumes independence between predicates, the multi-sequence model assumes *multi-predicate interactions*. To capture such interactions between all predicates in a sentence, we extend the single-sequence model to the multi-sequence model using Grid-RNNs (Graves and Schmidhuber, 2009; Kalchbrenner et al., 2016). Figure 6 presents the overall architecture for the multi-sequence model, which consists of three components:

Input Layer: Map words to M sequences of feature vectors for M predicates.

Grid Layer: Update the hidden states over different sequences using Grid-RNNs.

Output Layer: Compute the probability of each case label for each word using the softmax function.

In the following subsections, we describe these three components in detail.

4.1 Input Layer

The multi-sequence model takes as input a sentence $w_{1:\mathrm{T}} = (w_1, \cdots, w_{\mathrm{T}})$ and all predicates $\{p_m\}_1^M$ in the sentence. For each predicate p_m, the input layer creates a sequence of feature vectors $\mathbf{X}_m = (\mathbf{x}_{m,1}, \cdots, \mathbf{x}_{m,\mathrm{T}})$ by mapping each input word w_t to a feature vector $\mathbf{x}_{m,t}$ based on Eq 1. That is, for M predicates, M sequences of feature vectors $\{\mathbf{X}_m\}_1^M$ are created.

Then, using Eq. 2, each feature vector $\mathbf{x}_{m,t}$ is mapped to $\mathbf{h}_{m,t}^{(0)}$, and a feature sequence is created for a predicate p_m, i.e., $\mathbf{H}_m^{(0)} = (\mathbf{h}_{m,1}^{(0)}, \cdots, \mathbf{h}_{m,\mathrm{T}}^{(0)})$. Consequently, for M predicates, we obtain M feature sequences $\{\mathbf{H}_m^{(0)}\}_1^M$.

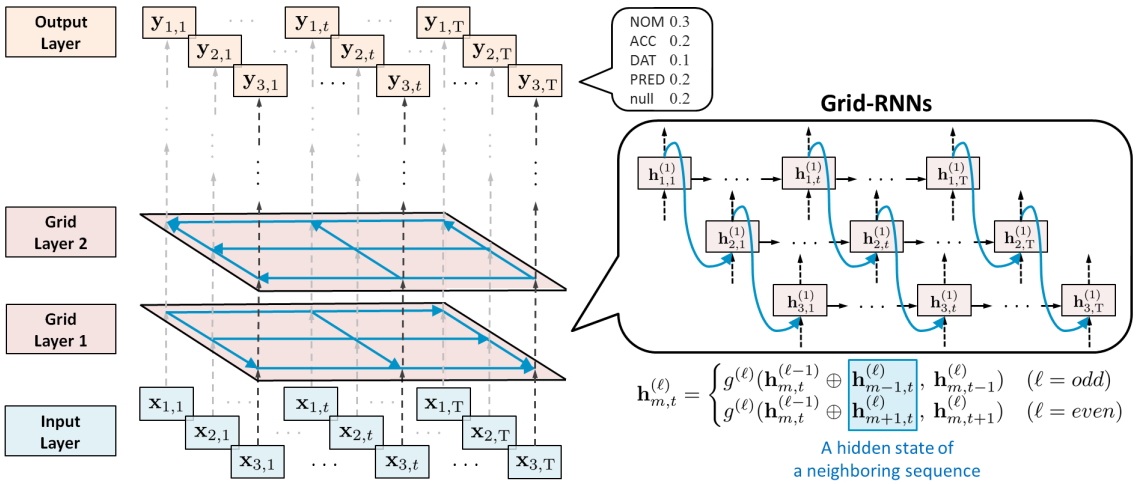


Figure 6: Overall architecture of the multi-sequence model: an example of three sequences.

4.2 Grid Layer

Inter-Sequence Connections

For the grid layers, we use Grid-RNNs to propagate the feature information over the different sequences (*inter-sequence connections*). The figure on the right in Figure 6 shows the first grid layer. The hidden state is recurrently calculated from the upper-left ($m = 1, t = 1$) to the lower-right ($m = M, t = \mathrm{T}$).

Formally, in the ℓ-th layer, the hidden state $\mathbf{h}^{(\ell)}_{m,t}$ is calculated as follows:

$$\mathbf{h}^{(\ell)}_{m,t} = \begin{cases} g^{(\ell)}(\mathbf{h}^{(\ell-1)}_{m,t} \oplus \mathbf{h}^{(\ell)}_{m-1,t}, \mathbf{h}^{(\ell)}_{m,t-1}) & (\ell = \text{odd}) \\ g^{(\ell)}(\mathbf{h}^{(\ell-1)}_{m,t} \oplus \mathbf{h}^{(\ell)}_{m+1,t}, \mathbf{h}^{(\ell)}_{m,t+1}) & (\ell = \text{even}) \end{cases}$$

This equation is similar to Eq. 3. The main difference is that the hidden state of a neighboring sequence, $\mathbf{h}^{(\ell)}_{m-1,t}$ (or $\mathbf{h}^{(\ell)}_{m+1,t}$), is concatenated ($\oplus$) with the hidden state of the previous ($\ell - 1$) layer, $\mathbf{h}^{(\ell-1)}_{m,t}$, and is taken as input of the function $g^{(\ell)}$.

In the figure on the right in Figure 6, the blue curved lines represent the inter-sequence connections. Taking as input the hidden states of neighboring sequences, the network propagates feature information over multiple sequences (i.e., predicates). By calculating the hidden states until the L-th layer, we obtain M sequences of the hidden states, i.e., $\{\mathbf{H}^{(L)}_m\}^M_1$, in which $\mathbf{H}^{(L)}_m = (\mathbf{h}^{(L)}_{m,1}, \cdots, \mathbf{h}^{(L)}_{m,\mathrm{T}})$.

Residual Connections

As more layers are stacked, it becomes more difficult to learn the model parameters, owing to various challenges such as the vanishing gradient problem (Pascanu et al., 2013). In this work, we integrate residual connections (He et al., 2015; Wu et al., 2016) with our networks to form connections between layers. Specifically, the input vector $\mathbf{h}^{(\ell-1)}_{m,t}$ of the ℓ-th layer is added to the output vector $\mathbf{h}^{(\ell)}_{m,t}$. Residual connections can also be applied to the single-sequence model. Thus, we can perform experiments on both models with/without residual connections.

4.3 Output Layer

As with the single-sequence model, we use the softmax function to calculate the probability of the case labels of each word w_t for each predicate p_m:

$$\mathbf{y}_{m,t} = \mathrm{softmax}(\mathbf{W}_y\, \mathbf{h}^{(L)}_{m,t})$$

where $\mathbf{h}^{(L)}_{m,t}$ is a hidden state vector calculated in the last grid layer.

5 Related Work

5.1 Japanese PAS Analysis Approaches

Existing approaches to Japanese PAS analysis are divided into two categories: (i) the *pointwise approach* and (ii) the *joint approach*. The pointwise approach involves estimating the score of each argument candidate for one predicate, and then selecting the argument candidate with the maximum score as an argument (Taira et al., 2008; Imamura et al., 2009; Hayashibe et al., 2011; Iida et al., 2016). The joint approach involves scoring all the predicate-argument combinations in one sentence, and then selecting the combination with the highest score (Yoshikawa et al., 2011; Sasano and Kurohashi,

2011; Ouchi et al., 2015; Shibata et al., 2016). Compared with the pointwise approach, the joint approach achieves better results.

5.2 Multi-Predicate Interactions

Ouchi et al. (2015) reported that it is beneficial to Japanese PAS analysis to capture the interactions between all predicates in a sentence. This is based on the linguistic intuition that the predicates in a sentence are semantically related to each other, and that the information regarding this semantic relation can be useful for PAS analysis.

Similarly, in semantic role labeling (SRL), Yang and Zong (2014) reported that their reranking model, which captures the multi-predicate interactions, is effective for the English constituent-based SRL task (Carreras and Màrquez, 2005). Taking this a step further, we propose a neural architecture that effectively models the multi-predicate interactions.

5.3 Neural Approaches

Japanese PAS

In recent years, several attempts have been made to apply neural networks to Japanese PAS analysis (Shibata et al., 2016; Iida et al., 2016)[4]. In Shibata et al. (2016), a feed-forward neural network is used for the score calculation part of the joint model proposed by Ouchi et al. (2015). In Iida et al. (2016), multi-column convolutional neural networks are used for the zero anaphora resolution task.

Both models exploit syntactic and selectional preference information as the atomic features of neural networks. Overall, the use of neural networks has resulted in advantageous performance levels, mitigating the cost of manually designing combination features. In this work, we demonstrate that even without such syntactic information, our neural models can realize comparable performance exclusively using the word sequence information of a sentence.

English SRL

Some neural models have achieved high performance without syntactic information in English SRL. Collobert et al. (2011) and Zhou and Xu (2015) worked on the English constituent-based SRL task (Carreras and Màrquez, 2005) using neural networks. In Collobert et al. (2011), their model exploited a convolutional neural network and achieved a 74.15% F-measure without syntactic information. In Zhou and Xu (2015), their model exploited bidirectional RNNs with linear-chain conditional random fields (CRFs) and achieved the state-of-the-art result, an 81.07% F-measure. Our models should be regarded as an extension of their model.

The main differences between Zhou and Xu (2015) and our work are: (i) constituent-based vs dependency-based argument identification and (ii) the multi-predicate consideration. For the constituent-based SRL, Zhou and Xu (2015) used CRFs to capture the IOB label dependencies, because systems are required to identify the *spans* of arguments for each predicate. By contrast, for Japanese dependency-based PAS analysis, we replaced the CRFs with the softmax function, because in Japanese, arguments are rarely adjacent to each other.[5] Furthermore, whereas the model described in Zhou and Xu (2015) predicts arguments for each predicate independently, our multi-sequence model jointly predicts arguments for all predicates in a sentence concurrently by considering the multi-predicate interactions.

6 Experiments

6.1 Experimental Settings

Dataset

We used the NAIST Text Corpus 1.5, which consists of 40,000 sentences from Japanese newspapers (Iida et al., 2007). For the experiments, we adopted standard data splits (Taira et al., 2008; Imamura et al., 2009; Ouchi et al., 2015):

Train:	Articles: Jan 1-11, Editorials: Jan-Aug
Dev:	Articles: Jan 12-13, Editorials: Sept
Test:	Articles: Jan 14-17, Editorials: Oct-Dec

We used the word boundaries annotated in the NAIST Text Corpus and the target predicates that have at least one argument in the same sentence. We did not use any external resources.

Learning

We trained the model parameters by minimizing

[4] These previous studies used unpublished datasets and evaluated the performance with different experimental settings. Consequently, we cannot compare their models with ours.

[5] In our preliminary experiment, we could not confirm the performance improvement by CRFs.

the cross-entropy loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{n}\sum_{t} \log P(y_t|x_t) + \frac{\lambda}{2}||\boldsymbol{\theta}||^2 \quad (4)$$

where θ is a set of model parameters, and the hyper-parameter λ is the coefficient governing the L2 weight decay.

Implementation Details

We implemented our neural models using a deep learning library, Theano (Bastien et al., 2012). The number of epochs was set at 50, and we reported the result of the test set in the epoch with the best F-measure from the development set. The parameters were optimized using the stochastic gradient descent method (SGD) via a mini-batch, whose size was selected from $\{2, 4, 8\}$. The learning rate was automatically adjusted using Adam (Kingma and Ba, 2014). For the L2 weight decay, the hyper-parameter λ in Eq. 4 was selected from $\{0.001, 0.0005, 0.0001\}$.

In the neural models, the number of the RNN and Grid layers were selected from $\{2, 4, 6, 8\}$. The window size C for the PRED feature (Sec. 3.1) was set at 5. Words with a frequency of 2 or more in the training set were mapped to each word index, and the remaining words were mapped to the unknown word index. The dimensions d_{word} and d_{mark} of the embeddings were set at 32. In the single-sequence model, the parameters of GRUs were set at 32×32. In the multi-sequence model, the parameters of GRUs related to the input values were set at 64×32, and the remaining were set at 32×32. The initial values of all parameters were sampled according to a uniform distribution from $[-\frac{\sqrt{6}}{\sqrt{row+col}}, \frac{\sqrt{6}}{\sqrt{row+col}}]$, where row and col are the number of rows and columns of each matrix, respectively.

Baseline Models

We compared our models to existing models in previous works (Sec. 5.1) that use the NAIST Text Corpus 1.5. As a baseline for the pointwise approach, we used the pointwise model[6] proposed in Imamura et al. (2009). In addition, as a baseline for the joint approach, we used the model proposed in Ouchi et al. (2015). These models exploit gold annotations in the NAIST Text Corpus as POS tags and dependency relations.

[6]We compared the results of the model reimplemented by Ouchi et al. (2015).

	Dep	*Zero*	*All*
Imamura+ 09	85.06	41.65	78.15
Ouchi+ 15	86.07	44.09	79.23
Single-Seq	88.10	46.10	81.15
Multi-Seq	**88.17** †	**47.12** †	**81.42** †

Table 1: F-measures in the test set. Single-Seq is the single-sequence model, and Multi-Seq is the multi-sequence model. Imamura+ 09 is the model in Imamura et al. (2009) reimplemented by Ouchi et al. (2015), and Ouchi+ 15 is the ALL-Cases Joint Model in Ouchi et al. (2015). The mark † denotes the significantly better results with the significance level $p < 0.05$, comparing Single-Seq and Multi-Seq.

6.2 Results

Neural Models vs Baseline Models

Table 1 presents F-measures from our neural sequence models with eight RNN or Grid layers and the baseline models on the test set. For the significant test, we used the bootstrap resampling method. According to all metrics, both the single- (Single-Seq) and multi-sequence models (Multi-Seq) outperformed the baseline models. This confirms that our neural models realize high performance, even without syntactic information, by learning contextual information effective for PAS analysis from the word sequence of the sentence.

In particular, for zero arguments (*Zero*), our models achieved a considerable improvement compared to the joint model in Ouchi et al. (2015). Specifically, the single-sequence model improved by approximately 2.0 points, and the multi-sequence model by approximately 3.0 points according to the F-measure. These results suggest that modeling the context of the entire sentence using RNNs are beneficial to Japanese PAS analysis, particularly to zero argument identification.

Effects of Multiple Predicate Consideration

As Table 1 shows, the multi-sequence model significantly outperformed the single-sequence model in terms of the F-measure overall (81.42% vs 81.15%). These results demonstrate that the grid-type neural architecture can effectively capture multi-predicate interactions by connecting the sequences of the argument candidates for all predicates in a sentence.

Compared to the single-sequence model for dif-

L		Single-Seq		Multi-Seq	
		$+res.$	$-res.$	$+res.$	$-res.$
2	*Dep*	**87.34**	87.10	87.43	**87.73**
	Zero	**47.98**	47.90	**47.66**	46.93
	All	**80.62**	80.24	**80.71**	80.68
4	*Dep*	87.27	**87.41**	**87.60**	87.09
	Zero	50.43	**50.83**	48.10	**48.58**
	All	80.92	**80.99**	**80.99**	80.59
6	*Dep*	**87.73**	87.11	**88.04**	87.39
	Zero	48.81	**49.51**	**48.98**	48.91
	All	**81.05**	80.63	**81.19**	80.68
8	*Dep*	**87.98**	87.23	**87.65**	87.07
	Zero	47.40	**48.38**	**49.34**	48.23
	All	**81.31**	80.33	**81.33**	80.40

Table 2: Performance comparison for different numbers of layers on the development set in F-measures. L is the number of the RNN or Grid layers. $+res.$ or $-res.$ indicates whether the model has residual connections ($+$) or not ($-$).

M	Type	No. Args	Single-Seq	Multi-Seq
1	*Dep*	2,733	**89.97**	89.66
	Zero	154	47.62	**53.54**
	All	2,887	**88.08**	88.01
2	*Dep*	5,674	89.64	**90.11**
	Zero	836	53.87	**54.21**
	All	6,510	85.39	**85.95**
3	*Dep*	6,067	87.72	**88.06**
	Zero	1,357	49.98	**51.82**
	All	7,424	81.43	**82.11**
4	*Dep*	4,616	87.80	**87.84**
	Zero	1,205	47.27	**48.50**
	All	5,821	80.31	**80.69**
5+	*Dep*	6.983	**86.63**	86.30
	Zero	2,467	39.83	**40.66**
	All	9,450	**76.17**	76.00

Table 3: Performance comparison for different numbers (M) of predicates in a sentence on the test set in F-measures.

ferent argument types, the multi-sequence model achieved slightly better results for direct dependency arguments (*Dep*) (88.10% vs 88.17%). In addition, for zero arguments (*Zero*), which have no syntactic dependency on their predicate, the multi-sequence model outperformed the single-sequence model by approximately 1.0 point according to the F-measure (46.10% vs 47.12%). This shows that capturing multi-predicate interactions is particularly effective for zero arguments, which is consistent with the results in Ouchi et al. (2015).

Effects of Network Depth

Table 2 presents F-measures from the neural sequence models with different network depths and with/without residual connections. The performance tends to improve as the RNN or Grid layers get deeper with residual connections. In particular, the two models with eight layers and residual connections achieved considerable improvements of approximately 1.0 point according to the F-measure compared to models without residual connections. This means that residual connections contribute to effective parameter learning of deeper models.

Effects of the Number of Predicates

Table 3 presents F-measures from the neural sequence models with different numbers of predicates in a sentence. In Table 3, M denotes how many predicates appear in a sentence. For example, the sentence in Figure 1 includes two predicates, "arrested" and "escaped", and thus in this example $M = 2$.

Overall, performance of both models gradually deteriorated as the number of predicates in a sentence increased, because sentences that contain many predicates are complex and difficult to analyze. However, compared to the single-sequence model, the multi-sequence model suppressed performance degradation, especially for zero arguments ($Zero$). By contrast, for direct dependency arguments (Dep), both models either achieved almost equivalent performance or the single-sequence model outperformed the multi-sequence model. A Detailed investigation of the relation between the number of predicates in a sentence and the complexity of PAS analysis is an interesting line for future work.

Comparison per Case Role

Table 4 shows F-measures for each case role. For reference, we show the results of the previous studies using the NAIST Text Corpus 1.4β with external resources as well.[7]

[7]The major difference between the NAIST Text Corpus 1.4β and 1.5 is the revision of the annotation criterion for the dative case (DAT) (corresponding to Japanese case marker "に"). Argument and adjunct usages of the case marker "に" are not distinguished in 1.4β, making the identification of the dative case seemingly easy (Ouchi et al., 2015).

	Dep			*Zero*		
	NOM	ACC	DAT	NOM	ACC	DAT
NAIST Text Corpus 1.5						
Imamura+ 09	86.50	92.84	30.97	45.56	21.38	0.83
Ouchi+ 15	88.13	92.74	38.39	48.11	24.43	4.80
Single-Seq	88.32	93.89	65.91	49.51	35.07	9.83
Multi-Seq	88.75	93.68	64.38	50.65	32.35	7.52
NAIST Text Corpus 1.4β						
Taira+ 08*	75.53	88.20	89.51	30.15	11.41	3.66
Imamura+ 09*	87.0	93.9	80.8	50.0	30.8	0.0
Sasano+ 11*	-	-	-	39.5	17.5	8.9

Table 4: Performance comparison for different case roles on the test set in F-measures. NOM, ACC or DAT is the nominal, accusative or dative case, respectively. The asterisk (*) indicates that the model uses external resources.

Comparing the models using the NAIST Text Corpus 1.5, the single- and multi-sequence models outperformed the baseline models according to all metrics. In particular, for the dative case, the two neural models achieved much higher results, by approximately 30 points. This suggests that although dative arguments appear infrequently compared with the other two case arguments, the neural models can learn them robustly.

In addition, for zero arguments (*Zero*), the neural models achieved better results than the baseline models. In particular, for zero arguments of the nominative case (NOM), the multi-sequence model demonstrated a considerable improvement of approximately 2.5 points according to the F-measure compared with the joint model in Ouchi et al. (2015). To achieve high accuracy for the analysis of such zero arguments, it is necessary to capture long distance dependencies (Iida et al., 2005; Sasano and Kurohashi, 2011; Iida et al., 2015). Therefore, the improvements of the results suggest that the neural models effectively capture long distance dependencies using RNNs that can encode the context of the entire sentence.

7 Conclusion

In this work, we introduced neural sequence models that automatically induce effective feature representations from the word sequence information of a sentence for Japanese PAS analysis. The experiments on the NAIST Text Corpus demonstrated that the models realize high performance without the need for syntactic information. In particular, our multi-sequence model improved the performance of *zero argument* identification, one of the problematic issues facing Japanese PAS analysis, by considering the *multi-predicate interactions* with Grid-RNNs.

Because our neural models are applicable to SRL, applying our models for multilingual SRL tasks presents an interesting future research direction. In addition, in this work, the model parameters were learned without any external resources. In future work, we plan to explore effective methods for exploiting large-scale unlabeled data to learn the neural models.

Acknowledgments

This work was partially supported by JST CREST Grant Number JPMJCR1513 and JSPS KAKENHI Grant Number 15K16053. We are grateful to the members of the NAIST Computational Linguistics Laboratory and the anonymous reviewers for their insightful comments.

References

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL*. pages 152–164.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder

for statistical machine translation. In *Proceedings of EMNLP*. pages 1724–1734.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* .

Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *Proceedings of Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop*.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional LSTM networks for improved phoneme classification and recognition. In *Proceedings of International Conference on Artificial Neural Networks*. pages 799–804.

Alex Graves and Jürgen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In *Proceedings of NIPS*. pages 545–552.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2013. Japanese zero reference resolution considering exophora and author/reader mentions. In *Proceedings of EMNLP*. pages 924–934.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of IJCNLP*. pages 201–209.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385* .

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2005. Anaphora resolution by antecedent identification followed by anaphoricity determination. *ACM Transactions on Asian Language Information Processing (TALIP)* 4(4):417–434.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the Linguistic Annotation Workshop*. pages 132–139.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ILP solution to zero anaphora resolution. In *Proceedings of ACL-HLT*. pages 804–813.

Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Julien Kloetzer. 2015. Intra-sentential zero anaphora resolution using subject sharing recognition. In *Proceedings of EMNLP*. pages 2179–2189.

Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh, Canasai Kruengkrai, and Julien Kloetzer. 2016. Intra-sentential subject zero anaphora resolution using multi-column convolutional neural network. In *Proceedings of EMNLP*. pages 1244–1254.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of ACL-IJCNLP*. pages 85–88.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2016. Grid long short-term memory. In *Proceedings of ICLR*.

D.P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

Hiroki Ouchi, Hiroyuki Shindo, Kevin Duh, and Yuji Matsumoto. 2015. Joint case argument identification for Japanese predicate argument structure analysis. In *Proceedings of ACL-IJCNLP*. pages 961–970.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of IJCNLP*. pages 758–766.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* pages 2673–2681.

Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. 2016. Neural network-based model for Japanese predicate argument structure analysis. In *Proceedings of ACL*. pages 1235–1244.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of EMNLP*. pages 523–532.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Haitong Yang and Chengqing Zong. 2014. Multi-predicate semantic role labeling. In *Proceedings of EMNLP*. pages 363–373.

Katsumasa Yoshikawa, Masayuki Asahara, and Yuji Matsumoto. 2011. Jointly extracting Japanese predicate-argument relation with markov logic. In *Proceedings of IJCNLP*. pages 1125–1133.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL-IJCNLP*.

TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension

Mandar Joshi[†] **Eunsol Choi**[†] **Daniel S. Weld**[†] **Luke Zettlemoyer**[†‡]

[†] Paul G. Allen School of Computer Science & Engineering, Univ. of Washington, Seattle, WA
{mandar90, eunsol, weld, lsz}@cs.washington.edu

[‡] Allen Institute for Artificial Intelligence, Seattle, WA
lukez@allenai.org

Abstract

We present TriviaQA, a challenging reading comprehension dataset containing over 650K question-answer-evidence triples. TriviaQA includes 95K question-answer pairs authored by trivia enthusiasts and independently gathered evidence documents, six per question on average, that provide high quality distant supervision for answering the questions. We show that, in comparison to other recently introduced large-scale datasets, TriviaQA (1) has relatively complex, compositional questions, (2) has considerable syntactic and lexical variability between questions and corresponding answer-evidence sentences, and (3) requires more cross sentence reasoning to find answers. We also present two baseline algorithms: a feature-based classifier and a state-of-the-art neural network, that performs well on SQuAD reading comprehension. Neither approach comes close to human performance (23% and 40% vs. 80%), suggesting that TriviaQA is a challenging testbed that is worth significant future study.[1]

1 Introduction

Reading comprehension (RC) systems aim to answer any question that could be posed against the facts in some reference text. This goal is challenging for a number of reasons: (1) the questions can be complex (e.g. have highly compositional semantics), (2) finding the correct answer can require complex reasoning (e.g. combining facts from multiple sentences or background knowledge) and (3) individual facts can be difficult to recover from text (e.g. due to lexical and syntactic variation). Figure 1 shows examples of all these phenomena. This paper presents TriviaQA, a new reading comprehension dataset designed to simultaneously test all of these challenges.

Question: The Dodecanese Campaign of WWII that was an attempt by the Allied forces to capture islands in the Aegean Sea was the inspiration for which acclaimed 1961 commando film?
Answer: The Guns of Navarone
Excerpt: The Dodecanese Campaign of World War II was an attempt by Allied forces to capture the Italian-held Dodecanese islands in the Aegean Sea following the surrender of Italy in September 1943, and use them as bases against the German-controlled Balkans. The failed campaign, and in particular the Battle of Leros, inspired the 1957 novel **The Guns of Navarone** and the successful 1961 movie of the same name.

Question: American Callan Pinckney's eponymously named system became a best-selling (1980s-2000s) book/video franchise in what genre?
Answer: Fitness
Excerpt: Callan Pinckney was an American fitness professional. She achieved unprecedented success with her Callanetics exercises. Her 9 books all became international best-sellers and the video series that followed went on to sell over 6 million copies. Pinckney's first video release "Callanetics: 10 Years Younger In 10 Hours" outsold every other **fitness** video in the US.

Figure 1: Question-answer pairs with sample excerpts from evidence documents from TriviaQA exhibiting lexical and syntactic variability, and requiring reasoning from multiple sentences.

Recently, significant progress has been made by introducing large new reading comprehension datasets that primarily focus on one of the challenges listed above, for example by crowdsourcing the gathering of question answer pairs (Rajpurkar et al., 2016) or using cloze-style sentences instead of questions (Hermann et al., 2015; Onishi et al., 2016) (see Table 1 for more examples). In general, system performance has improved rapidly as each resource is released. The best models of-

[1]Data and code available at http://nlp.cs.washington.edu/triviaqa/

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1601–1611
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1147

Dataset	Large scale	Freeform Answer	Well formed	Independent of Evidence	Varied Evidence
TriviaQA	✓	✓	✓	✓	✓
SQuAD (Rajpurkar et al., 2016)	✓	✓	✓	✗	✗
MS Marco (Nguyen et al., 2016)	✓	✓	✗	✓	✓
NewsQA(Trischler et al., 2016)	✓	✓	✓	✗*	✗
WikiQA (Yang et al., 2016)	✗	✗	✗	✓	✗
TREC (Voorhees and Tice, 2000)	✗	✓	✓	✓	✓

Table 1: Comparison of TriviaQA with existing QA datasets. Our dataset is unique in that it is naturally occurring, well-formed questions collected independent of the evidences. *NewsQA uses evidence articles indirectly by using only article summaries.

ten achieve near-human performance levels within months or a year, fueling a continual need to build ever more difficult datasets. We argue that TriviaQA is such a dataset, by demonstrating that a high percentage of its questions require solving these challenges and showing that there is a large gap between state-of-the-art methods and human performance levels.

TriviaQA contains over 650K question-answer-evidence triples, that are derived by combining 95K Trivia enthusiast authored question-answer pairs with on average six supporting evidence documents per question. To our knowledge, TriviaQA is the first dataset where full-sentence questions are authored organically (i.e. independently of an NLP task) and evidence documents are collected *retrospectively* from Wikipedia and the Web. This decoupling of question generation from evidence collection allows us to control for potential bias in question style or content, while offering organically generated questions from various topics. Designed to engage humans, TriviaQA presents a new challenge for RC models. They should be able to deal with large amount of text from various sources such as news articles, encyclopedic entries and blog articles, and should handle inference over multiple sentences. For example, our dataset contains three times as many questions that require inference over multiple sentences than the recently released SQuAD (Rajpurkar et al., 2016) dataset. Section 4 present a more detailed discussion of these challenges.

Finally, we present baseline experiments on the TriviaQA dataset, including a linear classifier inspired by work on CNN Dailymail and MCTest (Chen et al., 2016; Richardson et al., 2013) and a state-of-the-art neural network baseline (Seo et al., 2017). The neural model performs best, but only achieves 40% for TriviaQA in comparison to 68% on SQuAD, perhaps due to the challenges listed above. The baseline results also fall far short of human performance levels, 79.7%, suggesting significant room for the future work. In summary, we make the following contributions.

- We collect over 650K question-answer-evidence triples, with questions originating from trivia enthusiasts independent of the evidence documents. A high percentage of the questions are challenging, with substantial syntactic and lexical variability and often requiring multi-sentence reasoning. The dataset and code are available at `http://nlp.cs.washington.edu/triviaqa/`, offering resources for training new reading-comprehension models.
- We present a manual analysis quantifying the quality of the dataset and the challenges involved in solving the task.
- We present experiments with two baseline methods, demonstrating that the TriviaQA tasks are not easily solved and are worthy of future study.
- In addition to the automatically gathered large-scale (but noisy) dataset, we present a clean, human-annotated subset of 1975 question-document-answer triples whose documents are certified to contain all facts required to answer the questions.

2 Overview

Problem Formulation We frame reading comprehension as the problem of answering a question q given the textual evidence provided by document set D. We assume access to a dataset of tuples $\{(q_i, a_i, D_i)|i = 1 \ldots n\}$ where a_i is a text string that defines the correct answer

to question q_i. Following recent formulations (Rajpurkar et al., 2016), we further assume that a_i appears as a substring for some document in the set D_i.[2] However, we differ by setting D_i as a *set* of documents, where previous work assumed a single document (Hermann et al., 2015) or even just a short paragraph (Rajpurkar et al., 2016).

Data and Distant Supervision Our evidence documents are automatically gathered from either Wikipedia or more general Web search results (details in Section 3). Because we gather evidence using an automated process, the documents are not *guaranteed* to contain all facts needed to answer the question. Therefore, they are best seen as a source of *distant supervision*, based on the assumption that the presence of the answer string in an evidence document implies that the document *does* answer the question.[3] Section 4 shows that this assumption is valid over 75% of the time, making evidence documents a strong source of distant supervision for training machine reading systems.

In particular, we consider two types of distant supervision, depending on the source of our documents. For web search results, we expect the documents that contain the correct answer a to be highly redundant, and therefore let each question-answer-document tuple be an independent data point. ($|D_i| = 1$ for all i and $q_i = q_j$ for many i, j pairs). However, in Wikipedia we generally expect most facts to be stated only once, so we instead pool all of the evidence documents and never repeat the same question in the dataset ($|D_i| = 1.8$ on average and $q_i \neq q_j$ for all i, j). In other words, each question (paired with the union of all of its evidence documents) is a single data point.

These are far from the only assumptions that could be made in this distant supervision setup. For example, our data would also support multi-instance learning, which makes the *at least once assumption*, from relation extraction (Riedel et al., 2010; Hoffmann et al., 2011) or many other possibilities. However, the experiments in Section 6 show that these assumptions do present a strong signal for learning; we believe the data will fuel significant future study.

[2]The data we will present in Section 3 would further support a task formulation where some documents D do not have the correct answer and the model must learn when to abstain. We leave this to future work.

[3]An example context for the first question in Figure 1 where such an assumption fails would be the following evidence string: *The Guns of Navarone is a 1961 British-American epic adventure war film directed by J. Lee Thompson.*

Total number of QA pairs	95,956
Number of unique answers	40,478
Number of evidence documents	662,659
Avg. question length (word)	14
Avg. document length (word)	2,895

Table 2: TriviaQA: Dataset statistics.

3 Dataset Collection

We collected a large dataset to support the reading comprehension task described above. First we gathered question-answer pairs from 14 trivia and quiz-league websites. We removed questions with less than four tokens, since these were generally either too simple or too vague.

We then collected textual evidence to answer questions using two sources: documents from Web search results and Wikipedia articles for entities in the question. To collect the former, we posed each question[4] as a search query to the Bing Web search API, and collected the top 50 search result URLs. To exclude the trivia websites, we removed from the results all pages from the trivia websites we scraped and any page whose url included the keywords *trivia*, *question*, or *answer*. We then crawled the top 10 search result Web pages and pruned PDF and other ill formatted documents. The search output includes a diverse set of documents such as blog articles, news articles, and encyclopedic entries.

Wikipedia pages for entities mentioned in the question often provide useful information. We therefore collected an additional set of evidence documents by applying TAGME, an off-the-shelf entity linker (Ferragina and Scaiella, 2010), to find Wikipedia entities mentioned in the question, and added the corresponding pages as evidence documents.

Finally, to support learning from distant supervision, we further filtered the evidence documents to exclude those missing the correct answer string and formed evidence document sets as described in Section 2. This left us with 95K question-answer pairs organized into (1) 650K training examples for the Web search results, each contain-

[4]Note that we did *not* use the answer as a part of the search query to avoid biasing the results.

Property	Example annotation	Statistics
Avg. entities / question	Which politician won the **Nobel Peace Prize** in 2009?	1.77 per question
Fine grained answer type	What **fragrant essential oil** is obtained from Damask Rose?	73.5% of questions
Coarse grained answer type	**Who** won the Nobel Peace Prize in 2009?	15.5% of questions
Time frame	What was photographed for the first time in **October 1959**	34% of questions
Comparisons	What is the appropriate name of the **largest** type of frog?	9% of questions

Table 3: Properties of questions on 200 annotated examples show that a majority of TriviaQA questions contain multiple entities. The boldfaced words hint at the presence of corresponding property.

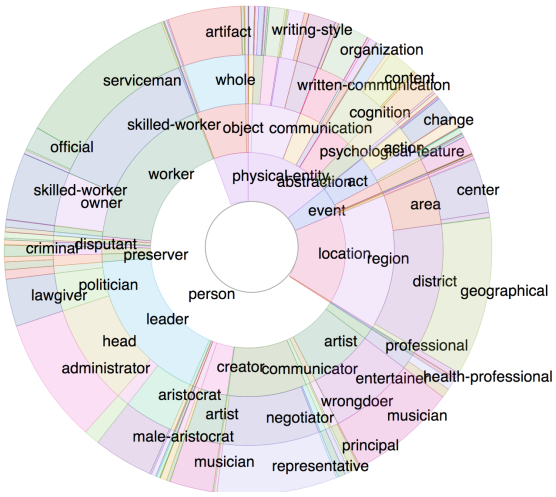


Figure 2: Distribution of hierarchical WordNet synsets for entities appearing in the answer. The arc length is proportional to the number of questions containing that category.

ing a single (combined) evidence document, and (2) 78K examples for the Wikipedia reading comprehension domain, containing on average 1.8 evidence documents per example. Table 2 contains the dataset statistics. While not the focus of this paper, we have also released the full unfiltered dataset which contains 110,495 QA pairs and 740K evidence documents to support research in allied problems such as open domain and IR-style question answering.

4 Dataset Analysis

A quantitative and qualitative analysis of TriviaQA shows it contains complex questions about a diverse set of entities, which are answerable using the evidence documents.

Question and answer analysis TriviaQA questions, authored by trivia enthusiasts, cover various topics of people's interest. The average question length is 14 tokens indicating that many questions are highly compositional. For qualitative analysis, we sampled 200 question answer pairs and manually analysed their properties. About 73.5% of these questions contain phrases that describe a fine grained category to which the answer belongs, while 15.5% hint at a coarse grained category (one of *person*, *organization*, *location*, and *miscellaneous*). Questions often involve reasoning over time frames, as well as making comparisons. A summary of the analysis is presented in Table 3.

Type	Percentage
Numerical	4.17
Free text	2.98
Wikipedia title	92.85
Person	32
Location	23
Organization	5
Misc.	40

Table 4: Distribution of answer types on 200 annotated examples.

Answers in TriviaQA belong to a diverse set of types. 92.85% of the answers are titles in Wikipedia,[5] 4.17% are numerical expressions (e.g., 9 kilometres) while the rest are open ended noun and verb phrases. A coarse grained type analysis of answers that are Wikipedia entities presented in Table 4. It should be noted that not all Wikipedia titles are named entities; many are common phrases such as *barber* or *soup*. Figure 2 shows diverse topics indicated by WordNet synsets of answer entities.

Evidence analysis A qualitative analysis of TriviaQA shows that the evidence contains answers for 79.7% and 75.4% of questions from the Wikipedia and Web domains respectively. To analyse the quality of evidence and evaluate baselines, we asked a human annotator to answer 986 and 1345 (dev and test set) questions from the Wikipedia and Web domains respectively. Trivia

[5]This is a very large set since Wikipedia has more than 11 million titles.

Reasoning	**Lexical variation (synonym)**
	Major correspondences between the question and the answer sentence are synonyms.
Frequency	41% in Wiki documents, 39% in web documents.
Examples	Q What is solid CO2 commonly called?
	S The frozen solid form of CO2, known as **dry ice** ...
	Q Who wrote the novel The Eagle Has landed?
	S The Eagle Has Landed is a book by British writer **Jack Higgins**
Reasoning	**Lexical variation and world knowledge**
	Major correspondences between the question and the document require common sense or external knowledge.
Frequency	17% in Wiki documents, 17% in web documents.
Examples	Q What is the first name of Madame Bovary in Flaubert's 1856 novel?
	S Madame Bovary (1856) is the French writer Gustave Flaubert's debut novel. The story focuses on a doctor's wife, **Emma** Bovary
	Q Who was the female member of the 1980's pop music duo, Eurythmics?
	S Eurythmics were a British music duo consisting of members **Annie Lennox** and David A. Stewart.
Reasoning	**Syntactic Variation**
	After the question is paraphrased into declarative form, its syntactic dependency structure does not match that of the answer sentence
Frequency	69% in Wiki documents, 65% in web documents.
Examples	Q In which country did the Battle of El Alamein take place?
	S The 1942 Battle of El Alamein in **Egypt** was actually two pivotal battles of World War II
	Q Whom was Ronald Reagan referring to when he uttered the famous phrase evil empire in a 1983 speech?
	S The phrase evil empire was first applied to the **Soviet Union** in 1983 by U.S. President Ronald Reagan.
Reasoning	**Multiple sentences**
	Requires reasoning over multiple sentences.
Frequency	40% in Wiki documents, 35% in web documents.
Examples	Q Name the Greek Mythological hero who killed the gorgon Medusa.
	S **Perseus** asks god to aid him. So the goddess Athena and Hermes helps him out to kill Medusa.
	Q Who starred in and directed the 1993 film A Bronx Tale?
	S **Robert De Niro** To Make His Broadway Directorial Debut With A Bronx Tale: The Musical. The actor starred and directed the 1993 film.
Reasoning	**Lists, Table**
	Answer found in tables or lists
Frequency	7% in web documents.
Examples	Q In Moh's Scale of hardness, Talc is at number 1, but what is number 2?
	Q What is the collective name for a group of hawks or falcons?

Table 5: Analysis of reasoning used to answer TriviaQA questions shows that a high proportion of evidence sentence(s) exhibit syntactic and lexical variation with respect to questions. Answers are indicated by boldfaced text.

questions contain multiple clues about the answer(s) not all of which are referenced in the documents. The annotator was asked to answer a question if the minimal set of facts (ignoring temporal references like *this year*) required to answer the question are present in the document, and abstain otherwise. For example, it is possible to answer the question, *Who became president of the Mormons in 1844, organised settlement of the Mormons in Utah 1847 and founded Salt Lake City?* using only the fact that Salt Lake City was founded by Brigham Young. We found that the accuracy (evaluated using the original answers) for the Wikipedia and Web domains was 79.6 and 75.3 respectively. We use the correctly answered questions (and documents) as verified sets for evaluation (section 6).

Challenging problem A comparison of evidence with respect to the questions shows a high proportion of questions require reasoning over multiple sentences. To compare our dataset against previous datasets, we classified 100 question-evidence pairs each from Wikipedia and the Web according to the form of reasoning required to answer them. We focus the analysis on Wikipedia since the analysis on Web documents are similar. Categories are not mutually exclusive: single example can fall into multiple categories. A summary of the analysis is presented in Table 5.

On comparing evidence sentences with their corresponding questions, we found that 69% of the questions had a different syntactic structure while 41% were lexically different. For 40% of the questions, we found that the information re-

quired to answer them was scattered over multiple sentences. Compared to SQuAD, over three times as many questions in TriviaQA require reasoning over multiple sentences. Moreover, 17% of the examples required some form of world knowledge. Question-evidence pairs in TriviaQA display more lexical and syntactic variance than SQuAD. This supports our earlier assertion that decoupling question generation from evidence collection results in a more challenging problem.

5 Baseline methods

To quantify the difficulty level of the dataset for current methods, we present results on neural and other models. We used a random entity baseline and a simple classifier inspired from previous work (Wang et al., 2015; Chen et al., 2016), and compare these to BiDAF (Seo et al., 2017), one of the best performing models for the SQuAD dataset.

5.1 Random entity baseline

We developed the random entity baseline for the Wikipedia domain since the provided documents can be directly mapped to candidate answers. In this heuristic approach, we first construct a candidate answer set using the entities associated with the provided Wikipedia pages for a given question (on average 1.8 per question). We then randomly pick a candidate that does not occur in the question. If no such candidate exists, we pick any random candidate from the candidate set.

5.2 Entity classifier

We also frame the task as a ranking problem over candidate answers in the documents. More formally, given a question q_i, an answer a_i^+, and a evidence document D_i, we want to learn a scoring function $score$, such that

$$score(a_i^+|q_i, D_i) > score(a_i^-|q_i, D_i)$$

where a_i^- is any candidate other than the answer. The function $score$ is learnt using LambdaMART (Wu et al., 2010),[6] a boosted tree based ranking algorithm.

This is similar to previous entity-centric classifiers for QA (Chen et al., 2016; Wang et al., 2015), and uses context and Wikipedia catalog based features. To construct the candidate answer set, we consider sentences that contain at least one word in common with the question. We then add every n-gram ($n \in [1, 5]$) that occurs in these sentences and is a title of some Wikipedia article.[7]

5.3 Neural model

Recurrent neural network models (RNNs) (Hermann et al., 2015; Chen et al., 2016) have been very effective for reading comprehension. For our task, we modified the BiDAF model (Seo et al., 2017), which takes a sequence of context words as input and outputs the start and end positions of the predicted answer in the context. The model utilizes an RNN at the character level, token level, and phrase level to encode context and question and uses attention mechanism between question and context.

Authored independently from the evidence document, TriviaQA does not contain the exact spans of the answers. We approximate the answer span by finding the first match of answer string in the evidence document. Developed for a dataset where the evidence document is a single paragraph (average 122 words), the BiDAF model does not scale to long documents. To overcome this, we truncate the evidence document to the first 800 words.[8]

When the data contains more than one evidence document, as in our Wikipedia domain, we predict for each document separately and aggregate the predictions by taking a sum of confidence scores. More specifically, when the model outputs a candidate answer A_i from n documents $D_{i,1}, ...D_{i,n}$ with confidences $c_{i,1}, ...c_{i,n}$, the score of A_i is given by

$$score(A_i) = \sum_k c_{i,k}$$

We select candidate answer with the highest score.

6 Experiments

An evaluation of our baselines shows that both of our tasks are challenging, and that the TriviaQA dataset supports significant future work.

[6] We use the RankLib implementation https://sourceforge.net/p/lemur/wiki/RankLib/

[7] Using a named entity recognition system to generate candidate entities is not feasible as answers can be common nouns or phrases.

[8] We found that splitting documents into smaller sub documents degrades performance since a majority of sub documents do not contain the answer.

		Train	Dev	Test
Wikipedia	Questions	61,888	7,993	7,701
	Documents	110,648	14,229	13,661
Web	Questions	76,496	9,951	9,509
	Documents	528,979	68,621	65,059
Wikipedia verified	Questions	-	297	584
	Documents	-	305	592
Web verified	Questions	-	322	733
	Documents	-	325	769

Table 6: Data statistics for each task setup. The Wikipedia domain is evaluated over questions while the web domain is evaluated over documents.

6.1 Evaluation Metrics

We use the same evaluation metrics as SQuAD – exact match (EM) and F1 over words in the answer(s). For questions that have *Numerical* and *FreeForm* answers, we use a single given answer as ground truth. For questions that have Wikipedia entities as answers, we use Wikipedia aliases as valid answer along with the given answer.

Since Wikipedia and the web are vastly different in terms of style and content, we report performance on each source separately. While using Wikipedia, we evaluate at the question level since facts needed to answer a question are generally stated only once. On the other hand, due to high information redundancy in web documents (around 6 documents per question), we report document level accuracy and F1 when evaluating on web documents. Lastly, in addition to distant supervision, we also report evaluation on the clean dev and test questions collection using a human annotator (section 4)

6.2 Experimental Setup

We randomly partition QA pairs in the dataset into train (80%), development (10%), and test set (10%). In addition to distant supervision evaluation, we also evaluate baselines on verified subsets (see section 4) of the dev and test partitions. Table 6 contains the number of questions and documents for each task. We trained the entity classifier on a random sample of 50,000 questions from the training set. For training BiDAF on the web domain, we first randomly sampled 80,000 documents. For both domains, we used only those (training) documents where the answer appears in the first 400 tokens to keep training time manageable. Designing scalable techniques that can use the entirety of the data is an interesting direction for future work.

6.3 Results

The performance of the proposed models is summarized in Table 7. The poor performance of the random entity baseline shows that the task is not already solved by information retrieval. For both Wikipedia and web documents, BiDAF (40%) outperforms the classifier (23%). The oracle score is the upper bound on the exact match accuracy.[9] All models lag significantly behind the human baseline of 79.7% on the Wikipedia domain, and 75.4% on the web domain.

We analyse the performance of BiDAF on the development set using Wikipedia as the evidence source by question length and answer type. The accuracy of the system steadily decreased as the length of the questions increased – with 50% for questions with 5 or fewer words to 32% for 20 or more words. This suggests that longer compositional questions are harder for current methods.

6.4 Error analysis

Our qualitative error analysis reveals that compositionality in questions and lexical variation and low signal-to-noise ratio in (full) documents is still a challenge for current methods. We randomly sampled 100 incorrect BiDAF predictions from the development set and used Wikipedia evidence documents for manual analysis. We found that 19 examples lacked evidence in any of the provided documents, 3 had incorrect ground truth, and 3 were valid answers that were not included in the answer key. Furthermore, 12 predictions were partially correct (*Napoleonic* vs *Napoleonic Wars*). This seems to be consistent with human performance of 79.7%.

For the rest, we classified each example into one or more categories listed in Table 8. Distractor entities refers to the presence of entities similar to ground truth. E.g., for the question, *Rebecca Front plays Detective Chief Superintendent Innocent in which TV series?*, the evidence describes all roles played by Rebecca Front.

The first two rows suggest that long and noisy documents make the question answering task more difficult, as compared for example to the short passages in SQuAD. Furthermore, a high proportion of errors are caused by paraphrasing, and the answer is sometimes stated indirectly. For

[9] A question q is considered answerable for the oracle score if the correct answer is found in the evidence D or, in case of the classifier, is a part of the candidate set. Since we truncate documents, the upper bound is not 100%.

Method	Domain	Distant Supervision						Verified					
		Dev			Test			Dev			Test		
		EM	F1	Oracle	EM	F1	Oracle	EM	F1	Oracle	EM	F1	Oracle
Random	Wiki	12.72	22.91	16.30	12.74	22.35	16.28	14.81	23.31	19.53	15.41	25.44	19.19
Classifier		23.42	27.68	71.41	22.45	26.52	71.67	24.91	29.43	80.13	27.23	31.37	77.74
BiDAF		**40.26**	**45.74**	**82.55**	**40.32**	**45.91**	**82.82**	**47.47**	**53.70**	**90.23**	**44.86**	**50.71**	**86.81**
Classifier	web	24.64	29.08	66.78	24.00	28.38	66.35	27.38	31.91	77.23	30.17	34.67	76.72
BiDAF		**41.08**	**47.40**	**82.93**	**40.74**	**47.05**	**82.95**	**51.38**	**55.47**	**90.46**	**49.54**	**55.80**	**89.99**

Table 7: Performance of all systems on TriviaQA using distantly supervised evaluation. The best performing system is indicated in bold.

Category	Proportion
Insufficient evidence	19
Prediction from incorrect document(s)	7
Answer not in clipped document	15
Paraphrasing	29
Distractor entities	11
Reasoning over multiple sentences	18

Table 8: Qualitative error analysis of BiDAF on Wikipedia evidence documents.

example, the evidence for the question *What was Truman Capote's last name before he was adopted by his stepfather?* consists of the following text *Truman Garcia Capote born Truman Streckfus Persons, was an American ... In 1933, he moved to New York City to live with his mother and her second husband, Joseph Capote, who adopted him as his stepson and renamed him Truman Garca Capote.*

7 Related work

Recent interest in question answering has resulted in the creation of several datasets. However, they are either limited in scale or suffer from biases stemming from their construction process. We group existing datasets according to their associated tasks, and compare them against TriviaQA. The analysis is summarized in Table 1.

7.1 Reading comprehension

Reading comprehension tasks aims to test the ability of a system to understand a document using questions based upon its contents. Researchers have constructed cloze-style datasets (Hill et al., 2015; Hermann et al., 2015; Paperno et al., 2016; Onishi et al., 2016), where the task is to predict missing words, often entities, in a document. Cloze-style datasets, while easier to construct large-scale automatically, do not contain natural language questions.

Datasets with natural language questions include MCTest (Richardson et al., 2013), SQuAD (Rajpurkar et al., 2016), and NewsQA (Trischler et al., 2016). MCTest is limited in scale with only 2640 multiple choice questions. SQuAD contains 100K crowdsourced questions and answers paired with short Wikipedia passages. NewsQA uses crowdsourcing to create questions solely from news article summaries in order to control potential bias. The crucial difference between SQuAD/NewsQA and TriviaQA is that TriviaQA questions have not been crowdsourced from preselected passages. Additionally, our evidence set consists of web documents, while SQuAD and NewsQA are limited to Wikipedia and news articles respectively. Other recently released datasets include (Lai et al., 2017).

7.2 Open domain question answering

The recently released MS Marco dataset (Nguyen et al., 2016) also contains independently authored questions and documents drawn from the search results. However, the questions in the dataset are derived from search logs and the answers are crowdsourced. On the other hand, trivia enthusiasts provided both questions and answers for our dataset.

Knowledge base question answering involves converting natural language questions to logical forms that can be executed over a KB. Proposed datasets (Cai and Yates, 2013; Berant et al., 2013; Bordes et al., 2015) are either limited in scale or in the complexity of questions, and can only retrieve facts covered by the KB.

A standard task for open domain IR-style QA is the annual TREC competitions (Voorhees and Tice, 2000), which contains questions from various domains but is limited in size. Many advances from the TREC competitions were used in the IBM Watson system for *Jeopardy!* (Ferrucci et al., 2010). Other datasets includes SearchQA

(Dunn et al., 2017) where *Jeopardy!* questions are paired with search engine snippets, the WikiQA dataset (Yang et al., 2015) for answer sentence selection, and the Chinese language WebQA (Li et al., 2016) dataset, which focuses on the task of answer phrase extraction. TriviaQA contains examples that could be used for both stages of the pipeline, although our focus on this paper is instead on using the data for reading comprehension where the answer is always present.

Other recent approaches attempt to combine structured high precision KBs with semi-structured information sources like OpenIE triples (Fader et al., 2014), HTML tables (Pasupat and Liang, 2015), and large (and noisy) corpora (Sawant and Chakrabarti, 2013; Joshi et al., 2014; Xu et al., 2015). TriviaQA, which has Wikipedia entities as answers, makes it possible to leverage structured KBs like Freebase, which we leave to future work. Furthermore, about 7% of the TriviaQA questions have answers in HTML tables and lists, which could be used to augment these existing resources.

Trivia questions from quiz bowl have been previously used in other question answering tasks (Boyd-Graber et al., 2012). Quiz bowl questions are paragraph length and pyramidal.[10] A number of different aspects of this problem have been carefully studied, typically using classifiers over a pre-defined set of answers (Iyyer et al., 2014) and studying incremental answering to answer as quickly as possible (Boyd-Graber et al., 2012) or using reinforcement learning to model opponent behavior (He et al., 2016). These competitive challenges are not present in our single-sentence question setting. Developing joint models for multi-sentence reasoning for questions and answer documents is an important area for future work.

8 Conclusion and Future Work

We present TriviaQA, a new dataset of 650K question-document-evidence triples. To our knowledge, TriviaQA is the first dataset where questions are authored by trivia enthusiasts, independently of the evidence documents. The evidence documents come from two domains – Web search results and Wikipedia pages – with highly differing levels of information redundancy. Results from current state-of-the-art baselines indicate that TriviaQA is a challenging testbed that deserves significant future study.

While not the focus of this paper, TriviaQA also provides a provides a benchmark for a variety of other tasks such as IR-style question answering, QA over structured KBs and joint modeling of KBs and text, with much more data than previously available.

[10] Pyramidal questions consist of a series of clues about the answer arranged in order from most to least difficult.

Acknowledgments

This work was supported by DARPA contract FA8750-13-2-0019, the WRF/Cable Professorship, gifts from Google and Tencent, and an Allen Distinguished Investigator Award. The authors would like to thank Minjoon Seo for the BiDAF code, and Noah Smith, Srinivasan Iyer, Mark Yatskar, Nicholas FitzGerald, Antoine Bosselut, Dallas Card, and anonymous reviewers for helpful comments.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1533–1544. http://aclweb.org/anthology/D/D13/D13-1160.pdf.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR* abs/1506.02075. https://arxiv.org/abs/1506.02075.

Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daumé III. 2012. Besting the quiz master: Crowdsourcing incremental classification games. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, pages 1290–1301. http://www.aclweb.org/anthology/D12-1118.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 423–433. http://www.aclweb.org/anthology/P13-1042.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the

cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2358–2367. http://www.aclweb.org/anthology/P16-1223.

Matthew Dunn, Levent Sagun, Mike Higgins, Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *CoRR* https://arxiv.org/abs/1704.05179.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '14, pages 1156–1165. https://doi.org/10.1145/2623330.2623677.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, USA, CIKM '10, pages 1625–1628. https://doi.org/10.1145/1871437.1871689.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the deepqa project. *AI MAGAZINE* 31(3):59–79.

He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. 2016. Opponent modeling in deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, New York, New York, USA, volume 48 of *Proceedings of Machine Learning Research*, pages 1804–1813. http://proceedings.mlr.press/v48/he16.html.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. http://arxiv.org/abs/1506.03340.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *CoRR* https://arxiv.org/abs/1511.02301.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 541–550. http://www.aclweb.org/anthology/P11-1055.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 633–644. http://www.aclweb.org/anthology/D14-1070.

Mandar Joshi, Uma Sawant, and Soumen Chakrabarti. 2014. Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1104–1114. http://www.aclweb.org/anthology/D14-1117.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *CoRR* https://arxiv.org/abs/1704.04683.

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. 2016. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *CoRR* https://arxiv.org/abs/1607.06275.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Workshop in Advances in Neural Information Processing Systems*. https://arxiv.org/pdf/1611.09268.pdf.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2230–2235. https://aclweb.org/anthology/D16-1241.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1525–1534. http://www.aclweb.org/anthology/P16-1144.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the*

Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers. pages 1470–1480. http://aclweb.org/anthology/P/P15/P15-1142.pdf.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 193–203. http://www.aclweb.org/anthology/D13-1020.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*. Springer-Verlag, Berlin, Heidelberg, ECML PKDD'10, pages 148–163. http://dl.acm.org/citation.cfm?id=1889788.1889799.

Uma Sawant and Soumen Chakrabarti. 2013. Learning joint query interpretation and response ranking. In *Proceedings of the 22Nd International Conference on World Wide Web*. ACM, New York, NY, USA, WWW '13, pages 1099–1110. https://doi.org/10.1145/2488388.2488484.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1611.01603.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *CoRR* https://arxiv.org/abs/1611.09830.

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '00, pages 200–207. https://doi.org/10.1145/345508.345577.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 700–706. http://www.aclweb.org/anthology/P15-2115.

Qiang Wu, Christopher J. Burges, Krysta M. Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Inf. Retr.* 13(3):254–270. https://doi.org/10.1007/s10791-009-9112-1.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*. https://arxiv.org/abs/1502.03044.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2013–2018. http://aclweb.org/anthology/D15-1237.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1480–1489. http://www.aclweb.org/anthology/N16-1174.

Learning Semantic Correspondences in Technical Documentation

Kyle Richardson and **Jonas Kuhn**
Institute of Natural Language Processing
University of Stuttgart
{kyle,jonas}@ims.uni-stuttgart.de

Abstract

We consider the problem of translating high-level textual descriptions to formal representations in technical documentation as part of an effort to model the meaning of such documentation. We focus specifically on the problem of learning translational correspondences between text descriptions and grounded representations in the target documentation, such as formal representation of functions or code templates. Our approach exploits the parallel nature of such documentation, or the tight coupling between high-level text and the low-level representations we aim to learn. Data is collected by mining technical documents for such parallel text-representation pairs, which we use to train a simple semantic parsing model. We report new baseline results on sixteen novel datasets, including the standard library documentation for nine popular programming languages across seven natural languages, and a small collection of Unix utility manuals.

1 Introduction

Technical documentation in the computer domain, such as source code documentation and other how-to manuals, provide high-level descriptions of how lower-level computer programs and utilities work. Often these descriptions are coupled with formal representations of these lower-level features, expressed in the target programming languages. For example, Figure 1.1 shows the source code documentation (in red/bold) for the `max` function in the Java programming language paired with the representation of this function in the underlying Java language (in black). This formal representation captures the name of the function, the return value, the types of arguments the function takes, among other details related to the function's place and visibility in the overall source code collection or API.

1. Java Documentation

```
*Returns the greater of two long values
*
* @param a an argument
* @param b another argument
* @return the larger of a and b
* @see java.lang.Long#MAX_VALUE
*/
 public static long max(long a, long b)
```

2. Clojure Documentation

```
(defn random-sample
  "Returns items from coll with random
  probability of prob (0.0 - 1.0)"
  ([prob coll] ...))
```

3. PHP documentation (French)

```
Ajoute une valeur comme dernier élément
*
* @param value La valeur á ajouter
* @see ArrayIterations::next()
*/
public void append(mixed $value)
```

Figure 1: Example source code documentation.

Given the high-level nature of the textual annotations, modeling the meaning of any given description is not an easy task, as it involves much more information than what is directly provided in the associated documentation. For example, capturing the meaning of the description *the greater of* might require having a background theory about quantity/numbers and relations between different quantities. A first step towards capturing the meaning, however, is learning to translate this description to symbols in the target representation, in this case to the `max` symbol. By doing this translation to a formal language, modeling and learning the subsequent semantics becomes easier since we are eliminating the ambiguity of ordinary lan-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1612–1622
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1148

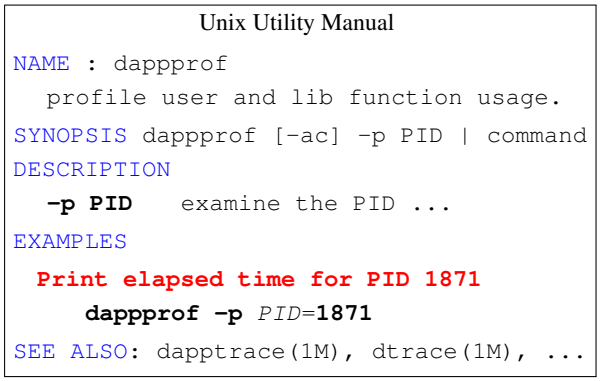


Figure 2: An example computer utility manual in the Unix domain. Descriptions of example uses are shown in red.

guage. Similarly, we would want to first translate the description *two long values*, which specifies the number and type of argument taken by this function, to the sequence `long a,long b`.

By focusing on translation, we can create new datasets by mining these types of source code collections for sets of parallel text-representation pairs. Given the wide variety of available programming languages, many such datasets can be constructed, each offering new challenges related to differences in the formal representations used by different programming languages. Figure 1.2 shows example documentation for the Clojure programming language, which is part of the Lisp family of languages. In this case, the description *Returns random probability of* should be translated to the function name `random-sample` since it describes what the overall function does. Similarly, the argument descriptions *from coll* and *of prob* should translate to `coll` and `prob`.

Given the large community of programmers around the world, many source code collections are available in languages other than English. Figure 1.3 shows an example entry from the French version of the PHP standard library, which was translated by volunteer developers. Having multilingual data raises new challenges, and broadens the scope of investigations into this type of semantic translation.

Other types of technical documentation, such as utility manuals, exhibit similar features. Figure 2 shows an example manual in the domain of Unix utilities. The textual description in red/bold describes an example use of the *dappprof* utility paired with formal representations in the form of executable code. As with the previous examples, such formal representations do not capture the full meaning of the different descriptions, but serve as a convenient operationalization, or *translational semantics*, of the meaning in Unix. *Print elapsed time*, for example, roughly describes what the `dappprof` utility does, whereas *PID 1871* describes the second half of the code sequence.

In both types of technical documentation, information is not limited to raw pairs of descriptions and representations, but can include other information and clues that are useful for learning. Java function annotations include textual descriptions of individual arguments and return values (shown in green). Taxonomic information and pointers to related functions or utilities are also annotated (e.g., the `@see` section in Figure 1, or `SEE ALSO` section in Figure 2). Structural information about code sequences, and the types of abstract arguments these sequences take, are described in the `SYNOPSIS` section of the Unix manual. This last piece of information allows us to generate abstract code templates, and generalize individual arguments. For example, the raw argument `1871` in the sequence `dappprof -p 1871` can be typed as a `PID` instance, and an argument of the `-p` flag.

Given this type of data, a natural experiment is to see whether we can build programs that translate high-level textual descriptions to correct formal representations. We aim to learn these translations using raw text-meaning pairs as the sole supervision. Our focus is on learning function translations or representations within nine programming language APIs, each varying in size, representation style, and source natural language. To our knowledge, our work is the first to look at translating source code descriptions to formal representations using such a wide variety of programming and natural languages. In total, we introduce fourteen new datasets in the source code domain that include seven natural languages, and report new results for an existing dataset. As well, we look at learning simple code templates using a small collection of English Unix manuals.

The main goal of this paper is to establish strong baselines results on these resources, which we hope can be used for benchmarking and developing new semantic parsing methods. We achieved initial baselines using the language modeling and translation approach of Deng and Chrupała (2014). We also show that modest improvements can be achieved by using a more conventional

discriminative model (Zettlemoyer and Collins, 2009) that, in part, exploits document-level features from the technical documentation sets.

2 Related Work

Our work is situated within research on semantic parsing, which focuses on the problem of generating formal meaning representations from text for natural language understanding applications. Recent interest in this topic has centered around learning meaning representation from example text-meaning pairs, for applications such as automated question-answering (Berant et al., 2013), robot control (Matuszek et al., 2012) and text generation (Wong and Mooney, 2007a).

While generating representations for natural language understanding is a complex task, most studies focus on the translation or generation problem independently of other semantic or knowledge representation issues. Earlier work looks at supervised learning of logical representations using example text-meaning pairs using tools from statistical machine translation (Wong and Mooney, 2006) and parsing (Zettlemoyer and Collins, 2009). These methods are meant to be applicable to a wide range of translation problems and representation types, which make new parallel datasets or resources useful for furthering the research.

In general, however, such datasets are hard to construct since building them requires considerable domain knowledge and knowledge of logic. Alternatively, we construct parallel datasets automatically from technical documentation, which obviates the need for annotation. While the formal representations are not actual logical forms, they still provide a good test case for testing how well semantic parsers learn translations to representations.

To date, most benchmark datasets are limited to small controlled domains, such as geography and navigation. While attempts have been made to do open-domain semantic parsing using larger, more complex datasets (Berant et al., 2013; Pasupat and Liang, 2015), such resources are still scarce. In Figure 3, we compare the details of one widely used dataset, Geoquery (Zelle and Mooney, 1996), to our new datasets. Our new resources are on average much larger than geoquery in terms of the number of example pairs, and the size of the different language vocabularies. Most existing datasets are also primarily English-based, while we focus on learning in a multilingual setting using several new moderately sized datasets.

Within semantic parsing, there has also been work on situated or grounded learning, that involves learning in domains with weak supervision and indirect cues (Liang, 2016; Richardson and Kuhn, 2016). This has sometimes involved learning from automatically generated parallel data and representations (Chen and Mooney, 2008) of the type we consider in this paper. Here one can find work in technical domains, including learning to generate regular expressions (Manshadi et al., 2013; Kushman and Barzilay, 2013) and other types of source code (Quirk et al., 2015), which ultimately aim to solve the problem of natural language programming. We view our work as one small step in this general direction.

Our work is also related to software components retrieval and builds on the approach of Deng and Chrupała (2014). Robustly learning the translation from language to code representations can help to facilitate natural language querying of API collections (Lv et al., 2015). As part of this effort, recent work in machine learning has focused on the similar problem of learning code representations using resources such as StackOverflow and Github. These studies primarily focus on learning longer programs (Allamanis et al., 2015) as opposed to function representations, or focus narrowly on a single programming language such as Java (Gu et al., 2016) or on related tasks such as text generation (Iyer et al., 2016; Oda et al., 2015). To our knowledge, none of this work has been applied to languages other than English or such a wide variety of programming languages.

3 Mapping Text to Representations

In this section, we formulate the basic problem of translating to representations in technical documentation.

3.1 Problem Description

We use the term *technical documentation* to refer to two types of resources: textual descriptions inside of source code collections, and computer utility manuals. In this paper, the first type includes high-level descriptions of functions in standard library source code documentation. The second type includes a collection of Unix manuals, also known as man pages. Both types include pairs of text and code representations.

Dataset	#Pairs	#Descr.	Symbols	#Words	Vocab.	Example Pairs (x, z), **Goal:** learn a function $x \rightarrow z$
Java	7,183	4,804	4,072	82,696	3,721	x : Compares this Calendar to the specified Object. z : `boolean util.Calendar.equals(Object obj)`
Ruby	6,885	1,849	3,803	67,274	5,131	x : Computes the arc tangent given y and x. z : `Math.atan2(y,x) → Float`
PHP$_{en}$	6,611	13,943	8,308	68,921	4,874	x : Delete an entry in the archive using its name. z : `bool ZipArchive::deleteName(string $name)`
Python	3,085	429	3,991	27,012	2,768	x : Remove the specific filter from this handler. z : `logging.Filterer.removeFilter(filter)`
Elisp	2,089	1,365	1,883	30,248	2,644	x : This function returns the total height, in lines, of the window. z : `(window-total-height window round)`
Haskell	1,633	255	1,604	19,242	2,192	x : Extract the second component of a pair. z : `Data.Tuple.snd :: (a, b) -> b`
Clojure	1,739	–	2,569	17,568	2,233	x : Returns a lazy seq of every nth item in coll. z : `(core.take-nth n coll)`
C	1,436	1,478	1,452	12,811	1,835	x : Returns the current file position of the stream stream. z : `long int ftell(FILE *stream)`
Scheme	1,301	376	1,343	15,574	1,756	x : Returns a new port with type port-type and the given state. z : `(make-port port-type state)`
Unix	921	940	1,000	11,100	2,025	x : To get policies for a specific user account. z : `pwpolicy -u username -getpolicy`
Geoquery	880	–	167	6,663	279	x : What is the tallest mountain in America? z : `(highest(mountain(loc_2(countryid usa))))`

Figure 3: Description of our English corpus collection with example text/function pairs.

We will refer to the target representations in these resources as *API components*, or components. In source code, components are formal representations of functions, or *function signatures* (Deng and Chrupała, 2014). The form of a function signature varies depending on the resource, but in general gives a specification of how a function is named and structured. The example function signatures in Figure 3 all specify a function name, a list of arguments, and other optional information such as a return value and a namespace. Components in utility manuals are short executable code sequences intended to show an example use of a utility. We assume typed code sequences following Richardson and Kuhn (2014), where the constituent parts of the sequences are abstracted by type.

Given a set of example text-component pairs, $D = \{(x_i, z_i)\}_{i=1}^{n}$, the goal is to learn how to generate correct, well-formed components $z \in \mathcal{C}$ for each input x. Viewed as a semantic parsing problem, this treats the target components as a kind of formal meaning representation, analogous to a logical form. In our experiments, we assume that the complete set of output components are known. In the API documentation sets, this is because each standard library contains a finite number of function representations, roughly corresponding to the number of pairs as shown in Figure 3. For a given input, therefore, the goal is to find the best candidate function translation within the space of the total API components $\mathcal{C}$ (Deng and Chrupała, 2014).

Given these constraints, our setup closely resembles that of Kushman et al. (2014), who learn to parse algebra word problems using a small set of equation templates. Their approach is inspired by template-based information extraction, where templates are recognized and instantiated by slot-filling. Our function signatures and code templates have a similar slot-like structure, consisting of slots such as return value, arguments, function name and namespace.

3.2 Language Modeling Baselines

Existing approaches to semantic parsing formalize the mapping from language to logic using a variety of formalisms including CFGs (Börschinger et al., 2011), CCGs (Kwiatkowski et al., 2010), synchronous CFGs (Wong and Mooney, 2007b). Deciding to use one formalism over another is often motivated by the complexities of the target representations being learned. For example, recent interest in learning graph-based representations such as those in the AMR bank (Banarescu et al., 2013)

requires parsing models that can generate complex graph shaped derivations such as CCGs (Artzi et al., 2015) or HRGs (Peng et al., 2015). Given the simplicity of our API representations, we opt for a simple semantic parsing model that exploits the finiteness of our target representations.

Following ((Deng and Chrupała, 2014); henceforth DC), we treat the problem of component translation as a language modeling problem (Song and Croft, 1999). For a given *query* sequence or text $x = w_i, .., w_I$ and *component* sequence $z = u_j, .., u_J$, the probability of the component given the query is defined as follows using Bayes' theorem: $p(z|x) \propto p(x|z)p(z)$.

By assuming a uniform prior over the probability of each component $p(z)$, the problem reduces to computing $p(x|z)$, which is where language modeling is used. Given each word w_i in the query, a unigram model is defined as $p(x|z) = \prod_{i=1}^{I} p(w_i|z)$. Using this formulation, we can then define different models to estimate $p(w|z)$.

Term Matching As a baseline for $p(w|z)$, DC define a *term matching* approach that exploits the fact that many queries in our English datasets share vocabulary with target component vocabulary. A smoothed version of this baseline is defined below, where $f(w|z)$ is the frequency of matching terms in the target signature, $f(w|\mathcal{C})$ is frequency of the term word in the overall documentation collection, and λ is a smoothing parameter (for Jelinek-Mercer smoothing):

$$p(x|z) = \prod_{w \in x} (1-\lambda) f(w|z) + \lambda f(w|\mathcal{C})$$

Translation Model In order to account for the co-occurrence between non-matching words and component terms, DC employ a word-based translation model, which models the relation between natural language words w_j and individual component terms u_j. In this paper, we limit ourselves to sequence-based word alignment models (Och and Ney, 2003), which factor in the following manner:

$$p(x|z) = \prod_{i=1}^{I} \sum_{j=0}^{J} p_t(w_i|u_j) p_d(l(j)|i, I, J)$$

Here each $p_t(w_i|u_j)$ defines an (unsmoothed) multinomial distribution over a given component term u_j for all words w_j. The function p_d is a distortion parameter, and defines a dependency between the alignment positions and the lengths of both input strings. This function, and the definition of $l(j)$, assumes different forms according to the particular alignment model being used. We consider three different types of alignment models each defined in the following way:

$$p_d(l(j)|...) = \begin{cases} \frac{1}{J+1} & (1) \\ a(j|i, I, J) & (2) \\ a(t(j)|i, I, tlen(J)) & (3) \end{cases}$$

Models (1-2) are the classic IBM word-alignment models of Brown et al. (1993). IBM Model 1, for example, assumes a uniform distribution over all positions, and is the main model investigated in DC. For comparison, we also experiment with IBM Model 2, where each $l(j)$ refers to the string position of j in the component input, and $a(..)$ defines a multinomial distribution such that $\sum_{j=0}^{J} a(j|i, I, J) = 1.0$.

We also define a new tree based alignment model (3) that takes into account the syntax associated with the function representations. Each $l(j)$ is the relative tree position of the alignment point, shown as $t(j)$, and $tlen(J)$ is the length of the tree associated with z. This approach assumes a tree representation for each z. We generated these trees heuristically by preserving the information that is lost when components are converted to a linear sequence representation. An example structure for PHP is shown in Figure 4, where the red solid line indicates the types of potential errors avoided by this model.

Learning is done by applying the standard EM training procedure of Brown et al. (1993).

Algorithm 1 Rank Decoder

Input: Query x, Components $\mathcal{C}$ of size m, rank k, model $\mathcal{A}$, sort function K-BEST
Output: Top k components ranked by $\mathcal{A}$ model score p

```
procedure RANKCOMPONENTS(x, C, k, A)
    SCORES ← [ ]                          ▷ Initialize score list
    for each component c ∈ C do
        p ← ALIGN_A(x, c)                 ▷ Score using A
        SCORES += (c, p)                  ▷ Add to list
    return K-Best(SCORES,k)               ▷ k best components
```

3.3 Ranking and Decoding

Algorithm 1 shows how to rank API components. For a text input x, we iterate through all known API components $\mathcal{C}$ and assign a score using a model $\mathcal{A}$. We then rank the components by their scores using a K-BEST function. This method serves as a type of word-based decoding algorithm

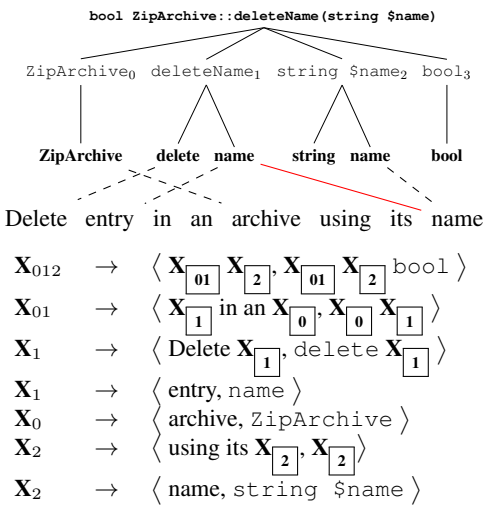


Figure 4: An example tree structure (above) associated with an input component. Below are Hiero rules (Chiang, 2007) extracted from the alignment and tree information.

which is simplified by the finite nature of the target language. The complexity of the scoring procedure, lines 3-5, is linear over the number components m in $\mathcal{C}$. In practice, we implement the K-BEST sorting function on line 6 as a binary insertion sort on line 5, resulting in an overall complexity of $O(m\ log\ m)$.

While iterating over m API components might not be feasible given more complicated formal languages with recursion, a more clever decoding algorithm could be applied, e.g., one based on the lattice decoding approach of (Dyer et al., 2008). Since we are interested in providing initial baseline results, we leave this for future work.

4 Discriminative Approach

In this section, we introduce a new model that aims to improve on the previous baseline methods.

While the previous models are restricted to word-level information, we extend this approach by using a discriminative reranking model that captures phrase information to see if this leads to an improvement. This model can also capture document-level information from the APIs, such as the additional textual descriptions of parameters, *see also* declarations or classes of related functions and syntax information.

4.1 Modeling

Like in most semantic parsing approaches (Zettlemoyer and Collins, 2009; Liang et al., 2011), our model is defined as a conditional log-linear model over components $z \in \mathcal{C}$ with parameters $\theta \in \mathbb{R}^b$, and a set of feature functions $\phi(x, z)$: $p(\, z|\, x; \theta) \propto e^{\theta \cdot \phi(x,z)}$.

Formally, our training objective is to maximize the conditional log-likehood of the correct component output z for each input x: $\mathcal{O}(\theta) = \sum_{i=1}^{n} \log\ p\,(z_i \,|\, x_i; \theta)$.

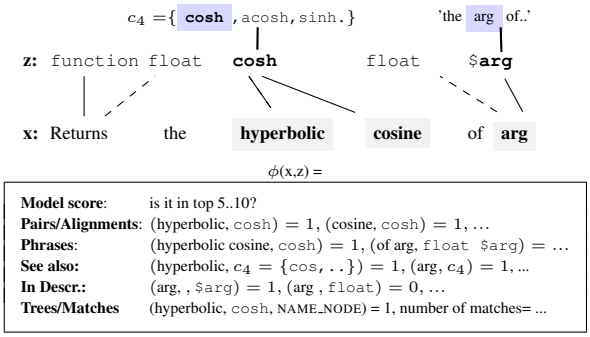


Figure 5: Example features used by our rerankers.

4.2 Features

Our model uses word-level features, such as word match, word pairs, as well as information from the underlying aligner model such as Viterbi alignment information and model score. Two additional categories of non-word features are described below. An illustration of the feature extraction procedure is shown in Figure 5 [1].

Phrases Features We extract phrase features (e.g., (hyper. cosine, `cosh`) in Figure 5) from example text component pairs by training symmetric word aligners and applying standard word-level heuristics (Koehn et al., 2003). Additional features, such as phrase match/overlap, tree positions of phrases, are defined over the extracted phrases.

We also extract hierarchical phrases (Chiang, 2007) using a variant of the SAMT method of Zollmann and Venugopal (2006) and the component syntax trees. Example rules are shown in Figure 4, where *gaps* (i.e., symbols in square brackets) are filled with smaller phrase-tree alignments.

Document Level Features Document features are of two categories. The first includes additional textual descriptions of parameters, return values, and modules. One class of features is whether certain words under consideration appear in the `@param` and `@return` descriptions of the target components. For example, the *arg* token in

[1] A more complete description of features is included as supplementary material, along with all source code.

Algorithm 2 Online Rank Learner

Input: Dataset D, components $\mathcal{C}$, iterations T, rank k, learning rate α, model $\mathcal{A}$, ranker function RANK
Output: Weight vector θ

1: **procedure** LEARNRERANKER($D, \mathcal{C}, T, k, \alpha, \mathcal{A}$, RANK)
2: $\theta \leftarrow 0$ ▷ Initialize
3: **for** t ∈ $1..T$ **do**
4: **for** pairs $(x_i, z_i) \in D$ **do**
5: $\mathcal{S} = \text{RANK}(x_i, \mathcal{C}, k, \mathcal{A})$ ▷ Scored candidates
6: $\Delta = \phi(x_i, z_i) - E_{s \in \mathcal{S} \sim p(s|x_i;\theta)}[\phi(x_i, s)]$
7: $\theta = \theta + \alpha\Delta$ ▷ Update online
8: **return** θ

Figure 5 appears in the textual description of the `$arg` parameter elsewhere in the documentation string.

Other features relate to general information about abstract symbol categories, as specified in *see-also* assertions, or *hyper-link* pointers. By exploiting this information, we extract general classes of functions, for example the set of hyperbolic function (e.g., `sinh`, `cosh`, shown as c_4 in Figure 5), and associate these classes with words and phrases (e.g., *hyperbolic* and *hyperbolic cosine*).

4.3 Learning

To optimize our objective, we use Algorithm 2. We estimate the model parameters θ using a K-best approximation of the standard stochastic gradient updates (lines 6-7), and a ranker function RANK. We note that while we use the ranker described in Algorithm 1, any suitable ranker or decoding method could be used here.

5 Experimental Setup

5.1 Datasets

Source code documentation Our source code documentation collection consists of the standard library for nine programming languages, which are listed in Figure 3. We also use the translated version of the PHP collection for six additional languages, the details of which are shown in Figure 6. The Java dataset was first used in DC, while we extracted all other datasets for this work.

The size of the different datasets are detailed in both figures. The number of pairs is the number of single sentences paired with function representations, which constitutes the core part of these datasets. The number of descriptions is the number of additional textual descriptions provided in the overall document, such as descriptions of parameters or return values.

Dataset	# Pairs	#Descr.	Symbols	Words	Vocab.
PHP_{fr}	6,155	14,058	7,922	70,800	5,904
PHP_{es}	5,823	13,285	7,571	69,882	5,790
PHP_{ja}	4,903	11,251	6,399	65,565	3,743
PHP_{ru}	2,549	6,030	3,340	23,105	4,599
PHP_{tr}	1,822	4,414	2,725	16,033	3,553
PHP_{de}	1,538	3,733	2,417	17,460	3,209

Figure 6: The non-English PHP datasets.

We also quantify the different datasets in terms of unique symbols in the target representations, shown as *Symbols*. All function representations and code sequences are linearized, and in some cases further tokenized, for example, by converting out of camel case or removing underscores.

Man pages The collection of man pages is from Richardson and Kuhn (2014) and includes 921 text-code pairs that span 330 Unix utilities and man pages. Using information from the synopsis and parameter declarations, the target code representations are abstracted by type. The extra descriptions are extracted from parameter descriptions, as shown in the `DESCRIPTION` section in Figure 1, as well as from the `NAME` sections of each manual.

5.2 Evaluation

For evaluation, we split our datasets into separate training, validation and test sets. For Java, we reserve 60% of the data for training and the remaining 40% for validation (20%) and testing (20%). For all other datasets, we use a 70%-30% split. From a retrieval perspective, these left out descriptions are meant to mimic unseen queries to our model. After training our models, we evaluate on these held out sets by ranking all known components in each resource using Algorithm 1. A predicted component is counted as correct if it matches *exactly* a gold component.

Following DC, we report the accuracy of predicting the correct representation at the first position in the ranked list (Accuracy @1) and within the top 10 positions (Accuracy @10). We also report the mean reciprocal rank MRR, or the multiplicative inverse of the rank of the correct answer.

Baselines For comparison, we trained a bag-of-words classifier (the BoW Model in Table 1). This model uses the occurrence of word-component symbol pairs as binary features, and aims to see if word co-occurrence alone is sufficient to for ranking representations.

Method	Java	PHP_{en}	Python	Haskell	Clojure	Ruby	Elisp	C
BOW Model	16.4 63.8 31.8	08.0 40.5 18.1	04.1 33.3 13.6	05.6 55.6 21.7	03.0 49.2 16.4	07.0 38.0 16.9	09.9 54.6 23.5	08.8 48.8 20.0
Term Match	15.7 41.3 24.8	15.6 37.0 23.1	16.6 41.7 24.8	15.4 41.8 24.0	20.7 49.2 30.0	23.1 46.9 31.2	29.3 65.4 41.4	13.1 37.5 21.9
IBM M1	**34.3 79.8 50.2**	**35.5 70.5 47.2**	**22.7 61.0 35.8**	**22.3 70.3 39.6**	**29.6 69.2 41.6**	**31.4 68.5 44.2**	**30.6 67.4 43.5**	21.8 **63.7** 34.4
IBM M2	30.3 77.2 46.5	33.2 67.7 45.0	21.4 58.0 34.4	13.8 68.2 31.8	26.5 64.2 38.2	27.9 66.0 41.4	28.1 66.1 40.7	**23.7** 60.9 **34.6**
Tree Model	29.3 75.4 45.3	28.0 63.2 39.8	17.5 55.4 30.7	17.8 65.4 35.2	23.0 60.3 34.4	27.1 63.3 39.5	26.8 63.2 39.7	18.1 56.2 29.4
M1 Descr.	33.3 77.0 48.7	34.1 71.1 47.2	22.7 62.3 35.9	23.9 69.5 40.2	29.6 69.2 41.6	32.5 70.0 45.5	30.3 73.4 44.7	21.8 62.7 33.9
Reranker	**35.3 81.5 51.4**	**36.9 74.2 49.3**	**25.5 66.0 38.7**	**24.7 73.9 43.0**	**35.0 76.9 47.9**	**35.1 72.5 48.0**	**37.6 80.5 53.3**	**29.7 67.4 40.1**

Method	Scheme	PHP_{fr}	PHP_{es}	PHP_{ja}	PHP_{ru}	PHP_{tr}	PHP_{de}	Unix
BOW Model	06.1 58.1 21.4	06.1 36.9 16.0	05.9 37.8 15.8	04.7 33.2 13.8	04.4 43.6 16.6	05.4 43.4 17.6	04.3 39.2 15.3	08.6 49.6 21.0
Term Match	25.5 61.2 37.4	04.0 15.8 07.7	02.9 10.4 05.4	02.3 11.2 05.2	01.0 09.3 03.6	01.4 08.7 03.6	03.8 09.4 06.2	15.1 33.8 22.4
IBM M1	**32.1 75.5 46.2**	**32.1 65.1 43.5**	**29.5 63.7 41.2**	**23.0 58.1 34.9**	20.3 58.4 **33.3**	**25.9 61.6 38.6**	**22.8 62.5 36.8**	**30.2 66.9 42.2**
IBM M2	29.5 71.4 43.9	30.6 62.2 41.2	26.7 59.8 38.3	22.2 56.1 33.3	18.5 54.5 30.6	23.3 57.6 35.8	19.8 58.6 33.0	23.0 60.4 36.0
Tree Model	26.1 71.2 40.3	27.9 59.3 38.6	25.9 61.0 37.6	22.6 57.8 34.1	**20.6 59.0** 32.9	18,9 55.1 32.0	18.5 56.0 30.6	23.0 58.2 34.3
M1 Descr.	33.1 75.5 47.1	31.0 64.8 42.7	28.6 64.9 41.1	25.4 60.4 37.0	**21.1** 62.6 34.5	29.1 62.0 **41.4**	26.7 62.0 38.8	**34.5** 71.9 47.4
Reranker	**34.6 77.5 48.9**	**32.7 66.8 44.2**	**30.6 66.3 42.6**	**25.8 61.8 37.8**	21.1 **66.8 35.9**	**29.9 63.8** 41.2	**28.0 65.9 40.5**	34.5 **74.8 48.5**

| Accuracy @1 | Accuracy @10 | Mean Reciprocal Rank (MRR) |

Table 1: Test results according to the table below.

Since our discriminative models use more data than the baseline models, which therefore make the results not directly comparable, we train a more comparable translation model, shown as *M1 Descr.* in Table 1, by adding the additional textual data (i.e. parameter and return or module descriptions) to the models' parallel training data.

6 Results and Discussion

Test results are shown in Table 1. Among the baseline models, IBM Model 1 outperforms virtually all other models and is in general a strong baseline. Of particular note is the poor performance of the higher-order translation models based on Model 2 and the Tree Model. While Model 2 is known to outperform Model 1 on more conventional translation tasks (Och and Ney, 2003), it appears that such improvements are not reflected in this type of semantic translation context.

The bag-of-words (BoW) and Term Match baselines are outperformed by all other models. This shows that translation in this context is more complicated than simple word matching. In some cases the term matching baseline is competitive with other models, suggesting that API collections differ in how language descriptions overlap with component names and naming conventions. It is clear, however, that this heuristic only works for English, as shown by results on the non-English PHP datasets in Table 1.

We achieve improvements on many datasets by adding additional data to the translation model (M1 Descr.). We achieve further improvements on all datasets using the discriminative model (Reranker), with most increases in performance occurring at how the top ten items are ranked. This last result suggests that phrase-level and document-level features can help to improve the overall ranking and translation, though in some cases the improvement is rather modest.

Despite the simplicity of our semantic parsing model and decoder, there is still much room for improvement, especially on achieving better Accuracy @1. While one might expect better results when moving from a word-based model to a model that exploits phrase and hierarchical phrase features, the sparsity of the component vocabulary is such that most phrase patterns in the training are not observed in the evaluation. In many benchmark semantic parsing datasets, such sparsity issues do not occur (Cimiano and Minock, 2009), suggesting that state-of-the-art methods will have similar problems when applied to our datasets.

Recent approaches to open-domain semantic parsing have dealt with this problem by using paraphrasing techniques (Berant and Liang, 2014) or distant supervision (Reddy et al., 2014). We expect that these methods can be used to improve our models and results, especially given the wide availability of technical documentation, for example, distributed within the Opus project (Tiedemann, 2012).

Model Errors We performed analysis on some of the incorrect predictions made by our models. For some documentation sets, such as those in the GNU documentation collection[2], information is organized into a small and concrete set of categories/chapters, each corresponding to various features or modules in the language and related functions. Given this information, Figure

[2]https://www.gnu.org/doc/doc.en.html

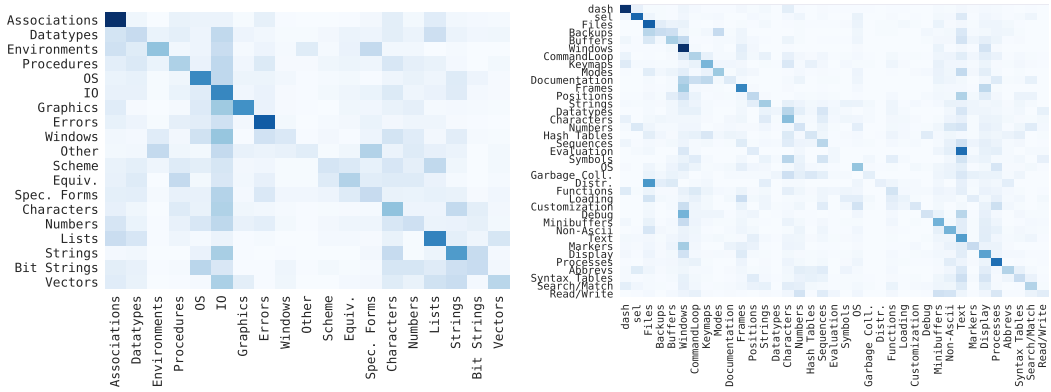


Figure 7: Function predictions by documentation category for Scheme (left) and Elisp (right).

7 shows the confusion between predicting different categories of functions, where the rows show the categories of functions to be predicted and the columns show the different categories predicted. We built these plots by finding the categories of the top 50 non-gold (or erroneous) representations generated for each validation example.

The step-like lines through the diagonal of both plots show that alternative predictions (shaded according to occurrence) are often of the same category, most strikingly for the corner categories. This trend seems stable across other datasets, even among datasets with large numbers of categories. Interestingly, many confusions appear to be between related categories. For example, when making predictions about `Strings` functions in Scheme, the model often generates function related to `BitStrings`, `Characters` and `IO`. Again, this trend seems to hold for other documentation sets, suggesting that the models are often making semantically sensible decisions.

Looking at errors in other datasets, one common error involves generating functions with the same name and/or functionality. In large libraries, different modules sometimes implement that same core functions, such the `genericpath` or `posixpath` modules in Python. When generating a representation for the text *return size of file*, our model confuses the `getsize(filename)` function in one module with others. Similarly, other subtle distinctions that are not explicitly expressed in the text descriptions are not captured, such as the distinction in Haskell between *safe* and *unsafe* bit shifting functions.

While many of these predictions might be correct, our evaluation fails to take into account these various equivalences, which is an issue that should be investigated in future work. Future work will also look systematically at the effect that types (i.e., in statically typed versus dynamic languages) have on prediction.

7 Future Work

We see two possible use cases for this data. First, for benchmarking semantic parsing models on the task of semantic translation. While there has been a trend towards learning executable semantic parsers (Berant et al., 2013; Liang, 2016), there has also been renewed interest in supervised learning of formal representations in the context of neural semantic parsing models (Dong and Lapata, 2016; Jia and Liang, 2016). We believe that good performance on our datasets should lead to better performance on more conventional semantic parsing tasks, and raise new challenges involving sparsity and multilingual learning.

We also see these resources as useful for investigations into natural language programming. While our experiments look at learning rudimentary translational correspondences between text and code, a next step might be learning to synthesize executable programs via these translations, along the lines of (Desai et al., 2016; Raza et al., 2015). Other document-level features, such as example input-output pairs, unit tests, might be useful in this endeavor.

Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG) via SFB 732, project D2. Thanks also to our IMS colleagues, in particular Christian Scheible, for providing feedback on earlier drafts, as well as to Jonathan Berant for helpful discussions.

References

Miltiadis Allamanis, Daniel Tarlow, Andrew D Gordon, and Yi Wei. 2015. Bimodal modelling of source code and natural language. In *Proceedings of the 32th International Conference on Machine Learning*. volume 951, page 2015.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1699–1710.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *In Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *in Proceedings of EMNLP-2013*. pages 1533–1544.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL-2014*. pages 1415–1425.

Benjamin Börschinger, Bevan K. Jones, and Mark Johnson. 2011. Reducing grounded learning tasks to grammatical inference. In *Proceedings of EMNLP-2011*. pages 1416–1425.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics* 19(2):263–311.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of ICML-2008*. pages 128–135.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics* 33(2):201–228.

Philipp Cimiano and Michael Minock. 2009. Natural language interfaces: what is the problem?–a data-driven quantitative analysis. In *International Conference on Application of Natural Language to Information Systems*. Springer, pages 192–206.

Huijing Deng and Grzegorz Chrupała. 2014. Semantic approaches to software component retrieval with English queries. In *Proceedings of LREC-14*. pages 441–450.

Aditya Desai, Sumit Gulwani, Vineet Hingorani, Nidhi Jain, Amey Karkare, Mark Marron, Subhajit Roy, et al. 2016. Program synthesis using natural language. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, pages 345–356.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280* .

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. *Proceedings of ACL-08* page 1012.

Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. 2016. Deep API Learning. *arXiv preprint arXiv:1605.08535* .

Srinivasan Iyer, Ioannis Kostas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. *Proceedings of ACL-2016* .

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622* .

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the NACL-2003*. pages 48–54.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of ACL-2014*. pages 271–281.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of NAACL-2013*.

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

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL-11*. pages 590–599.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM* 59(9):68–76.

Fei Lv, Hongyu Zhang, Jian-guang Lou, Shaowei Wang, Dongmei Zhang, and Jianjun Zhao. 2015. Codehow: Effective code search based on api understanding and extended boolean model (e). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*. IEEE, pages 260–270.

Mehdi Hafezi Manshadi, Daniel Gildea, and James F Allen. 2013. Integrating programming by example and natural language programming. In *Proceedings of AAAI-2013*.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. 2012. Learning to parse natural language commands to a robot control system. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics* 29(1):19–51.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*. IEEE, pages 574–584.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of ACL-2015*.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A Synchronous Hyperedge Replacement Grammar based approach for AMR parsing. *Proceedings of CoNLL-2015* page 32.

Chris Quirk, Raymond J Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of ACL-2015*. pages 878–888.

Mohammad Raza, Sumit Gulwani, and Natasa Milic-Frayling. 2015. Compositional program synthesis from natural language and examples. In *IJCAI*. pages 792–800.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics* 2:377–392.

Kyle Richardson and Jonas Kuhn. 2014. UnixMan corpus: A resource for language learning in the Unix domain. In *Proceedings of LREC-2014*.

Kyle Richardson and Jonas Kuhn. 2016. Learning to make inferences in a semantic parsing task. *Transactions of the Association for Computational Linguistics* 4:155–168.

F. Song and W.B Croft. 1999. A general language model for information retrieval. In *in Proceedings International Conference on Information and Knowledge Management*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *LREC*. volume 2012, pages 2214–2218.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of HLT-NAACL-2006*. pages 439–446.

Yuk Wah Wong and Raymond J Mooney. 2007a. Generation by inverting a semantic parser that uses statistical machine translation. In *Proceedings of HLT-NAACL-2007*. pages 172–179.

Yuk Wah Wong and Raymond J. Mooney. 2007b. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of ACL-2007*. Prague, Czech Republic.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of AAAI-1996*. pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of ACL-2009*. pages 976–984.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*. pages 138–141.

Bridging Text and Knowledge by Learning Multi-Prototype Entity Mention Embedding

Yixin Cao[1], **Lifu Huang**[2], **Heng Ji**[2], **Xu Chen**[1], **Juanzi Li**[1*]
[1] Tsinghua National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, China 100084
{caoyixin2011,successcx,lijuanzi2008}@gmail.com
[2] Dept. of Computer Science, Rensselaer Polytechnic Institute, USA 12180
{huangl7,jih}@rpi.edu

Abstract

Integrating text and knowledge into a unified semantic space has attracted significant research interests recently. However, the ambiguity in the common space remains a challenge, namely that the same mention phrase usually refers to various entities. In this paper, to deal with the ambiguity of entity mentions, we propose a novel Multi-Prototype Mention Embedding model, which learns multiple sense embeddings for each mention by jointly modeling words from textual contexts and entities derived from a knowledge base. In addition, we further design an efficient language model based approach to disambiguate each mention to a specific sense. In experiments, both qualitative and quantitative analysis demonstrate the high quality of the word, entity and multi-prototype mention embeddings. Using entity linking as a study case, we apply our disambiguation method as well as the multi-prototype mention embeddings on the benchmark dataset, and achieve state-of-the-art performance.

1 Introduction

Jointly learning text and knowledge representations in a unified vector space greatly benefits many Natural Language Processing (NLP) tasks, such as knowledge graph completion (Han et al., 2016; Wang and Li, 2016), relation extraction (Weston et al., 2013), word sense disambiguation (Mancini et al., 2016), entity classification (Huang et al., 2017) and linking (Huang et al., 2015).

Existing work can be roughly divided into two categories. One is encoding words and entities into a unified vector space using Deep Neural Networks (DNN). These methods suffer from the problems of expensive training and great limitations on the size of word and entity vocabulary (Han et al., 2016; Toutanova et al., 2015; Wu et al., 2016). The other is to learn word and entity embeddings separately, and then align similar words and entities into a common space with the help of Wikipedia hyperlinks, so that they share similar representations (Wang et al., 2014; Yamada et al., 2016).

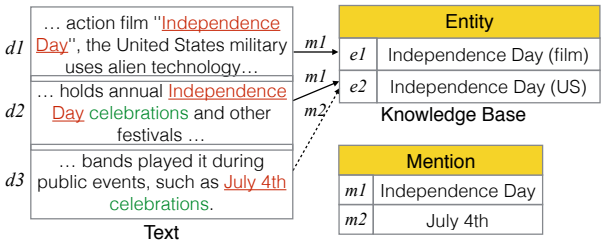


Figure 1: Examples.

However, there are two major problems arising from directly integrating word and entity embeddings into a unified semantic space. First, mention phrases are highly ambiguous and can refer to multiple entities in the common space. As shown in Figure 1, the same mention *independence day* (m_1) can either refer to a holiday: *Independence Day (US)* or a film: *Independence Day (film)*. Second, an entity often has various aliases when mentioned in various contexts, which implies a much larger size of mention vocabulary compared with entities. For example, in Figure 1, the documents d_2 and d_3 describes the same entity *Independence Day (US)* (e_2) with distinct mentions: *independence day* and *July 4th*. We observe tens of millions of mentions referring to 5 millions of entities in Wikipedia.

To address these issues, we propose to learn multiple embeddings for mentions inspired by the Word Sense Disambiguation (WSD) task (Reisinger and Mooney, 2010; Huang et al., 2012;

*Corresponding author.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1623–1633
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1149

Tian et al., 2014; Neelakantan et al., 2014; Li and Jurafsky, 2015). The basic idea behind it is to consider entities in KBs that can provide a meaning repository of mentions (i.e. words or phrases) in texts. That is, each mention has one or multiple meanings, namely **mention senses**, and each sense corresponds to an entity. Furthermore, we assume that different mentions referring to the same entity express the same meaning and share a common mention sense embedding, which largely reduces the size of mention vocabulary to be learned. For example, the mentions *Independence Day* in d_2 and *July 4th* in d_3 have a common mention sense embedding during training since they refer to the same holiday. Thus, text and knowledge are bridged via mention sense.

In this paper, we propose a novel **M**ulti-**P**rototype **M**ention **E**mbedding (MPME) model, which jointly learns the representations of words, entities, and mentions at sense level. Different mention senses are distinguished by taking advantage of both textual context information and knowledge of reference entities. Following the frameworks in (Wang et al., 2014; Yamada et al., 2016), we use separate models to learn the representations for words, entities and mentions, and further align them by a unified optimization objective. Extending from skip-gram model and CBOW model, our model can be trained efficiently (Mikolov et al., 2013a,b) from a large scale corpus. In addition, we also design a language model based approach to determine the sense for each mention in a document based on multi-prototype mention embeddings.

For evaluation, we first provide qualitative analysis to verify the effectiveness of MPME to bridge text and knowledge representations at the sense level. Then, separate tasks for words and entities show improvements by using our word, entity and mention representations. Finally, using entity linking as a case study, experimental results on the benchmark dataset demonstrate the effectiveness of our embedding model as well as the disambiguation method.

2 Preliminaries

In this section, we formally define the input and output of multi-prototype mention embedding.

A **knowledge base** $\mathcal{KB}$ contains a set of entities $\mathcal{E} = \{e_j\}$, and their relations. We use Wikipedia as the given knowledge base, and organize it as a directed knowledge network: nodes denote entities, and edges are outlinks from Wikipedia pages. In the directed network, we define the entities that point to e_j as its neighbors $\mathcal{N}(e_j)$, but ignore those entities that e_j points to, so that the repeated computations on the same edge would be avoided if edges were undirected.

A **text corpus** $\mathcal{D}$ is a set of sequential words $\mathcal{D} = \{w_1, \cdots, w_i, \cdots, w_{|\mathcal{D}|}\}$, where w_i is the ith word and $|\mathcal{D}|$ is the length of the word sequence. Since an entity mention m_l may consist of multiple words, we define an annotated text corpus[1] as $\mathcal{D}' = \{x_1, \cdots, x_i, \cdots, x_{|\mathcal{D}'|}\}$, where x_i corresponds to either a word w_i or a mention m_l. We define the words around x_i within a predefined window as its context words $\mathcal{C}(x_i)$.

An **Anchor** is a Wikipedia hyperlink from a mention m_l linking to its entity e_j, and is represented as a pair $< m_h, e_j > \in \mathcal{A}$. The anchors provide mention boundaries as well as their reference entities from Wikipedia articles. These Wikipedia articles are used as an annotated text corpus $\mathcal{D}'$ in this paper.

Multi-Prototype Mention Embedding . Given a $\mathcal{KB}$, an annotated text corpus $\mathcal{D}'$ and a set of anchors $\mathcal{A}$, we aim to learn multi-prototype mention embedding, namely multiple sense embeddings $\mathbf{s_j}^l \in \mathbb{R}^k$ for each mention m_l as well as word embeddings $\mathbf{w}$ and entity embeddings $\mathbf{e}$. We use $\mathcal{M}_l^* = \{s_j^l\}$ to denote the sense set of mention m_l, where each s_j^l refers to an entity e_j. Thus, the vocabulary size is reduced to a fixed number $|\{s_j^*\}| = |\mathcal{E}|$. We use s_j^* to denote the shared sense of mentions referring to entity e_j.

Example As shown in Figure 1, *Independence Day* (m_1) has two mention senses s_1^1, s_2^1, and *July 4th* (m_2) has one mention sense s_2^2. Based on the assumption in Section 1, we have $s_2^* = s_2^1 = s_2^2$ referring to entity *Independence Day (US)* (e_2).

3 An Overview of Our Method

Given a knowledge base $\mathcal{KB}$, an annotated text corpus $\mathcal{D}'$ and a set of anchors $\mathcal{A}$, we aim to jointly learn word, entity and mention sense representations: $\mathbf{w}$, $\mathbf{e}$, $\mathbf{s}$.

As shown in Figure 2, our framework contains two key components:

[1]Generally, the mention boundary can be obtained by using NER tools like Standford NER (Finkel et al., 2005). In this paper, we use Wikipedia anchors as annotations of Wikipedia text corpus for the concentration of our main purpose.

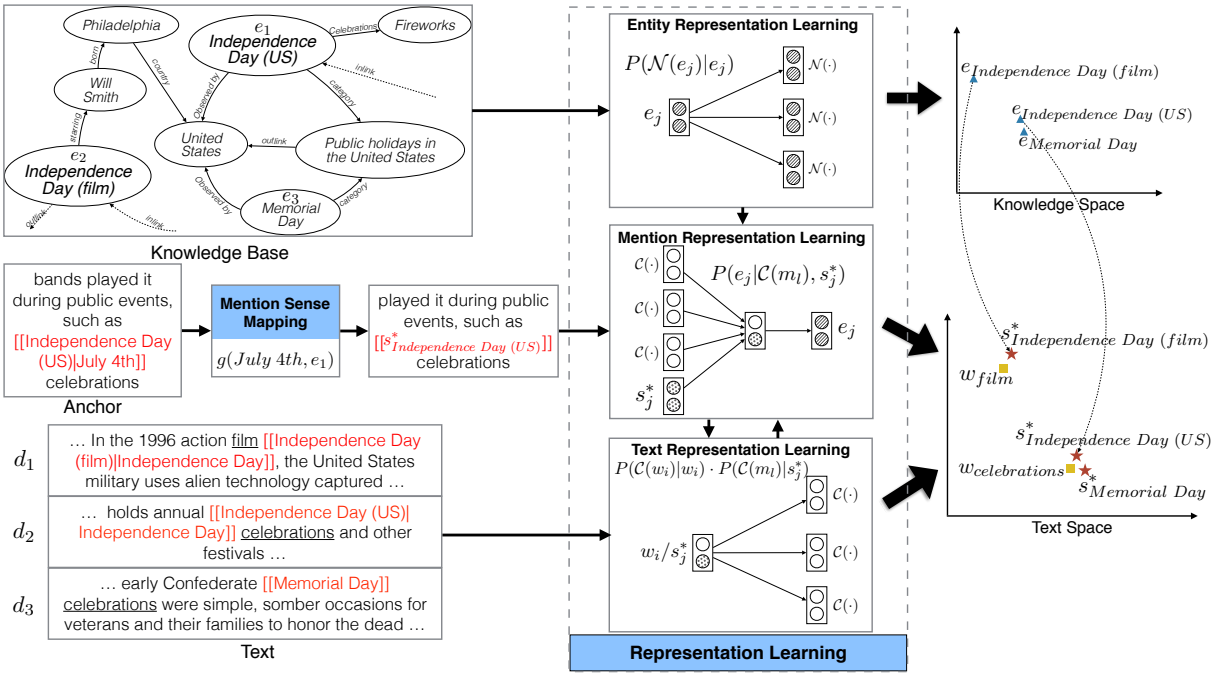


Figure 2: Framework of Multi-Prototype Mention Embedding model.

Mention Sense Mapping To reduce the size of the mention vocabulary, each mention is mapped to a set of shared mention senses according to a predefined dictionary. We build the dictionary by collecting entity-mention pairs $< m_l, e_j >$ from Wikipedia anchors and page titles, then create mention senses if there is a different entity. The sense number of a mention depends on how many different entity-mention pairs it is involved.

Formally, we have: $\mathcal{M}^*_l = g(m_l) = \bigcup g(< m_l, e_j >) = \{s^*_j\}$, where $g(\cdot)$ denotes the mapping function from an entity mention to its mention sense given an anchor. We directly use the anchors contained in the annotated text corpus $D^{'}$ for training. As Figure 2 shows, we replace the anchor *<July 4th, Independence Day (US)>* with the corresponding mention sense: $s^*_{Independence\ Day\ (US)}$.

Representation Learning Using $\mathcal{KB}$, $\mathcal{A}$ and $\mathcal{D}'$ as input, we design three separate models and a unified optimization objective to jointly learn entity, word and mention sense representations into two semantic spaces. As shown in the knowledge space in Figure 2, entity embeddings can reflect their relatedness in the network. For example, *Independence Day (US)* (e_1) and *Memorial Day* (e_3) are close to each other because they share some common neighbors, such as *United States* and *Public holidays in the United States*.

Word and mention embeddings are learned in the same semantic space. As two basic units in $\mathcal{D}'$, their embeddings represent their distributed semantics in texts. For example, mention *Independence Day* and word *celebrations* co-occur frequently when it refers to the holiday: *Independence Day (US)*, thus they have similar representations. Without disambiguating the mention senses, some words, such as *film* will also share similar representations as *Independence Day*.

Besides, by introducing entity embeddings into our MPME framework, the knowledge information will also be distilled into mention sense embeddings, so that the mention sense *Memorial Day* will be similar as *Independence Day (US)*.

Mention Sense Disambiguation According to our predefined dictionary, each mention has been mapped to more than one senses, and learned with multiple embedding vectors. Consequently, to induce the correct sense for a mention within a context is critical in the usage of the multiprototype embeddings, especially in an unsupervised way. Formally, given an annotated document $\mathcal{D}'$, we determine one sense $\hat{s}^*_j \in \mathcal{M}^*_l$ for each mention $m_l \in \mathcal{D}'$, where $\hat{s}^*_j$ is the correct sense.

Based on language model, we design a mention sense disambiguation method without using any supervision that takes into account three aspects: 1) **sense prior** denotes how dominant the sense is, 2) **local context information** reflects how semantically appropriate the sense is in the context, and

3) **global mention information** denotes how semantically consistent the sense is with the neighbor mentions. To better utilize the context information, we maintain a context cluster for each mention sense during training, which will be detailed in Section 4.4.

Since each mention sense corresponds to an entity in the given KB, the disambiguation method is equivalent to entity linking. Thus, text and knowledge base is bridged via the multiprototype mention embeddings. We will give more analysis in Section 6.4.

4 Representation Learning

Distributional representation learning plays an increasing important role in many fileds (Bengio et al., 2013; Zhang et al., 2017, 2016) due to its effectiveness for dimensionality reduction and addressing sparseness issue. For NLP tasks, this trends has been accelerated by the Skip-gram and CBOW models (Mikolov et al., 2013a,b) due to its efficiency and remarkable semantic compositionality of embedding vectors. In this section, we first briefly introduce the Skip-gram and CBOW models, and then extend them to three variants for the word, mention and entity representation learning.

4.1 Skip-Gram and CBOW model

The basic idea of the Skip-gram and CBOW models is to model the predictive relations among sequential words. Given a sequence of words $\mathcal{D}$, the optimization objective of Skip-gram model is to use the current word to predict its context words by maximizing the average log probability:

$$\mathcal{L} = \sum_{w_i \in \mathcal{D}} \sum_{w_o \in \mathcal{C}(w_i)} \log P(w_o|w_i) \tag{1}$$

In contrast, CBOW model aims to predict the current word given its context words:

$$\mathcal{L} = \sum_{w_i \in \mathcal{D}} \log P(w_i|\mathcal{C}(w_i)) \tag{2}$$

Formally, the conditional probability $P(w_o|w_i)$ is defined using a softmax function:

$$P(w_o|w_i) = \frac{\exp(\mathbf{w_i} \cdot \mathbf{w_o})}{\sum_{w_o \in \mathcal{D}} \exp(\mathbf{w_i} \cdot \mathbf{w_o})} \tag{3}$$

where $\mathbf{w_i}, \mathbf{w_o}$ denote the input and output word vectors during training. Furthermore, these two models can be accelerated by using hierarchical softmax or negative sampling (Mikolov et al., 2013a,b).

4.2 Entity Representation Learning

Given a knowledge base $\mathcal{KB}$, we aim to learn entity embeddings by modeling "contextual" entities, so that the entities sharing more common neighbors tend to have similar representations. Therefore, we extend Skip-gram model to a network by maximizing the log probability of being a neighbor entity.

$$\mathcal{L}_e = \sum_{e_j \in \mathcal{E}} \log P(\mathcal{N}(e_j)|e_j) \tag{4}$$

Clearly, the neighbor entities serve a similar role as the context words in Skip-gram model. As shown in Figure 2, entity *Memorial Day* (e_3) also share two common neighbors of *United States* and *Public holidays in the United States* with entity *Independence Day (US)*, thus their embeddings are close in the Knowledge Space. These entity embeddings will be later used to learn mention representations.

4.3 Mention Representation Learning

As mentioned above, the textual context information and reference entities are helpful to distinguish different senses for a mention. Thus, given an anchor $< m_l, e_j >$ and its context words $\mathcal{C}(m_l)$, we combine mention sense embeddings with its context word embeddings to predict the reference entity by extending CBOW model. The objective function is as follows:

$$\mathcal{L}_m = \sum_{<m_l, e_j> \in \mathcal{A}} \log P(e_j|\mathcal{C}(m_l), s_j^*) \tag{5}$$

where $s_j^* = g(< m_l, e_j >)$. Thus, if two mentions refer to similar entities and share similar contexts, they tend to be close in semantic vector space. Take Figure 1 as an example again, mentions *Independence Day* and *Memorial Day* refer to similar entities *Independence Day (US)* (e_1) and *Memorial Day* (e_2), they also share some similar context words, such as *celebrations* in documents d_2, d_3, so their sense embeddings are close to each other in the text space.

4.4 Text Representation Learning

Instead of directly using a word or a mention to predict the context words, we incorporate mention

sense to joint optimize word and sense representations, which can avoid some noise introduced by ambiguous mentions. For example, in Figure 2, without identifying the mention *Independence Day* as the holiday or the film, various dissimilar context words such as the words *celebrations* and *film* in documents d_1, d_2 will share similar semantics, which will further affect the performance of entity representations during joint training.

Given the annotated corpus $\mathcal{D}'$, we use a word w_i or a mention sense s_j^* to predict the context words by maximizing the following objective function:

$$\mathcal{L}_w = \sum_{w_i, m_l \in \mathcal{D}'} \log P(\mathcal{C}(w_i)|w_i) + \log P(\mathcal{C}(m_l)|s_j^*) \quad (6)$$

where $s_j^* = g(< m_l, e_j >)$ is obtained from anchors in Wikipedia articles.

Thus, words and mention senses will share the same vector space, where similar words and mention senses are close to each other, such as *celebrations* and *Independence Day (US)* because they frequently occur in the same contexts.

Similar to WDS, we maintain a context cluster for each mention sense, which can be used for mention sense disambiguation (Section 5). The context cluster of a mention sense s_j^* contains all the context vectors of its mention m_l. We compute context vector of m_l by averaging the sum of its context word embeddings: $\frac{1}{|\mathcal{C}(m_l)|}\sum_{w_j \in \mathcal{C}(m_l)} \mathbf{w_j}$. Further, the center of a context cluster μ_j^* is defined as the average of context vectors of all mentions which refer to the sense. These context clusters will be later used to disambiguate the sense of a given mention with its contexts.

4.5 Joint Training

Considering all of the above representation learning components, we define the overall objective function as linear combinations:

$$\mathcal{L} = \mathcal{L}_w + \mathcal{L}_e + \mathcal{L}_m \quad (7)$$

The goal of training MPME is to maximize the above function, and iteratively update three types of embeddings. Also, we use negative sampling technique for efficiency (Mikolov et al., 2013a).

MPME shares the same entity representation learning method with (Yamada et al., 2016), but the role of entities in the entire framework as well as mention representation learning is different in three aspects. First, we focus on learning embeddings for mentions, not merely words as in (Yamada et al., 2016). Clearly, MPME is more natural to integrate text and knowledge base. Second, we propose to learn multiple embeddings for each mention denoting its different meanings. Third, we prefer to use both mentions and context words to predict entities, so that the distribution of entities will help improve word embeddings, meanwhile, avoid being hurt if we force entity embeddings to satisfy word embeddings during training (Wang et al., 2014). We will give more analysis in experiments.

5 Mention Sense Disambiguation

As mentioned in Section 3, we induce a correct sense $\hat{s}_j^* \in \mathcal{M}_l^*$ for each mention m_l in an annotated document $\mathcal{D}'$. We regard this problem from the perspective of language model that maximizes a joint probability of all mention senses contained in the document. However, the global optimum is expensive with a time complexity of O($|\mathcal{M}||\mathcal{M}_l^*|$). Thus, we approximately identify each mention sense independently:

$$\begin{aligned} & P(\mathcal{D}', \ldots, s_j^*, \ldots,) \\ \approx & \prod P(\mathcal{D}'|s_j^*) \cdot P(s_j^*) \\ \approx & \prod P(\mathcal{C}(m_l)|s_j^*) \cdot P(\hat{\mathcal{N}}(m_l)|s_j^*) \cdot P(s_j^*) \end{aligned} \quad (8)$$

where $P(\mathcal{C}(m_l)|s_j^*)$, local context information (Section 3), denotes the probability of the local contexts of m_l given its mention sense s_j^*. we define it proportional to the cosine similarity between the current context vector and the sense context cluster center μ_j^* as described in Section 4.4. It measures how likely a mention sense occurring together with current context words. For example, given the mention sense *Independence Day (film)*, word *film* is more likely to appear within the context than the word *celebrations*.

$P(\hat{\mathcal{N}}(m_l)|s_j^l)$, global mention information, denotes the probability of the contextual mentions of m_l given its sense s_j^l, where $\hat{\mathcal{N}}(m_l)$ is the collection of the neighbor mentions occurring together with m_l in a predefined context window. We define it proportional to the cosine similarity between mention sense embeddings and the neighbor mention vector, which is computed similar to

context vector: $\sum \frac{1}{|\hat{\mathcal{N}}(m_l)|}\hat{\mathbf{s}}^{\mathbf{l}}_{\mathbf{j}}$, where $\hat{s}^l_j$ is the correct sense for m_l.

Considering there are usually multiple mentions in a document to be disambiguated. The mentions disambiguated first will be helpful for inducing the senses of the rest mentions. That is, how to choose the mentions disambiguated first will influence the performance. Intuitively, we adopt two orders similar to (Chen et al., 2014): 1) L2R (left to right) induces senses for all the mentions in the document following natural order that varies according to language, normally from left to right in the sequence. 2) S2C (simple to complex) denotes that we determine the correct sense for those mentions with fewer senses, which makes the problem easier.

Global mention information assumes that there should be consistent semantics in a context window, and measures whether all neighbor mentions are related. For instance, two mentions *Memorial Day* and *Independence Day* occur in the same document. If we already know that *Memorial Day* denotes a holiday, then obviously *Independence Day* has higher probability of being a holiday than a film.

$P(s^*_j)$, sense prior, is a prior probability of sense s^*_j indicating how possible it occurs without considering any additional information. We define it proportional to the frequency of sense s^*_j in Wikipedia anchors:

$$P(s^*_j) = (\frac{|\mathcal{A}_{s^*_j}|}{|\mathcal{A}|})^\gamma \quad \gamma \in [0, 1]$$

where $\mathcal{A}_{s^*_j}$ is the set of anchors annotated with s^*_j, and γ is a smoothing hyper-parameter to control the impact of prior on the overall probability, which is set by experiments (Section 6.4).

6 Experiment

Setup We choose Wikipedia, the March 2016 dump, as training corpus, which contains nearly 75 millions of anchors, 180 millions of edges among entities and 1.8 billions of tokens after preprocessing. We then train MPME[2] for 1.5 millions of words, 5 millions of entities and 1.7 millions of mentions. The entire training process in 10 iterations costs nearly 8 hours on the server with 64 core CPU and 188GB memory.

We use the default settings in word2vec[3], and set our embedding dimension as 200 and context window size as 5. For each positive example, we sample 5 negative examples[4].

Baseline Methods As far as we know, this is the first work to deal with mention ambiguity in the integration of text and knowledge representations, so there is no exact baselines for comparison. We use the method in (Yamada et al., 2016) as a baseline, marked as **ALIGN**[5], because (1) this is the most similar work that directly aligns word and entity embeddings. (2) it achieves the state-of-the-art performance in entity linking task.

To investigate the effect of multi-prototype, we degrade our method to single-prototype as another baseline, which means to use one sense to represent all mentions with the same phrase, namely **S**ingle-**P**rototype **M**ention **E**mbedding (SPME). For example, SPME only learns one unique sense vector for *Independence Day* whatever it denotes a holiday or a film.

6.1 Qualitative Analysis

We use cosine similarity to measure the similarity of two vectors, and present the top 5 nearest words and entities for two most popular senses of the mention *Independence Day*. Because ALIGN is incapable of dealing with multiple words, we only present the results of SPME and MPME.

As shown in Figure 1, without considering mention sense, the mention *Independence Day* can only show a dominant *holiday* sense based on SPME and ignore all other senses. Instead, MPME successfully learns two clear and distinct senses. For the sense *Independence Day (US)*, all of its nearest words and entities, such as *parades*, *celebrations*, and *Memorial Day*, are holiday related, while for another sense *Independence Day (film)*, its nearest words and entities, like *robocop* and *The Terminator*, are all science fiction films. The results verify the effectiveness of our framework in learning mention embeddings at the sense level.

[2]Our main code for MPME can be found in https://github.com/TaoMiner/bridgeGap.

[3]https://code.google.com/archive/p/word2vec/

[4]We tested different parameters (e.g. window size of 10 and dimension of 500) which achieve similar results, and report the current settings considering program runtime efficiency.

[5]We carefully re-implemented ALIGN and used the same shared parameters in our model for fairly comparison. However, we failed to fully reproduce the positive result in the original paper, meanwhile the authors are unable to release their code.

	Mention Sense	Nearest words	Nearest entities
SPME	Independence Day	lee-jackson, thanksgiving, diwali, strassenfest, chiraghan	National Aboriginal and Torres Strait Islander Education Policy, E. Chandrasekharan Nair, Jean Aileen Little, Thessalian barbel, 1825 in birding and ornithology
MPME	Independence Day (US)	thanksgiving, parades, lee-jackson, festivities, celebrations	Memorial Day, Labor Day, Thanksgiving, Thanksgiving (United States), Saint Patrick's Day
	Independence Day (film)	robocop, clockstoppers, mindhunters, tarantino, terminator	The Terminator, True Lies, Total Recall (1990 film), RoboCop 2, Die Hard

Table 1: The nearest neighbors of mention *Independence Day*.

6.2 Entity Relatedness

To evaluate the quality of entity embeddings, we conduct experiments using the dataset which is designed for measuring entity relatedness (Ceccarelli et al., 2013; Huang et al., 2015; Yamada et al., 2016). The dataset contains 3,314 entities, and each mention has 91 candidate entities on average with gold-standard labels indicating whether they are semantically related.

We compute cosine similarity between entity embeddings to measure their relatedness, and rank them in a descending order. To evaluate the ranking quality, we use two standard metrics: normalized discounted cumulative gain (NDCG) (Järvelin and Kekäläinen, 2002) and mean average precision (MAP) (Schütze, 2008).

We design another baseline method: **Entity2vec**, which learns entity embeddings using the method described in Section 4.2, without joint training with word and mention sense embeddings.

Table 2: Entity Relatedness.

	NDCG			MAP
	@1	@5	@10	
ALIGN	0.416	0.432	0.472	0.410
Entity2vec	0.593	0.595	0.636	0.566
SPME	0.593	0.594	0.636	0.566
MPME	**0.613**	**0.613**	**0.654**	**0.582**

As shown in Table 2, ALIGN achieves lower performance than Entity2vec, because it doesn't consider the mention phrase ambiguity and yields lots of noise when forcing entity embeddings to satisfy word embeddings and aligning them into the unified space. For example, the entity *Gente (magazine)* should be more relevant to the entity *France*, the place where its company locates. However, ALIGN mixed various meanings of mention *Gente* (e.g., the song) and ranked some bands higher (e.g., entity *Poolside (band)*).

SPME also doesn't consider the ambiguity of mentions but achieves comparative results with Entity2vec. We analyze the reasons and find that, it can avoid some noise by using word embeddings to predict entities. MPME outperforms all the other methods, which demonstrates that the unambiguous textual information is helpful to refine the entity embeddings.

6.3 Word Analogical Reasoning

Following (Mikolov et al., 2013a; Wang et al., 2014), we use the word analogical reasoning task to evaluate the quality of word embeddings. The dataset consists of 8,869 semantic questions (*"Paris":"France"::"Rome":?*), and 10,675 syntactic questions (e.g., *"sit":"sitting"::"walk":?*). We solve it by finding the closest word vector $\mathbf{w}_?$ to $\mathbf{w}_{France} - \mathbf{w}_{Paris} + \mathbf{w}_{Rome}$ according to cosine similarity. We compute accuracy for top 1 nearest word to measure the performance.

Table 3: Word Analogical Reasoning.

	Word2vec	ALIGN	SPME	MPME
Semantic	66.78	68.34	71.65	71.65
Syntactic	61.58	59.73	55.28	54.75

We also adopt Word2vec[6] as an additional baseline method, which provides a standard to measure the impact from other components on word embeddings.

Table 3 shows the results. We can see that ALIGN, SPME and MPME, achieve higher performance in dealing with semantic questions, because relations among entities (e.g., country-capital relation for entity *France* and *Paris*) enhance the semantics in word embeddings through jointly training. On the other hand, their performance for syntactic questions is weakened because more accurate semantics yields a bias to predict semantic relations even though given a syntactic query. For example, given the query *"pleasant":"unpleasant"::"possibly":?*, our

[6]https://code.google.com/archive/p/word2vec/

model tends to return the word (e.g., *probably*) highly semantical related to query words, such as *possibly*, instead of the syntactical similar word *impossibly*. In this scenario, we are more concerned about semantic task to incorporate knowledge of reference entities into word embeddings, and this issue could be tackled, to some extent, by using syntactic tool like stemming.

The word embeddings of MPME achieve the best performance for semantic questions mainly because (1) text representation learning has better generalization ability due to the larger size of training examples than entities (e.g., 1.8b v.s. 0.18b) as well as relatively smaller size of vocabulary (e.g., 1.5m v.s. 5m). (2) unambiguous mention embeddings capture both textual context information and knowledge, and thus enhance word and entity embeddings.

6.4 A Case Study: Entity Linking

Entity linking is a core NLP task of identifying the reference entity for mentions in texts. The main difficulty lies in the ambiguity of various entities sharing the same mention phrase. Previous work addressed this issue by taking advantage of the similarity between words and entities (Francis-Landau et al., 2016; Sun et al., 2015), and/or the relations among entities (Thien Huu Nguyen, 2016; Cao et al., 2015). Therefore, we use entity linking as a case study for a comprehensive measurement of the multi-prototype mention embeddings. Given mentions in a text, entity linking aims to link them to a predefined knowledge base. One of the main challenges in this task is the ambiguity of entity mentions.

We use the public dataset AIDA created by (Hoffart et al., 2011), which includes 1,393 documents and 27,816 mentions referring to Wikipedia entries. The dataset has been divided into 946, 216 and 231 documents for the purpose of training, developing and testing. Following (Pershina et al., 2015; Yamada et al., 2016), we use a publicly available dictionary to generate candidate entities and mention senses. For evaluation, we rank the candidate entities for each mention and report both standard micro (aggregates over all mentions) and macro (aggregates over all documents) precision over top-ranked entities.

Supervised Entity Linking

Yamada et al. (2016) designed a list of features for each mention and candidate entity pair. By incorporating these features into a supervised learning-to-rank algorithm, Gradient Boosting Regression Tree (GBRT), each pair is assigned a relevance score indicating whether they should be linked to each other. Following their recommended parameters, we set the number of trees as 10,000, the learning rate as 0.02 and the maximum depth of the decision tree as 4.

Based on word and entity embeddings learned by ALIGN, the key features in (Yamada et al., 2016) are from two aspects: (1) the cosine similarity between context words and candidate entity, and (2) the coherence among "contextual" entities in the same document.

To evaluate the performance of multi-prototype mention embeddings, we incorporate the following features into GBDT for comparison: (1) the cosine similarity between the current context vector and the sense context cluster center μ_j^*, which denotes how likely the mention sense refers to the candidate entity, (2) the cosine similarity between the current context vector and the mention sense embeddings.

Table 4: Performance of Supervised Method

	ALIGN	SPME	MPME
Micro P@1	0.828	0.820	**0.851**
Macro P@1	0.862	0.844	**0.881**

As shown in Table 4, we can see that ALIGN performs better than SPME. This is because SPME learns word embeddings and entity embeddings in separate semantic spaces, and fails to measure the similarity between context words and candidate entities. However, MPME computes the similarity between context words with mention sense instead of entities, thus achieves the best performance, which also demonstrates the high quality of the mention sense embeddings.

Unsupervised Entity Linking

Linking a mention to a specific entity equals to disambiguating mention senses since each candidate entity corresponds to a mention sense. As described in Section 5, we disambiguate senses in two orders: (1) **L2R** (from left to right), and (2) **S2C** (from simple to complex).

We evaluate our unsupervised disambiguation methods on the entire AIDA dataset. To be fair, we choose the state-of-the-art unsupervised methods, which are proposed in (Hoffart et al., 2011; Alhelbawy and Gaizauskas, 2014; Cucerzan, 2007;

Table 5: Performance of Unsupervised Methods

	Cucerzan	Kulkarni	Hoffart	Shirakawa	Alhelbawy	MPME (L2R)	MPME (S2C)
Micro P@1	0.510	0.729	0.818	0.823	0.842	0.882	**0.885**
Macro P@1	0.437	0.767	0.819	0.830	0.875	0.875	**0.890**

Kulkarni et al., 2009; Masumi Shirakawa and Nishio, 2011) using the same dataset.

Table 5 shows the results. We can see that our two methods outperform all other methods. MPME (L2R) is more efficient and easy to apply, while MPME (S2C) slightly outperforms it because the additional step of ranking mentions according to their candidates number guarantees a higher disambiguation performance for those simple mentions, which consequently help disambiguate those complex mentions through global mention information in Equation 8.

We analyze the results and observe a disambiguation bias to popular senses. For example, there are three mentions in the sentence "***Japan*** *began the defence of their* ***Asian Cup I*** *title with a lucky 2-1 win against* ***Syria*** *in a Group C championship match on Friday*", where the country name *Japan* and *Syria* actually denote their national football teams, while the football match name *Asian Cup I* has little ambiguity. Compared to the team, the sense of country occurs more frequently and has a dominant prior, which greatly affects the disambiguation. By incorporating local context information and global mention information, both the context words (e.g., *defence* or *match*) and the neighbor mentions (e.g., *Asian Cup I*) provide us enough clues to identify a soccer related mention sense instead of the country.

Influence of Smoothing Parameter As mentioned above, a mention sense may possess a dominant prior and greatly affect the disambiguation. So we introduce a smoothing parameter γ to control its importance to the overall probability. Figure 3 shows the linking accuracy under different values of γ on the dataset of AIDA. $\gamma = 0$ indicates we don't use any prior knowledge, and $\gamma = 1$ indicates the case without smoothing parameter.

We can see that both micro and macro accuracy decrease a lot if we don't use the parameter ($\gamma = 1$). Only using local and global probabilities for disambiguation ($\gamma = 0$) achieves a comparable performance when $\gamma = 0.05$, both accuracy reach their peaks, which is optimal and default value in our experiments.

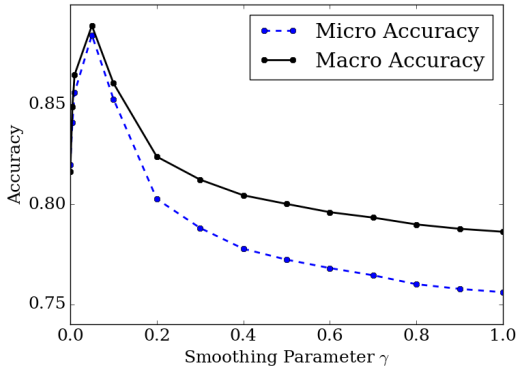


Figure 3: Impact of Smoothing Parameter γ.

7 Conclusions and Future Work

In this paper, we propose a novel Multi-Prototype Mention Embedding model that jointly learns word, entity and mention sense embeddings. These mention senses capture both textual context information and knowledge from reference entities, and provide an efficient approach to disambiguate mention sense in text. We conduct a series of experiments to demonstrate that multi-prototype mention embedding improves the quality of both word and entity representations. Using entity linking as a study case, we apply our disambiguation method as well as the multi-prototype mention embeddings on the benchmark dataset, and achieve the state-of-the-art.

In the future, we will improve the scalability of our model and learn multi-prototype embeddings for the mentions without reference entities in a knowledge base, and introduce compositional approaches to model the internal structures of multi-word mentions.

8 Acknowledgement

This work is supported by NSFC Key Program (No. 61533018), 973 Program (No. 2014CB340504), Fund of Online Education Research Center, Ministry of Education (No. 2016ZD102), Key Technologies Research and Development Program of China (No. 2014BAK04B03), NSFC-NRF (No. 61661146007) and the U.S. DARPA LORELEI Program No. HR0011-15-C-0115.

References

Ayman Alhelbawy and Robert J Gaizauskas. 2014. Graph ranking for collective named entity disambiguation. In *ACL (2)*. pages 75–80.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1798–1828.

Yixin Cao, Juanzi Li, Xiaofei Guo, Shuanhu Bai, Heng Ji, and Jie Tang. 2015. Name list only? target entity disambiguation in short texts. In *EMNLP*. pages 654–664. https://doi.org/10.18653/v1/D15-1077.

Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. 2013. Learning relatedness measures for entity linking. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, pages 139–148.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*. Citeseer, pages 1025–1035. https://doi.org/10.3115/v1/D14-1110.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data .

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 363–370.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of NAACL-HLT*. pages 1256–1261. https://doi.org/10.18653/v1/N16-1150.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2016. Joint representation learning of text and knowledge for knowledge graph completion. *CoRR* abs/1611.04125.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 782–792.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proc. ACL*.

Hongzhao Huang, Larry Heck, and Heng Ji. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *arXiv preprint arXiv:1504.07678* .

Lifu Huang, Jonathan May, Xiaoman Pan, Heng Ji, Xiang Ren, Jiawei Han, Lin Zhao, and James A Hendler. 2017. Liberal entity extraction: Rapid construction of fine-grained entity typing systems. *Big Data* 5(1):19–31.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)* 20(4):422–446.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 457–466.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proc. EMNLP*. https://doi.org/10.18653/v1/D15-1200.

Massimiliano Mancini, José Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2016. Embedding words and senses together via joint knowledge-enhanced training. *CoRR* abs/1612.02703.

Haixun Wang Yangqiu Song Zhongyuan Wang Kotaro Nakayama Takahiro Hara Masumi Shirakawa and Shojiro Nishio. 2011. Entity disambiguation based on a. technical report. In *Technical Report MSR-TR-2011-125*. Microsoft Research.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*. pages 3111–3119.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proc. EMNLP*. https://doi.org/10.3115/v1/D14-1113.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *HLT-NAACL*. pages 238–243.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proc. NAACL*.

Hinrich Schütze. 2008. Introduction to information retrieval. In *Proceedings of the international communication of association for computing machinery conference*.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *IJCAI*. pages 1333–1339.

Nicolas Fauceglia Mariano Rodriguez-Muro Oktie Hassanzadeh Alfio Massimiliano Gliozzo Mohammad Sadoghi Thien Huu Nguyen. 2016. Joint learning of local and global features for entity linking via neural networks. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 2310–2320.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. *ACL Association for Computational Linguistics* https://doi.org/10.18653/v1/D15-1174.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proc. EMNLP*. https://doi.org/10.3115/v1/D14-1167.

Zhigang Wang and Juan-Zi Li. 2016. Text-enhanced representation learning for knowledge graph. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proc. ACL*.

Jiawei Wu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Knowledge representation via joint learning of sequential text and knowledge graphs. *CoRR* .

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proc. CoNLL*. https://doi.org/10.18653/v1/K16-1025.

Hanwang Zhang, Zawlin Kyaw, Shih-Fu Chang, and Tat-Seng Chua. 2017. Visual translation embedding network for visual relation detection. *arXiv preprint arXiv:1702.08319* .

Hanwang Zhang, Xindi Shang, Wenzhuo Yang, Huan Xu, Huanbo Luan, and Tat-Seng Chua. 2016. Online collaborative learning for open-vocabulary visual classifiers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 2809–2817.

Interactive Learning of Grounded Verb Semantics towards Human-Robot Communication

Lanbo She and **Joyce Y. Chai**
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824, USA
{shelanbo, jchai}@cse.msu.edu

Abstract

To enable human-robot communication and collaboration, previous works represent grounded verb semantics as the potential change of state to the physical world caused by these verbs. Grounded verb semantics are acquired mainly based on the parallel data of the use of a verb phrase and its corresponding sequences of primitive actions demonstrated by humans. The rich interaction between teachers and students that is considered important in learning new skills has not yet been explored. To address this limitation, this paper presents a new interactive learning approach that allows robots to proactively engage in interaction with human partners by asking good questions to learn models for grounded verb semantics. The proposed approach uses reinforcement learning to allow the robot to acquire an optimal policy for its question-asking behaviors by maximizing the long-term reward. Our empirical results have shown that the interactive learning approach leads to more reliable models for grounded verb semantics, especially in the noisy environment which is full of uncertainties. Compared to previous work, the models acquired from interactive learning result in a 48% to 145% performance gain when applied in new situations.

1 Introduction

In communication with cognitive robots, one of the challenges is that robots do not have sufficient linguistic or world knowledge as humans do. For example, if a human asks a robot to *boil the water* but the robot has no knowledge what this verb phrase means and how this verb phrase relates to its own actuator, the robot will not be able to execute this command. Thus it is important for robots to continuously learn the meanings of new verbs and how the verbs are grounded to its underlying action representations from its human partners.

To support learning of grounded verb semantics, previous works (She et al., 2014; Misra et al., 2015; She and Chai, 2016) rely on multiple instances of human demonstrations of corresponding actions. From these demonstrations, robots capture the state change of the environment caused by the actions and represent verb semantics as the desired goal state. One advantage of such state-based representation is that, when robots encounter the same verbs/commands in a new situation, the desired goal state will trigger the action planner to automatically plan a sequence of primitive actions to execute the command.

While the state-based verb semantics provides an important link to connect verbs to the robot's actuator, previous works also present several limitations. First of all, previous approaches were developed under the assumption of perfect perception of the environment (She et al., 2014; Misra et al., 2015; She and Chai, 2016). However, this assumption does not hold in real-world situated interaction. The robot's representation of the environment is often incomplete and error-prone due to its limited sensing capabilities. Thus it is not clear whether previous approaches can scale up to handle noisy and incomplete environment.

Second, most previous works rely on multiple demonstration examples to acquire grounded verb models. Each demonstration is simply a sequence of primitive actions associated with a verb. No other type of interaction between humans and robots is explored. Previous cognitive studies (Bransford et al., 2000) on how people learn have shown that social interaction (e.g., conver-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1634–1644
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1150

sation with teachers) can enhance student learning experience and improve learning outcomes. For robotic learning, previous work (Cakmak and Thomaz, 2012) has also demonstrated the necessity of question answering in the learning process. Thus, in our view, interactive learning beyond demonstration of primitive actions should play a vital role in the robot's acquisition of more reliable models of grounded verb semantics. This is especially important because the robot's perception of the world is noisy and incomplete, human language can be ambiguous, and the robot may lack the relevant linguistic or world knowledge during the learning process.

To address these limitations, we have developed a new interactive learning approach where robots actively engage with humans to acquire models of grounded verb semantics. Our approach explores the space of interactive question answering between humans and robots during the learning process. In particular, motivated by previous work on robot learning (Cakmak and Thomaz, 2012), we designed a set of questions that are pertinent to verb semantic representations. We further applied reinforcement learning to learn an optimal policy that guides the robot in deciding when to ask what questions. Our empirical results have shown that this interactive learning process leads to more reliable representations of grounded verb semantics, which contribute to significantly better action performance in new situations. When the environment is noisy and uncertain (as in a realistic situation), the models acquired from interactive learning result in a performance gain between 48% and 145% when applied in new situations. Our results further demonstrate that the interaction policy acquired from reinforcement learning leads to the most efficient interaction and the most reliable verb models.

2 Related Work

To enable human-robot communication and collaboration, recent years have seen an increasing amount of works which aim to learn semantics of language that are grounded to agents' perception (Gorniak and Roy, 2007; Tellex et al., 2014; Kim and Mooney, 2012; Matuszek et al., 2012a; Liu et al., 2014; Liu and Chai, 2015; Thomason et al., 2015, 2016; Yang et al., 2016; Gao et al., 2016) and action (Matuszek et al., 2012b; Artzi and Zettlemoyer, 2013; She et al., 2014; Misra et al., 2014, 2015; She and Chai, 2016). Specifically for verb semantics, recent works explored the connection between verbs and action planning (She et al., 2014; Misra et al., 2014, 2015; She and Chai, 2016), for example, by representing grounded verbs semantics as the desired goal state of the physical world that is a result of the corresponding actions. Such representations are learned based on example actions demonstrated by humans. Once acquired, these representations will allow agents to interpret verbs/commands issued by humans in new situations and apply action planning to execute actions. Given its clear advantage in connecting verbs with actions, our work also applies the state-based representation for verb semantics. However, we have developed a new approach which goes beyond learning from demonstrated examples by exploring how rich interaction between humans and agents can be used to acquire models for grounded verb semantics.

This approach was motivated by previous cognitive studies (Bransford et al., 2000) on how people learn as well as recent findings on robot skill learning (Cakmak and Thomaz, 2012). One of the principles for human learning is that "learning is enhanced through socially supported interactions". Studies have shown that social interaction with teachers and peers (e.g., substantive conversation) can enhance student learning experience and improve learning outcomes. In recent work on interactive robot learning of new skills (Cakmak and Thomaz, 2012), researchers identified three types of questions that can be used by a human/robot student to enhance learning outcomes: 1) *demonstration query* (i.e., asking for a full or partial demonstration of the task), 2) *label query* (i.e., asking whether an execution is correct), and 3) *feature query* (i.e., asking for a specific feature or aspect of the task). Inspired by these previous findings, our work explores interactive learning to acquire grounded verb semantics. In particular, we aim to address when to ask what questions during interaction to improve learning.

3 Acquisition of Grounded Verb Semantics

This section gives a brief review on acquisition of grounded verb semantics and illustrates the differences between previous approaches and our approach using interactive learning.

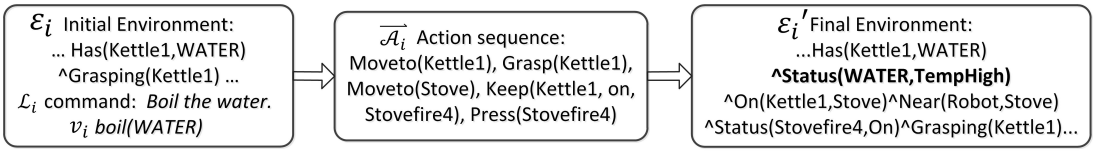


Figure 1: An example of acquiring state-based representation for verb semantics based on an initial environment $\mathcal{E}_i$, and a language command $\mathcal{L}_i$, the primitive action sequence $\overrightarrow{\mathcal{A}_i}$ demonstrated by the human, and the final environment $\mathcal{E}_i'$ that results from the execution of $\overrightarrow{\mathcal{A}_i}$ in $\mathcal{E}_i$.

3.1 State-based Representation

As shown in Figure 1, the verb semantics (e.g., *boil(x)*) is represented by the goal state (e.g., $Status(x, TempHigh)$) which is the result of the demonstrated primitive actions. Given the verb phrase *boil the water* (i.e., $\mathcal{L}_i$), the human teaches the robot how to accomplish the corresponding action based on a sequence of primitive actions $\overrightarrow{\mathcal{A}_i}$. By comparing the final environment $\mathcal{E}_i'$ with the initial environment $\mathcal{E}_i$, the robot is able to identify the state change of the environment, which becomes a hypothesis of goal state to represent verb semantics. Compared to procedure-based representations, the state-based representation supports automated planning at the execution time. It is environment-independent and more generalizable. In (She and Chai, 2016), instead of one hypothesis, it maintains a specific-to-general hypothesis space as shown in Figure 2 to capture all goal hypotheses of a particular verb frame. Specifically, it assumes that one verb frame may lead to different outcomes under different environments, where each possible outcome is represented by one node in the hierarchical graph and each node is a conjunction of multiple atomic fluents. [1]

Given a language command (i.e., a verb phrase), a robot will engage in the following processes:

- **Execution.** In this process, the robot will select a hypothesis from the space of hypotheses that is most relevant to the current situation and use the corresponding goal state to plan for actions to execute.

- **Learning.** When the robot fails to select a hypothesis or fails to execute the action, it will ask the human for a demonstration. Based on the demonstrated actions, the robot will learn a new representation (i.e., new nodes) and update the hypothesis space.

[1] In this work, we assume the set of atomic fluents representing environment state are given and do not address the question of whether these predicates are adequate to represent a domain.

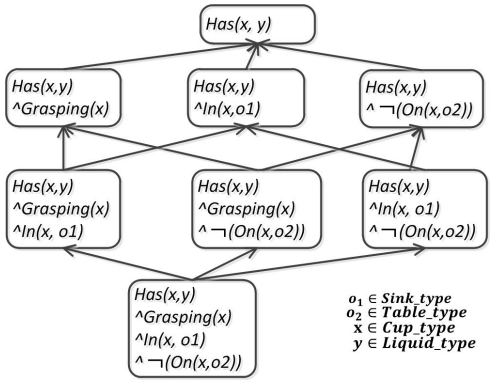


Figure 2: An example hypothesis space for the verb frame $fill(x, y)$.

3.2 Noisy Environment

Probabilistic Environment:
...^Has(Kettle1,Water) 0.64 ^Grasping(Kettle1) 0.91
^Status(Kettle1,HighTemp) 0.95 ^On(Kettle1,Stove) 0.2
^Near(Robot,Stove) 0.43 ^Status(Stovefire4,On) 0.6 ^...

Figure 3: An example probabilistic sensing result.

Previous works represent the environment $\mathcal{E}_i$ as a conjunction of grounded state fluents. Each fluent consists of a predicate and one or more arguments (i.e., objects in the physical world, or object status), representing one aspect of the perceived environment. An example of a fluent is "$Has(Kettle_1, WATER)$" meaning object $Kettle_1$ has some water inside, where Has is the predicate, and $Kettle_1$ and $WATER$ are arguments. The set of fluents include the status of the robot (e.g., $Grasping(Kettle_1)$), the status of different objects (e.g., $Status(WATER, TempHigh)$), and relations between objects (e.g., $On(Kettle_1, Stove)$). One limitation of the previous works is that the envi-

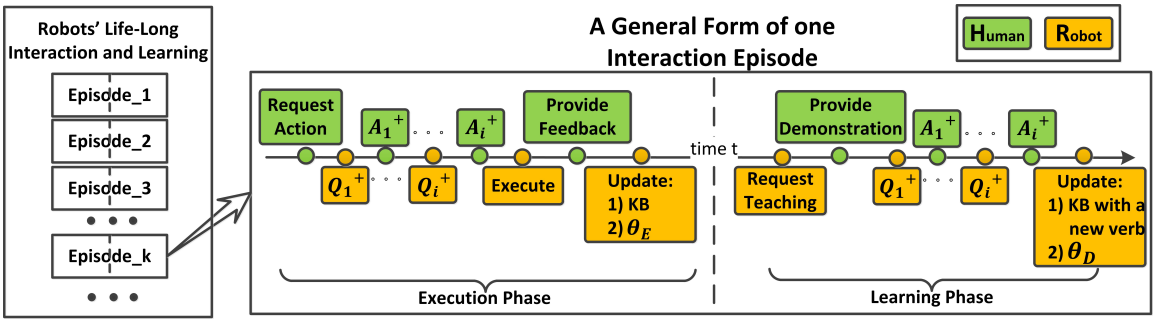


Figure 4: A general framework of robot interactive learning. KB stands for knowledge base, θ_E stands for Interaction Strategy for Execution, and θ_D stands for Interaction Strategy for Learning.

Only demonstration(She and Chai, 2016)	**Interactive learning (this work)**
Execution Example H1: Boil the water. R1: (*select a hypothesis and execute the action accordingly*) H2: (*provide true/false feedback on whether the execution is correct*)	**Execution Example** H1: Boil the water. R1: I see a silver object on the table, is that a pot? H2: Yes. R2: Should I leave the pot on the stove after boil? H3: Yes. R3: Ok. (*select a hypothesis to execute*) H4: Thank you.
Learning Example H3: Boil the water. R3: I don't know how to boil, could you show me an example? H4: (*human shows the robot how to achieve the corresponding action through a sequence of primitive actions*) R4: (*indicate the acceptance and induce a hypothesis to represent verb semantics*)	**Learning Example** H5: Boil the water. R5: I don't know how to boil, could you show me an example? H6: Ok. (*demonstrate the task, at the end put the pot on a heat pad*) R6: So after boiling something, should I put it on a heat pad? H7: No, you don't have to. R7: I see. (*induce goal state hypothesis*)

Table 1: Examples to show differences between learning through demonstrations as in the previous works (She and Chai, 2016) and the proposed learning from interaction.

ronment has a perfect, deterministic representation, as shown in Figure 1. This is clearly not the case in the realistic physical world.

In reality, given limitations of sensor capabilities, the environment representation is often partial, error prone, and full of uncertainties. Figure 3 shows an example of a more realistic representation where each fluent comes with a confidence between 0 and 1 to indicate how likely that particular fluent can be detected in the current environment. Thus, it is unclear whether the previous work is able to handle representations with uncertainties. Our interactive learning approach aims to address these uncertainties through interactive question answering with human partners.

4 Interactive Learning

4.1 Framework of Interactive Learning

Figure 4 shows a general framework for interactive learning of action verbs. It aims to support a life-long learning cycle for robots, where the robot can continuously (1) engage in collaboration and communication with humans based on its existing knowledge; (2) acquire new verbs by learning from humans and experiencing the change of the world (i.e., grounded verb semantics as in this work); and (3) learn how to interact (i.e., update interaction policies). The lifelong learning cycle is composed by a sequence of interactive learning episodes (Episode 1, 2...) where each episode consists of either an execution phase or a learning phase or both.

The execution phase starts with a human request for action (e.g., *boil the water*). According to its interaction policy, the robotic agent may choose to ask one or more questions (i.e., Q_i^+) and wait for human answers (i.e., A_i^+), or select a hypothesis from its existing knowledge base to execute the command (i.e., *Execute*). With the human feedback of the execution, the robot can update its interaction policy and existing knowledge.

In the learning phase, the robot can initiate the learning by requesting a demonstration from the human. After the human performs the task,

the robotic agent can either choose to update its knowledge if it feels confident, or it can choose to ask the human one or more questions before updating its knowledge.

4.2 Examples of Interactive Learning

Table 1 illustrates the differences between the previous approach that acquires verb models based solely on demonstrations and our current work that acquires models based on interactive learning. As shown in Table 1, under the *demonstration* setting, humans only provide a demonstration of primitive actions and there's no interactive question answering. In the *interactive learning* setting, the robot can proactively choose to ask questions regarding the uncertainties either about the environment (e.g., R1), the goal (e.g., R2), or the demonstrations (e.g., R6). Our hypothesis is that rich interactions based on question answering will allow the robot to learn more reliable models for grounded verb semantics, especially in a noisy environment.

Then the question is how to manage such interaction: when to ask and what questions to ask to most efficiently acquire reliable models and apply them in execution. Next we describe the application of reinforcement learning to manage interactive question answering for both the execution phase and the learning phase.

4.3 Formulation of Interactive Learning

Markov Decision Process (MDP) and its closely related Reinforcement Learning (RL) have been applied to sequential decision-making problems in dynamic domains with uncertainties, e.g., dialogue/interaction management (Singh et al., 2002; Paek and Pieraccini, 2008; Williams and Zweig, 2016), mapping language commands to actions (Branavan et al., 2009), interactive robot learning (Knox and Stone, 2011), and interactive information retrieval (Li et al., 2017). In this work, we formulate the choice of when to ask what questions during interaction as a sequential decision-making problem and apply reinforcement learning to acquire an optimal policy to manage interaction.

Specifically, each of the execution and learning phases is governed by one policy (i.e., θ_E and θ_D), which is updated by the reinforcement learning algorithm. The use of RL intends to obtain optimal policies that can lead to the highest long-term reward by balancing the cost of interaction (e.g., the length of interaction and difficulties of questions) and the quality of the acquired models. The reinforcement formulation for both the execution phase and the learning phase are described below.

State For the execution phase, each state $s_e \in S_E$ is a five tuple: $s_e = <l, e, KB, Grd, Goal>$. l is a language command, including a verb and multiple noun phrases extracted by the Stanford parser. For example, the command "*Microwave the ramen*" is represented as $l = microwave(ramen)$. The environment e is a probabilistic representation of the currently perceived physical world, consisting of a set of grounded fluents and the confidence of perceiving each fluent (an example is shown in Figure 3). KB stands for the existing knowledge of verb models. Grd accounts for the agent's current belief of object grounding: the probability of each noun in the l being grounded to different objects. $Goal$ represents the agent's belief of different goal state hypotheses of the current command. Within one interaction episode, command l and knowledge KB will stay the same, while e, Grd, and $Goal$ may change accordingly due to interactive question answering and robot actions. In the execution phase, Grd and $Goal$ are initialized with existing knowledge of learned verb models. For the learning phase, a state $s_d \in S_D$ is a four tuple: $s_d = <l, e_{start}, e_{end}, Grd>$. e_{start} and e_{end} stands for the environment before the demonstration and after the demonstration.

Action Motivated by previous studies on how humans ask questions while learning new skills (Cakmak and Thomaz, 2012), the agent's question set includes two categories: yes/no questions and wh- questions. These questions are designed to address ambiguities in noun phrase grounding, uncertain environment sensing, and goal states. They are domain independent in nature. For example, one of the questions is np_grd_ynq(n, o). It is a yes/no question asking whether the noun phrase n refers to an object o (e.g., "*I see a silver object, is that the pot?*"). Other questions are env_pred_ynq(p) (i.e., whether a fluent p is present in the environment; e.g., "*Is the microwave door open?*") and goal_pred_ynq(p) (i.e., whether a predicate p should be part of the goal; "*Should the pot be on a pot stand?*"). Table 2 lists all the actions available in the execution and learning phases. The select_hypo action (i.e., select a goal hypothesis to execute) is only for the execution. Ideally, after asking questions, the agent should be more likely to select a goal hy-

Action Name	Explanation	Question Example	Reward
1. **np_grd_whq**(n)	Ask for the grounding of a np.	"*Which is the cup, can you show me?*"	-6.5[1]
2. **np_grd_ynq**(n, o)	Confirm the grounding of a np.	"*I see a silver object, is that the pot?*"	-1.0 / -2.0
3. **env_pred_ynq**(p)	Confirm a predicate in current environment.	"*Is the microwave door open?*"	-1.0 / -2.0
4. **goal_pred_ynq**(p)	Confirm whether a predicate p should be in the final environment.	"*Is it true the pot should be on the counter?*"	-1.0 / -2.0
5. **select_hypo**(h)	Choose a hypothesis to use as goal and execute.		100 / -2.0
6. **bulk_np_grd_ynq**(n, o)	Confirm the grounding of multiple nps.	"*I think the pot is the red object and milk is in the white box, am I right?*"	-3.0 / -6.0[2]
7. **pred_change_ynq**(p)	Ask whether a predicate p has been changed by the action demonstration.	"*The pot is on a stand after the action, is that correct?*"	-1.0 / -2.0
8. **include_fluent**($\wedge p$)	Include $\wedge p$ into the goal state representation. Update the verb semantic knowledge.		100 / -2.0

Table 2: The action space for reinforcement learning, where n stands for a noun phrase, o a physical object, p a fluent representation of the current state of the world, h a goal hypothesis. Action 1 and 2 are shared by both the execution and learning phases. Action 3, 4, 5 are for the execution phase, and 6, 7, 8 are only used for the learning phase. -1.0/-2.0 are typically used for yes/no questions. When the human answers the question with a "yes", the reward is -1.0, otherwise it's -2.0.

pothesis that best describes the current situation. For the learning phase, the include_fluent($\wedge p$) action forms a goal hypothesis by conjoining a set of fluents ps where each p should have high probability of being part of the goal.

Transition The transition function takes action a in state s, and gives the next state s' according to human feedback. Note that the command l does not change during interaction. But the agent's belief of environment e, object grounding Grd, and goal hypotheses $Goal$ is changed according to the questions and human answers. For example, suppose the agent asks whether noun phrase n refers to the object o, if the human confirms it, the probability of n being grounded to o becomes 1.0, otherwise it will become 0.0.

Reward Finding a good reward function is a hard problem in reinforcement learning. Our current approach has followed the general practice in the spoken dialogue community (Schatzmann et al., 2006; Fang et al., 2014; Su et al., 2016). The immediate robot questions are assigned small costs to favor shorter and more efficient interaction. Furthermore, motivated by how humans ask questions (Cakmak and Thomaz, 2012), yes/no questions are easier for a human to answer than the open questions (e.g., wh-questions) and thus are given smaller costs. A large positive reward is given at the end of interaction when the task is completed successfully. Detailed reward assignment for different actions are shown in Table 2.

Algorithm 1: Policy learning. The execution and learning phases share the same learning process, but with different state s, action a spaces, and feature vectors ϕ. The e_{end} is only available to the learning phase.

Input : e, l (, e_{end});
Feature function ϕ;
Old policy θ (i.e., a weight vector)
Verb Goal States Hypotheses $\mathcal{H}$;
Initialize : state s initialized with e, l (, e_{end});
first action $a \sim P(a|s;\theta)$ with ϵ greedy
while *s is not terminal* **do**
  Take action a, receive reward r;
  $s' = T(s, a)$;
  Choose $a' \sim P(a'|s';\theta)$ with ϵ greedy;
  $\delta \leftarrow r + \gamma \cdot \theta^T \cdot \phi(s', a') - \theta^T \cdot \phi(s, a)$;
  $\theta \leftarrow \theta + \delta \cdot \eta \cdot \phi(s, a)$;
end
if *s terminates with positive feedback* **then**
  Update $\mathcal{H}$;
end
Output : Updated $\mathcal{H}$ and θ.

Learning The *SARSA* algorithm with linear function approximation is utilized to update policies θ_E and θ_D (Sutton and Barto, 1998). Specifically, the objective of training is to learn an optimal value function $Q(s, a)$ (i.e., the expected cu-

[1] According to the study in (Cakmak and Thomaz, 2012), the frequency of y/n questions used by humans is about 6.5 times the frequency of open questions (wh question), which motivates our assignment of -6.5 to wh questions.

[2] bulk_np_grd_ynq asks multiple object grounding all at once. This is harder to answer than asking for a single np. Therefore, its cost is assigned three times of the other yes/no questions.

Features shared by both phases
If a is a np_grd_whq(n).
The entropy of candidate groundings of n.
If n has more than 4 grounding candidates.
If a is a np_grd_ynq(n, o).
The probability of n grounded to o.
Additional Features specific for the Execution phase
If a is a select_hypo(h) action.
The probability of hypo h not satisfied in current environment.
Similarity between the ns used by command l and the commands from previous experiences.
Additional Features specific for the Learning phase
If a is a pred_change_ynq(p).
The probability of p been changed by demo.

Table 3: Example features used by the two phases. a stands for action. Other notations are the same as used in Table 2. The"If" features are binary, and the other features are real-valued.

mulative reward of taking action a in a state s). This value function is approximated by a linear function $Q(s,a) = \theta^{\intercal} \cdot \phi(s,a)$, where $\phi(s,a)$ is a feature vector and θ is a weight updated during training. Details of the algorithm is shown in Algorithm 1. During testing, the agent can take an action a that maximizes the Q value at a state s.

Feature Example features used by the two phases are listed in Table 3. These features intend to capture different dimensions of information such as specific types of questions, how well noun phrases are grounded to the environment, uncertainties of the environment, and consistencies between a hypothesis and the current environment.

5 Evaluation

5.1 Experiment Setup

Dataset. To evaluate our approach, we utilized the benchmark made available by (Misra et al., 2015). Individual language commands and corresponding action sequences are extracted similarly as (She and Chai, 2016). This dataset includes common tasks in the kitchen and living room domains, where each data instance comes with a language command (e.g., "*boil the water*", "*throw the beer into the trashcan*") and the corresponding sequence of primitive actions. In total, there are 979 instances, including 75 different verbs and 215 different noun phrases. The length of primitive action sequences range from 1 to 51 with an average of 4.82 (+/-4.8). We divided the dataset into three groups: (1) 200 data instances were used by reinforcement learning to acquire optimal interaction policies; (2) 600 data instances were used by different approaches (i.e., previous approaches and our interactive learning approach) to acquire grounded verb semantics models; and (3) 179 data instances were used as testing data to evaluate the learned verb models. The performance on applying the learned models to execute actions for the testing data is reported.

To learn interaction policies, a simulated human model is created from the dataset (Schatzmann et al., 2006) to continuously interact with the robot learner[3]. This simulated user can answer the robot's different types of questions and make decisions on whether the robot's execution is correct. During policy learning, one data instance can be used multiple times. At each time, the interaction sequence is different due to *exploitation and exploration* in RL in selecting the next action. The RL discount factor γ is set to 0.99, the ϵ in ϵ-greedy is 0.1, and the learning rate is 0.01.

Noisy Environment Representation. The original data provided by (Misra et al., 2015) is based on the assumption that environment sensing is perfect and deterministic. To enable incomplete and noisy environment representation, for each fluent (e.g., $grasping(Cup_3)$, $near(robot_1, Cup_3)$) in the original data, we independently sampled a confidence value to simulate the likelihood that a particular fluent can be detected correctly from the environment. We applied the following four different variations in sampling the confidence values, which correspond to different levels of sensor reliability.

(1) PerfectEnv represents the most reliable sensor. If a fluent is true in the original data, its sampled confidence is 1, and 0 otherwise.

(2) NormStd3 represents a relatively reliable sensor. For each fluent in the original environment, a confidence is sampled according to a normal distribution $\mathcal{N}(1, 0.3^2)$ with an interval [0,1]. This distribution has a large probability of sampling a number larger than 0.5, meaning the corresponding fluent is still more likely to be true.

(3) NormStd5 represents a less reliable sensor. The sampling distribution is $\mathcal{N}(1, 0.5^2)$, which has a larger probability of generating a number smaller than 0.5 compared to *NormStd3*.

[3]In our future work, interacting with real humans will be conducted through Amazon Mechanical Turk. And the policies acquired with a simulated user in this work will be used as initial policies.

(4) UniEnv represents an unreliable sensor. Each number is sampled with a uniform distribution between 0 and 1. This means the sensor works randomly. A fluent has a equal change to be true or false no matter what the true environment is.

Evaluation Metrics. We used the same evaluation metrics as in the previous works (Misra et al., 2015; She and Chai, 2016) to evaluate the performance of applying the learned models to testing instances on action planning.

- ***IED***: Instruction Editing Distance. This is a number between 0 and 1 measuring the similarity between the predicted action sequence and the ground-truth action sequence. ***IED*** equals 1 if the two sequences are exactly the same.
- ***SJI***: State Jaccard Index. This is a number between 0 and 1 measuring the similarity between the predicted and the ground-truth state changes. ***SJI*** equals 1 if action planning leads to exactly the same state change as in the ground-truth.

Configurations. To understand the role of interactive learning in model acquisition and action planning, we first compared the interactive learning approach with the previous leading approach (presented as *She16*). To further evaluate the interaction policies acquired by reinforcement learning, we also compared the learned policy (i.e., ***RLPolicy***) with the following two baseline policies:

- ***RandomPolicy*** which randomly selects questions to ask during interaction.
- ***ManualPolicy*** which continuously asks for yes/no confirmations (i.e., object grounding questions ($GroundQ$), environment questions ($EnvQ$), goal prediction questions ($GoalQ$)) until there's no more questions before making a decision on model acquisition or action execution.

5.2 Results

5.2.1 The Effect of Interactive Learning

Table 4 shows the performance comparison on the testing data between the previous approach *She16* and our interactive learning approach based on environment representations with different levels of noise. The verb models acquired by interactive learning perform better consistently across all four environment conditions. When the environment becomes noisy (i.e., *NormStd3*, *NormStd5*, and *UniEnv*), the performance of *She16* that only relies on demonstrations decreases significantly. While the interactive learning improves the performance under the perfect environment condition, its effect in noisy environment is more remarkable. It leads to a significant performance gain between 48% and 145%. These results validate our hypothesis that interactive question answering can help to alleviate the problem of uncertainties in environment representation and goal prediction.

	She16		*RL policy*		% improvement	
	IED	*SJI*	*IED*	*SJI*	*IED*	*SJI*
PerfectEnv	0.430	0.426	0.453	0.468	5.3%*	9.9%*
NormStd3	0.284	0.273	0.420	0.431	47.9%*	57.9%*
NormStd5	0.172	0.168	0.392	0.411	127.9%*	144.6%*
UniEnv	0.168	0.163	0.332	0.347	97.6%*	112.9%*

Table 4: Performance comparison between *She16* and our interactive learning based on environment representations with different levels of noise. All the improvements (marked *) are statistically significant ($p < 0.01$).

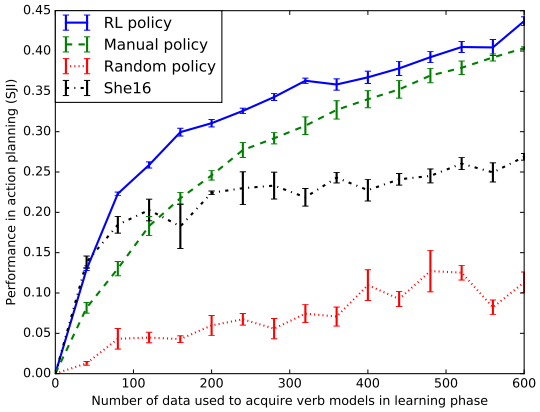


Figure 5: Performance (SJI) comparison by applying models acquired based on different interaction policies to the testing data.

Figure 5 shows the performance of the various learned models on the testing data, based on a varying number of training instances and different interaction policies. The interactive learning guided by the policy acquired from RL outperforms the previous approach *She16*. The RL policy slightly outperforms interactive learning using manually defined policy (i.e., *ManualPolicy*). However, as shown in the next section, the *Man-*

	Average number of questions							Performance	
	Learning Phase			Execution Phase					
	GroundQ	*EnvQ*	*TotalQ*	*GroundQ*	*EnvQ*	*GoalQ*	*TotalQ*	*IED*	*SJI*
RLPolicy	2.130* +/-0.231	2.615* +/-0.317	4.746* +/-0.307	0.383* +/-0.137	0.650* +/-0.366	2.626 +/-0.331	3.665* +/-0.469	0.420 +/-0.015	0.430* +/-0.018
ManualPolicy	2.495 +/-0.025	5.338 +/-0.008	7.833 +/-0.025	1.236 +/-0.002	3.202 +/-0.012	2.353 +/-0.023	6.792 +/-0.025	0.406 +/-0.002	0.404 +/-0.004
RandomPolicy	0.545 +/-0.016	0.368 +/-0.033	0.913 +/-0.040	0.678 +/-0.055	0.081 +/-0.030	0.151 +/-0.024	0.909 +/-0.018	0.114 +/-0.025	0.113 +/-0.029

Table 5: Comparison between different policies including the average number (and standard deviation) of different types of questions asked during the execution phase and the learning phase respectively, and the performance on action planning for the testing data. The results are based on the noisy environment sampled by *NormStd3*. * indicates statistically significant difference ($p < 0.05$) comparing *RLPolicy* with *ManualPolicy*.

ualPolicy results in much longer interaction (i.e., more questions) than the RL acquired policy.

5.2.2 Comparison of Interaction Policies

Table 5 compares the performance of different interaction policies. It shows the average number of questions asked under different policies. It is not surprising the *RandomPolicy* has the worst performance. For the *ManualPolicy*, its performance is similar to the *RLPolicy*. However, the average interaction length of *ManualPolicy* is 6.792, which is much longer than the *RLPolicy* (which is 3.127). These results further demonstrate that the policy learned from RL enables efficient interactions and the acquisition of more reliable verb models.

6 Conclusion

Robots live in a noisy environment. Due to the limitations in their external sensors, their representations of the shared environment can be error prone and full of uncertainties. As shown in previous work (Mourão et al., 2012), learning action models from the noisy and incomplete observation of the world is extremely challenging. The same problem applies to the acquisition of verb semantics that are grounded to the perceived world. To address this problem, this paper presents an interactive learning approach which aims to handle uncertainties of the environment as well as incompleteness and conflicts in state representation by asking human partners intelligent questions. The interaction strategies are learned through reinforcement learning. Our empirical results have shown a significant improvement in model acquisition and action prediction. When applying the learned models in new situations, the models acquired through interactive learning leads to over 140% performance gain in noisy environment.

The current investigation also has several limitations. As in previous works, we assume the world can be described by a closed set of predicates. This causes significant simplification for the physical world. One of the important questions to address in the future is how to learn new predicates through interaction with humans. Another limitation is that the current utility function is learned based on a set of pre-identified features. Future work can explore deep neural network to alleviate feature engineering.

As cognitive robots start to enter our daily lives, data-driven approaches to learning may not be possible in new situations. Human partners who work side-by-side with these cognitive robots are great resources that the robots can directly learn from. Recent years have seen an increasing amount of work on task learning from human partners (Saunders et al., 2006; Chernova and Veloso, 2008; Cantrell et al., 2012; Mohan et al., 2013; Asada et al., 2009; Mohseni-Kabir et al., 2015; Nejati et al., 2006; Liu et al., 2016). Our future work will incorporate interactive learning of verb semantics with task learning to enable autonomy that can learn by communicating with humans.

Acknowledgments

This work was supported by the National Science Foundation (IIS-1208390 and IIS-1617682) and the DARPA SIMPLEX program under a subcontract from UCLA (N66001-15-C-4035). The authors would like to thank Dipendra K. Misra and colleagues for providing the evaluation data, and the anonymous reviewers for valuable comments.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* Volume1(1):49–62.

Minoru Asada, Koh Hosoda, Yasuo Kuniyoshi, Hiroshi Ishiguro, Toshio Inui, Yuichiro Yoshikawa, Masaki Ogino, and Chisato Yoshida. 2009. Cognitive developmental robotics: A survey. *IEEE Transactions on Autonomous Mental Development* 1(1):12–34.

S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '09, pages 82–90.

John D. Bransford, Ann L. Brown, and Rodney R. Cocking. 2000. *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*. National Academy Press., Washington, DC.

Maya Cakmak and Andrea L. Thomaz. 2012. Designing robot learners that ask good questions. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction*. ACM, New York, NY, USA, HRI '12, pages 17–24.

R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz. 2012. Tell me when and why to do it! run-time planner model updates via natural language instruction. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction*. ACM, New York, NY, USA, HRI '12, pages 471–478.

Sonia Chernova and Manuela Veloso. 2008. Teaching multi-robot coordination using demonstration of communication and state sharing. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems, pages 1183–1186.

Rui Fang, Malcolm Doering, and Joyce Y. Chai. 2014. Collaborative models for referring expression generation in situated dialogue. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'14, pages 1544–1550.

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Y. Chai. 2016. Physical causality of action verbs in grounded language understanding. In *ACL (1)*. The Association for Computer Linguistics.

P. Gorniak and D. Roy. 2007. Situated language understanding as filtering perceived affordances. In *Cognitive Science*, volume 31(2), pages 197–231.

Joohyun Kim and Raymond J. Mooney. 2012. Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL '12)*. Jeju Island, Korea, pages 433–444.

W. Bradley Knox and Peter Stone. 2011. Understanding human teaching modalities in reinforcement learning environments: A preliminary report. In *IJCAI 2011 Workshop on Agents Learning Interactively from Human Teachers (ALIHT)*.

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017. Learning through dialogue interactions. In *ICLR*.

Changsong Liu and Joyce Y. Chai. 2015. Learning to mediate perceptual differences in situated human-robot dialogue. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'15, pages 2288–2294.

Changsong Liu, Lanbo She, Rui Fang, and Joyce Y. Chai. 2014. Probabilistic labeling for efficient referential grounding based on collaborative discourse. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 13–18.

Changsong Liu, Shaohua Yang, Sari Saba-Sadiya, Nishant Shukla, Yunzhong He, Song-chun Zhu, and Joyce Y. Chai. 2016. Jointly learning grounded task structures from language instruction and visual demonstration. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1482–1492.

Cynthia Matuszek, Nicholas Fitzgerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012a. A joint model of language and perception for grounded attribute learning. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. ACM, New York, NY, USA, pages 1671–1678.

Cynthia Matuszek, Evan Herbst, Luke S. Zettlemoyer, and Dieter Fox. 2012b. Learning to parse natural language commands to a robot control system. In Jaydev P. Desai, Gregory Dudek, Oussama Khatib, and Vijay Kumar, editors, *ISER*. Springer, volume 88 of *Springer Tracts in Advanced Robotics*, pages 403–415.

Dipendra K Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2014. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *Proceedings of Robotics: Science and Systems (RSS), Berkeley, USA* .

Dipendra Kumar Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the*

7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Association for Computational Linguistics, Beijing, China, pages 992–1002.

Shiwali Mohan, James Kirk, and John Laird. 2013. A computational model for situated task learning with interactive instruction. In *Proceedings of ICCM 2013 - 12th International Conference on Cognitive Modeling*.

Anahita Mohseni-Kabir, Charles Rich, Sonia Chernova, Candace L. Sidner, and Daniel Miller. 2015. Interactive hierarchical task learning from a single demonstration. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. ACM, HRI '15, pages 205–212.

Kira Mourão, Luke S. Zettlemoyer, Ronald P. A. Petrick, and Mark Steedman. 2012. Learning STRIPS operators from noisy and incomplete observations. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*. Catalina Island, CA, USA, pages 614–623.

Negin Nejati, Pat Langley, and Tolga Konik. 2006. Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on Machine learning*. ACM, pages 665–672.

Tim Paek and Roberto Pieraccini. 2008. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech Communication* 50(8-9):716–729.

Joe Saunders, Chrystopher L Nehaniv, and Kerstin Dautenhahn. 2006. Teaching robots by moulding behavior and scaffolding the environment. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. ACM, pages 118–125.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowl. Eng. Rev.* 21(2):97–126.

Lanbo She and Joyce Y. Chai. 2016. Incremental acquisition of verb hypothesis space towards physical world interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Y. Chai, and Ning Xi. 2014. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA*. pages 89–97.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research* 16:105–133.

Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2431–2441.

Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Stefanie Tellex, Pratiksha Thaker, Joshua Joseph, and Nicholas Roy. 2014. Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning* 94(2):151–167.

Jesse Thomason, Jivko Sinapov, Maxwell Svetlik, Peter Stone, and Raymond J. Mooney. 2016. Learning multi-modal grounded linguistic semantics by playing "i spy". In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16)*. New York City, pages 3477–3483.

Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. 2015. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*. pages 1923–1929.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269* .

Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y. Chai. 2016. Grounded semantic role labeling. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. pages 149–159.

Multimodal Word Distributions

Ben Athiwaratkun
Cornell University
pa338@cornell.edu

Andrew Gordon Wilson
Cornell University
andrew@cornell.edu

Abstract

Word embeddings provide point representations of words containing useful semantic information. We introduce multimodal word distributions formed from Gaussian mixtures, for multiple word meanings, entailment, and rich uncertainty information. To learn these distributions, we propose an energy-based max-margin objective. We show that the resulting approach captures uniquely expressive semantic information, and outperforms alternatives, such as word2vec skip-grams, and Gaussian embeddings, on benchmark datasets such as word similarity and entailment.

1 Introduction

To model language, we must represent words. We can imagine representing every word with a binary one-hot vector corresponding to a dictionary position. But such a representation contains no valuable semantic information: distances between word vectors represent only differences in alphabetic ordering. Modern approaches, by contrast, learn to map words with similar meanings to nearby points in a vector space (Mikolov et al., 2013a), from large datasets such as Wikipedia. These learned word embeddings have become ubiquitous in predictive tasks.

Vilnis and McCallum (2014) recently proposed an alternative view, where words are represented by a whole probability distribution instead of a deterministic point vector. Specifically, they model each word by a Gaussian distribution, and learn its mean and covariance matrix from data. This approach generalizes any deterministic point embedding, which can be fully captured by the mean vector of the Gaussian distribution. Moreover, the full distribution provides much richer information than point estimates for characterizing words, representing probability mass and uncertainty across a set of semantics.

However, since a Gaussian distribution can have only one mode, the learned uncertainty in this representation can be overly diffuse for words with multiple distinct meanings (polysemies), in order for the model to assign *some* density to any plausible semantics (Vilnis and McCallum, 2014). Moreover, the mean of the Gaussian can be pulled in many opposing directions, leading to a biased distribution that centers its mass mostly around one meaning while leaving the others not well represented.

In this paper, we propose to represent each word with an expressive multimodal distribution, for multiple distinct meanings, entailment, heavy tailed uncertainty, and enhanced interpretability. For example, one mode of the word 'bank' could overlap with distributions for words such as 'finance' and 'money', and another mode could overlap with the distributions for 'river' and 'creek'. It is our contention that such flexibility is critical for both qualitatively learning about the meanings of words, and for optimal performance on many predictive tasks.

In particular, we model each word with a mixture of Gaussians (Section 3.1). We learn all the parameters of this mixture model using a maximum margin energy-based ranking objective (Joachims, 2002; Vilnis and McCallum, 2014) (Section 3.3), where the energy function describes the affinity between a pair of words. For analytic tractability with Gaussian mixtures, we use the inner product between probability distributions in a Hilbert space, known as the expected likelihood kernel (Jebara et al., 2004), as our energy function (Section 3.4). Additionally, we propose transformations for numerical stability and initialization A.2, resulting in a robust, straightforward, and

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1645–1656
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1151

scalable learning procedure, capable of training on a corpus with billions of words in days. We show that the model is able to automatically discover multiple meanings for words (Section 4.3), and significantly outperform other alternative methods across several tasks such as word similarity and entailment (Section 4.4, 4.5, 4.7). We have made code available at `http://github.com/benathi/word2gm`, where we implement our model in Tensorflow (Abadi et. al, 2015).

2 Related Work

In the past decade, there has been an explosion of interest in word vector representations. `word2vec`, arguably the most popular word embedding, uses continuous bag of words and skip-gram models, in conjunction with negative sampling for efficient conditional probability estimation (Mikolov et al., 2013a,b). Other popular approaches use feedforward (Bengio et al., 2003) and recurrent neural network language models (Mikolov et al., 2010, 2011b; Collobert and Weston, 2008) to predict missing words in sentences, producing hidden layers that can act as word embeddings that encode semantic information. They employ conditional probability estimation techniques, including hierarchical softmax (Mikolov et al., 2011a; Mnih and Hinton, 2008; Morin and Bengio, 2005) and noise contrastive estimation (Gutmann and Hyvärinen, 2012).

A different approach to learning word embeddings is through factorization of word co-occurrence matrices such as `GloVe` embeddings (Pennington et al., 2014). The matrix factorization approach has been shown to have an implicit connection with skip-gram and negative sampling Levy and Goldberg (2014). Bayesian matrix factorization where row and columns are modeled as Gaussians has been explored in Salakhutdinov and Mnih (2008) and provides a different probabilistic perspective of word embeddings.

In exciting recent work, Vilnis and McCallum (2014) propose a Gaussian distribution to model each word. Their approach is significantly more expressive than typical point embeddings, with the ability to represent concepts such as *entailment*, by having the distribution for one word (e.g. 'music') encompass the distributions for sets of related words ('jazz' and 'pop'). However, with a unimodal distribution, their approach cannot capture multiple distinct meanings, much like most deterministic approaches.

Recent work has also proposed deterministic embeddings that can capture polysemies, for example through a cluster centroid of context vectors (Huang et al., 2012), or an adapted skip-gram model with an EM algorithm to learn multiple latent representations per word (Tian et al., 2014). Neelakantan et al. (2014) also extends skip-gram with multiple prototype embeddings where the number of senses per word is determined by a non-parametric approach. Liu et al. (2015) learns topical embeddings based on latent topic models where each word is associated with multiple topics. Another related work by Nalisnick and Ravi (2015) models embeddings in infinite-dimensional space where each embedding can gradually represent incremental word sense if complex meanings are observed.

Probabilistic word embeddings have only recently begun to be explored, and have so far shown great promise. In this paper, we propose, to the best of our knowledge, the first probabilistic word embedding that can capture multiple meanings. We use a Gaussian mixture model which allows for a highly expressive distributions over words. At the same time, we retain scalability and analytic tractability with an expected likelihood kernel energy function for training. The model and training procedure harmonize to learn descriptive representations of words, with superior performance on several benchmarks.

3 Methodology

In this section, we introduce our Gaussian mixture (GM) model for word representations, and present a training method to learn the parameters of the Gaussian mixture. This method uses an energy-based maximum margin objective, where we wish to maximize the similarity of distributions of nearby words in sentences. We propose an energy function that compliments the GM model by retaining analytic tractability. We also provide critical practical details for numerical stability and initialization. The code for model training and evaluation is available at `http://github.com/benathi/word2gm`.

3.1 Word Representation

We represent each word w in a dictionary as a Gaussian mixture with K components. Specifically, the distribution of w, f_w, is given by the

density

$$f_w(\vec{x}) = \sum_{i=1}^{K} p_{w,i}\, \mathcal{N}\left[\vec{x}; \vec{\mu}_{w,i}, \Sigma_{w,i}\right] \quad (1)$$

$$= \sum_{i=1}^{K} \frac{p_{w,i}}{\sqrt{2\pi|\Sigma_{w,i}|}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_{w,i})^\top \Sigma_{w,i}^{-1}(\vec{x}-\vec{\mu}_{w,i})},$$

where $\sum_{i=1}^{K} p_{w,i} = 1$. The mean vectors $\vec{\mu}_{w,i}$ represent the location of the i^{th} component of word w, and are akin to the point embeddings provided by popular approaches like `word2vec`. $p_{w,i}$ represents the component probability (mixture weight), and $\Sigma_{w,i}$ is the component covariance matrix, containing uncertainty information. Our goal is to learn all of the model parameters $\vec{\mu}_{w,i}, p_{w,i}, \Sigma_{w,i}$ from a corpus of natural sentences to extract semantic information of words. Each Gaussian component's mean vector of word w can represent one of the word's distinct meanings. For instance, one component of a polysemous word such as 'rock' should represent the meaning related to 'stone' or 'pebbles', whereas another component should represent the meaning related to music such as 'jazz' or 'pop'. Figure 1 illustrates our word embedding model, and the difference between multimodal and unimodal representations, for words with multiple meanings.

3.2 Skip-Gram

The training objective for learning $\theta = \{\vec{\mu}_{w,i}, p_{w,i}, \Sigma_{w,i}\}$ draws inspiration from the continuous skip-gram model (Mikolov et al., 2013a), where word embeddings are trained to maximize the probability of observing a word given another nearby word. This procedure follows the *distributional hypothesis* that words occurring in natural contexts tend to be semantically related. For instance, the words 'jazz' and 'music' tend to occur near one another more often than 'jazz' and 'cat'; hence, 'jazz' and 'music' are more likely to be related. The learned word representation contains useful semantic information and can be used to perform a variety of NLP tasks such as word similarity analysis, sentiment classification, modelling word analogies, or as a preprocessed input for complex system such as statistical machine translation.

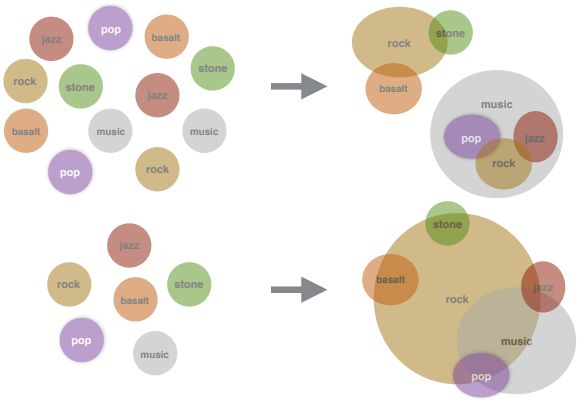


Figure 1: **Top:** A Gaussian Mixture embedding, where each component corresponds to a distinct meaning. Each Gaussian component is represented by an ellipsoid, whose center is specified by the mean vector and contour surface specified by the covariance matrix, reflecting subtleties in meaning and uncertainty. On the left, we show examples of Gaussian mixture distributions of words where Gaussian components are randomly initialized. After training, we see on the right that one component of the word 'rock' is closer to 'stone' and 'basalt', whereas the other component is closer to 'jazz' and 'pop'. We also demonstrate the entailment concept where the distribution of the more general word 'music' encapsulates words such as 'jazz', 'rock', 'pop'. **Bottom:** A Gaussian embedding model (Vilnis and McCallum, 2014). For words with multiple meanings, such as 'rock', the variance of the learned representation becomes unnecessarily large in order to assign some probability to both meanings. Moreover, the mean vector for such words can be pulled between two clusters, centering the mass of the distribution on a region which is far from certain meanings.

3.3 Energy-based Max-Margin Objective

Each sample in the objective consists of two pairs of words, (w, c) and (w, c'). w is sampled from a sentence in a corpus and c is a nearby word within a context window of length ℓ. For instance, a word w = 'jazz' which occurs in the sentence 'I listen to jazz music' has context words ('I', 'listen', 'to' , 'music'). c' is a negative context word (e.g. 'airplane') obtained from random sampling.

The objective is to maximize the energy between words that occur near each other, w and c, and minimize the energy between w and its negative context c'. This approach is similar to neg-

ative sampling (Mikolov et al., 2013a,b), which contrasts the dot product between positive context pairs with negative context pairs. The energy function is a measure of similarity between distributions and will be discussed in Section 3.4.

We use a max-margin ranking objective (Joachims, 2002), used for Gaussian embeddings in Vilnis and McCallum (2014), which pushes the similarity of a word and its positive context higher than that of its negative context by a margin m:

$$L_\theta(w, c, c') = \max(0, \\ m - \log E_\theta(w, c) + \log E_\theta(w, c'))$$

This objective can be minimized by mini-batch stochastic gradient descent with respect to the parameters $\theta = \{\vec{\mu}_{w,i}, p_{w,i}, \Sigma_{w,i}\}$ – the mean vectors, covariance matrices, and mixture weights – of our multimodal embedding in Eq. (1).

Word Sampling We use a word sampling scheme similar to the implementation in `word2vec` (Mikolov et al., 2013a,b) to balance the importance of frequent words and rare words. Frequent words such as 'the', 'a', 'to' are not as meaningful as relatively less frequent words such as 'dog', 'love', 'rock', and we are often more interested in learning the semantics of the less frequently observed words. We use subsampling to improve the performance of learning word vectors (Mikolov et al., 2013b). This technique discards word w_i with probability $P(w_i) = 1 - \sqrt{t/f(w_i)}$, where $f(w_i)$ is the frequency of word w_i in the training corpus and t is a frequency threshold.

To generate negative context words, each word type w_i is sampled according to a distribution $P_n(w_i) \propto U(w_i)^{3/4}$ which is a distorted version of the unigram distribution $U(w_i)$ that also serves to diminish the relative importance of frequent words. Both subsampling and the negative distribution choice are proven effective in `word2vec` training (Mikolov et al., 2013b).

3.4 Energy Function

For vector representations of words, a usual choice for similarity measure (energy function) is a dot product between two vectors. Our word representations are distributions instead of point vectors and therefore need a measure that reflects not only the point similarity, but also the uncertainty. We propose to use the *expected likelihood kernel*, which is a generalization of an inner product between vectors to an inner product between distributions (Jebara et al., 2004). That is,

$$E(f, g) = \int f(x)g(x)\, dx = \langle f, g \rangle_{L_2}$$

where $\langle \cdot, \cdot \rangle_{L_2}$ denotes the inner product in Hilbert space L_2. We choose this form of energy since it can be evaluated in a closed form given our choice of probabilistic embedding in Eq. (1).

For Gaussian mixtures f, g representing the words w_f, w_g, $f(x) = \sum_{i=1}^{K} p_i \mathcal{N}(x; \vec{\mu}_{f,i}, \Sigma_{f,i})$ and $g(x) = \sum_{i=1}^{K} q_i \mathcal{N}(x; \vec{\mu}_{g,i}, \Sigma_{g,i})$, $\sum_{i=1}^{K} p_i = 1$, and $\sum_{i=1}^{K} q_i = 1$, we find (see Section A.1) the log energy is

$$\log E_\theta(f, g) = \log \sum_{j=1}^{K} \sum_{i=1}^{K} p_i q_j e^{\xi_{i,j}} \qquad (2)$$

where

$$\begin{aligned} \xi_{i,j} &\equiv \log \mathcal{N}(0; \vec{\mu}_{f,i} - \vec{\mu}_{g,j}, \Sigma_{f,i} + \Sigma_{g,j}) \\ &= -\frac{1}{2} \log \det(\Sigma_{f,i} + \Sigma_{g,j}) - \frac{D}{2} \log(2\pi) \\ &-\frac{1}{2}(\vec{\mu}_{f,i} - \vec{\mu}_{g,j})^\top (\Sigma_{f,i} + \Sigma_{g,j})^{-1} (\vec{\mu}_{f,i} - \vec{\mu}_{g,j}) \end{aligned} \qquad (3)$$

We call the term $\xi_{i,j}$ partial (log) energy. Observe that this term captures the similarity between the i^{th} meaning of word w_f and the j^{th} meaning of word w_g. The total energy in Equation 2 is the sum of possible pairs of partial energies, weighted accordingly by the mixture probabilities p_i and q_j.

The term $-(\vec{\mu}_{f,i} - \vec{\mu}_{g,j})^\top (\Sigma_{f,i} + \Sigma_{g,j})^{-1} (\vec{\mu}_{f,i} - \vec{\mu}_{g,j})$ in $\xi_{i,j}$ explains the difference in mean vectors of semantic pair (w_f, i) and (w_g, j). If the semantic uncertainty (covariance) for both pairs are low, this term has more importance relative to other terms due to the inverse covariance scaling. We observe that the loss function L_θ in Section 3.3 attains a low value when $E_\theta(w, c)$ is relatively high. High values of $E_\theta(w, c)$ can be achieved when the component means across different words $\vec{\mu}_{f,i}$ and $\vec{\mu}_{g,j}$ are close together (e.g., similar point representations). High energy can also be achieved by large values of $\Sigma_{f,i}$ and $\Sigma_{g,j}$, which washes out the importance of the mean vector difference. The term $-\log \det(\Sigma_{f,i} + \Sigma_{g,j})$ serves as a regularizer that prevents the covariances from being pushed too high at the expense of learning a good mean embedding.

At the beginning of training, $\xi_{i,j}$ roughly are on the same scale among all pairs (i, j)'s. During this time, all components learn the signals from the word occurrences equally. As training progresses and the semantic representation of each mixture becomes more clear, there can be one term of $\xi_{i,j}$'s that is predominantly higher than other terms, giving rise to a semantic pair that is most related.

The negative KL divergence is another sensible choice of energy function, providing an asymmetric metric between word distributions. However, unlike the expected likelihood kernel, KL divergence does not have a closed form if the two distributions are Gaussian mixtures.

4 Experiments

We have introduced a model for multi-prototype embeddings, which expressively captures word meanings with whole probability distributions. We show that our combination of energy and objective functions, proposed in Section 3, enables one to learn interpretable multimodal distributions through unsupervised training, for describing words with multiple distinct meanings. By representing multiple distinct meanings, our model also reduces the unnecessarily large variance of a Gaussian embedding model, and has improved results on word entailment tasks.

To learn the parameters of the proposed mixture model, we train on a concatenation of two datasets: `UKWAC` (2.5 billion tokens) and `Wackypedia` (1 billion tokens) (Baroni et al., 2009). We discard words that occur fewer than 100 times in the corpus, which results in a vocabulary size of $314,129$ words. Our word sampling scheme, described at the end of Section 4.3, is similar to that of `word2vec` with one negative context word for each positive context word.

After training, we obtain learned parameters $\{\vec{\mu}_{w,i}, \Sigma_{w,i}, p_i\}_{i=1}^{K}$ for each word w. We treat the mean vector $\vec{\mu}_{w,i}$ as the embedding of the i^{th} mixture component with the covariance matrix $\Sigma_{w,i}$ representing its subtlety and uncertainty. We perform qualitative evaluation to show that our embeddings learn meaningful multi-prototype representations and compare to existing models using a quantitative evaluation on word similarity datasets and word entailment.

We name our model as Word to Gaussian Mixture (`w2gm`) in constrast to Word to Gaussian (`w2g`) (Vilnis and McCallum, 2014). Unless stated otherwise, `w2g` refers to our implementation of `w2gm` model with one mixture component.

4.1 Hyperparameters

Unless stated otherwise, we experiment with $K = 2$ components for the `w2gm` model, but we have results and discussion of $K = 3$ at the end of section 4.3. We primarily consider the spherical case for computational efficiency. We note that for diagonal or spherical covariances, the energy can be computed very efficiently since the matrix inversion would simply require $\mathcal{O}(d)$ computation instead of $\mathcal{O}(d^3)$ for a full matrix. Empirically, we have found diagonal covariance matrices become roughly spherical after training. Indeed, for these relatively high dimensional embeddings, there are sufficient degrees of freedom for the mean vectors to be learned such that the covariance matrices need not be asymmetric. Therefore, we perform all evaluations with spherical covariance models.

Models used for evaluation have dimension $D = 50$ and use context window $\ell = 10$ unless stated otherwise. We provide additional hyperparameters and training details in the supplementary material (A.2).

4.2 Similarity Measures

Since our word embeddings contain multiple vectors and uncertainty parameters per word, we use the following measures that generalizes similarity scores. These measures pick out the component pair with maximum similarity and therefore determine the meanings that are most relevant.

4.2.1 Expected Likelihood Kernel

A natural choice for a similarity score is the expected likelihood kernel, an inner product between distributions, which we discussed in Section 3.4. This metric incorporates the uncertainty from the covariance matrices in addition to the similarity between the mean vectors.

4.2.2 Maximum Cosine Similarity

This metric measures the maximum similarity of mean vectors among all pairs of mixture components between distributions f and g. That is, $d(f, g) = \max_{i,j=1,\ldots,K} \frac{\langle \boldsymbol{\mu}_{f,i}, \boldsymbol{\mu}_{g,j} \rangle}{||\boldsymbol{\mu}_{f,i}|| \cdot ||\boldsymbol{\mu}_{g,j}||}$, which corresponds to matching the meanings of f and g that are the most similar. For a Gaussian embedding, maximum similarity reduces to the usual cosine similarity.

Word	Co.	Nearest Neighbors
rock	0	basalt:1, boulder:1, boulders:0, stalagmites:0, stalactites:0, rocks:1, sand:0, quartzite:1, bedrock:0
rock	1	rock/:1, ska:0, funk:1, pop-rock:1, punk:1, indie-rock:0, band:0, indie:0, pop:1
bank	0	banks:1, mouth:1, river:1, River:0, confluence:0, waterway:1, downstream:1, upstream:0, dammed:0
bank	1	banks:0, banking:1, banker:0, Banks:1, bankas:1, Citibank:1, Interbank:1, Bankers:0, transactions:1
Apple	0	Strawberry:0, Tomato:1, Raspberry:1, Blackberry:1, Apples:0, Pineapple:1, Grape:1, Lemon:0
Apple	1	Macintosh:1, Mac:1, OS:1, Amiga:0, Compaq:0, Atari:1, PC:1, Windows:0, iMac:0
star	0	stars:0, Quaid:0, starlet:0, Dafoe:0, Stallone:0, Geena:0, Niro:0, Zeta-Jones:1, superstar:0
star	1	stars:1, brightest:0, Milky:0, constellation:1, stellar:0, nebula:1, galactic:1, supernova:1, Ophiuchus:1
cell	0	cellular:0, Nextel:0, 2-line:0, Sprint:0, phones.:1, pda:1, handset:0, handsets:1, pushbuttons:0
cell	1	cytoplasm:0, vesicle:0, cytoplasmic:1, macrophages:0, secreted:1, membrane:0, mitotic:0, endocytosis:1
left	0	After:1, back:0, finally:1, eventually:0, broke:0, joined:1, returned:1, after:1, soon:0
left	1	right-hand:0, hand:0, right:0, left-hand:0, lefthand:0, arrow:0, turn:0, righthand:0, Left:0

Word	Nearest Neighbors
rock	band, bands, Rock, indie, Stones, breakbeat, punk, electronica, funk
bank	banks, banking, trader, trading, Bank, capital, Banco, bankers, cash
Apple	Macintosh, Microsoft, Windows, Macs, Lite, Intel, Desktop, WordPerfect, Mac
star	stars, stellar, brightest, Stars, Galaxy, Stardust, eclipsing, stars., Star
cell	cells, DNA, cellular, cytoplasm, membrane, peptide, macrophages, suppressor, vesicles
left	leaving, turned, back, then, After, after, immediately, broke, end

Table 1: Nearest neighbors based on cosine similarity between the mean vectors of Gaussian components for Gaussian mixture embedding (top) (for $K = 2$) and Gaussian embedding (bottom). The notation `w:i` denotes the i^{th} mixture component of the word `w`.

4.2.3 Minimum Euclidean Distance

Cosine similarity is popular for evaluating embeddings. However, our training objective directly involves the Euclidean distance in Eq. (3), as opposed to dot product of vectors such as in `word2vec`. Therefore, we also consider the Euclidean metric: $d(f, g) = \min_{i,j=1,\ldots,K}[||\boldsymbol{\mu}_{f,i} - \boldsymbol{\mu}_{g,j}||]$.

4.3 Qualitative Evaluation

In Table 1, we show examples of polysemous words and their nearest neighbors in the embedding space to demonstrate that our trained embeddings capture multiple word senses. For instance, a word such as 'rock' that could mean either 'stone' or 'rock music' should have each of its meanings represented by a distinct Gaussian component. Our results for a mixture of two Gaussians model confirm this hypothesis, where we observe that the 0^{th} component of 'rock' being related to ('basalt', 'boulders') and the 1^{st} component being related to ('indie', 'funk', 'hip-hop'). Similarly, the word `bank` has its 0^{th} component representing the river bank and the 1^{st} component representing the financial bank.

By contrast, in Table 1 (bottom), see that for Gaussian embeddings with one mixture component, nearest neighbors of polysemous words are predominantly related to a single meaning. For instance, 'rock' mostly has neighbors related to rock music and 'bank' mostly related to the financial bank. The alternative meanings of these polysemous words are not well represented in the embeddings. As a numerical example, the cosine similarity between 'rock' and 'stone' for the Gaussian representation of Vilnis and McCallum (2014) is only 0.029, much lower than the cosine similarity 0.586 between the 0^{th} component of 'rock' and 'stone' in our multimodal representation.

In cases where a word only has a single popular meaning, the mixture components can be fairly close; for instance, one component of 'stone' is close to ('stones', 'stonework', 'slab') and the other to ('carving, 'relic', 'excavated'), which reflects subtle variations in meanings. In general, the mixture can give properties such as heavy tails and more interesting unimodal characterizations of uncertainty than could be described by a single Gaussian.

Embedding Visualization We provide an interactive visualization as part of our code repository: `https://github.com/benathi/word2gm#visualization` that allows real-time queries of words' nearest neighbors (in the `embeddings` tab) for $K = 1, 2, 3$ components. We use a notation similar to that of Table 1, where a token `w:i` represents the component `i` of a word `w`. For instance, if in the $K = 2$ link we search for `bank:0`, we obtain the nearest neigh-

bors such as river:1, confluence:0, waterway:1, which indicates that the 0^{th} component of 'bank' has the meaning 'river bank'. On the other hand, searching for bank:1 yields nearby words such as banking:1, banker:0, ATM:0, indicating that this component is close to the 'financial bank'. We also have a visualization of a unimodal (w2g) for comparison in the $K = 1$ link.

In addition, the embedding link for our Gaussian mixture model with $K = 3$ mixture components can learn three distinct meanings. For instance, each of the three components of 'cell' is close to ('keypad', 'digits'), ('incarcerated', 'inmate') or ('tissue', 'antibody'), indicating that the distribution captures the concept of 'cellphone', 'jail cell', or 'biological cell', respectively. Due to the limited number of words with more than 2 meanings, our model with $K = 3$ does not generally offer substantial performance differences to our model with $K = 2$; hence, we do not further display $K = 3$ results for compactness.

4.4 Word Similarity

We evaluate our embeddings on several standard word similarity datasets, namely, SimLex (Hill et al., 2014), WS or WordSim-353, WS-S (similarity), WS-R (relatedness) (Finkelstein et al., 2002), MEN (Bruni et al., 2014), MC (Miller and Charles, 1991), RG (Rubenstein and Goodenough, 1965), YP (Yang and Powers, 2006), MTurk(-287,-771) (Radinsky et al., 2011; Halawi et al., 2012), and RW (Luong et al., 2013). Each dataset contains a list of word pairs with a human score of how related or similar the two words are.

We calculate the Spearman correlation (Spearman, 1904) between the labels and our scores generated by the embeddings. The Spearman correlation is a rank-based correlation measure that assesses how well the scores describe the true labels.

The correlation results are shown in Table 2 using the scores generated from the expected likelihood kernel, maximum cosine similarity, and maximum Euclidean distance.

We show the results of our Gaussian mixture model and compare the performance with that of word2vec and the original Gaussian embedding by Vilnis and McCallum (2014). We note that our model of a unimodal Gaussian embedding w2g also outperforms the original model, which differs in model hyperparameters and initialization, for most datasets. Our multi-prototype model w2gm also performs better than skip-gram or Gaussian embedding methods on many datasets, namely, WS, WS-R, MEN, MC, RG, YP, MT-287, RW. The maximum cosine similarity yields the best performance on most datasets; however, the minimum Euclidean distance is a better metric for the datasets MC and RW. These results are consistent for both the single-prototype and the multi-prototype models.

We also compare out results on WordSim-353 with the multi-prototype embedding method by Huang et al. (2012) and Neelakantan et al. (2014), shown in Table 3. We observe that our single-prototype model w2g is competitive compared to models by Huang et al. (2012), even without using a corpus with stop words removed. This could be due to the auto-calibration of importance via the covariance learning which decrease the importance of very frequent words such as 'the', 'to', 'a', etc. Moreover, our multi-prototype model substantially outperforms the model of Huang et al. (2012) and the MSSG model of Neelakantan et al. (2014) on the WordSim-353 dataset.

4.5 Word Similarity for Polysemous Words

We use the dataset SCWS introduced by Huang et al. (2012), where word pairs are chosen to have variations in meanings of polysemous and homonymous words.

We compare our method with multiprototype models by Huang (Huang et al., 2012), Tian (Tian et al., 2014), Chen (Chen et al., 2014), and MSSG model by (Neelakantan et al., 2014). We note that Chen model uses an external lexical source WordNet that gives it an extra advantage.

We use many metrics to calculate the scores for the Spearman correlation. MaxSim refers to the maximum cosine similarity. AveSim is the average of cosine similarities with respect to the component probabilities.

In Table 4, the model w2g performs the best among all single-prototype models for either 50 or 200 vector dimensions. Our model w2gm performs competitively compared to other multi-prototype models. In SCWS, the gain in flexibility in moving to a probability density approach appears to dominate over the effects of using a multi-prototype. In most other examples, we see w2gm surpass w2g, where the multi-prototype structure is just as important for good performance as the

Dataset	sg*	w2g*	w2g/mc	w2g/el	w2g/me	w2gm/mc	w2gm/el	w2gm/me
SL	29.39	**32.23**	29.35	25.44	25.43	29.31	26.02	27.59
WS	59.89	65.49	71.53	61.51	64.04	**73.47**	62.85	66.39
WS-S	69.86	76.15	76.70	70.57	72.3	**76.73**	70.08	73.3
WS-R	53.03	58.96	68.34	54.4	55.43	**71.75**	57.98	60.13
MEN	70.27	71.31	72.58	67.81	65.53	**73.55**	68.5	67.7
MC	63.96	70.41	76.48	72.70	**80.66**	79.08	76.75	80.33
RG	70.01	71	73.30	72.29	72.12	**74.51**	71.55	73.52
YP	39.34	41.5	41.96	38.38	36.41	**45.07**	39.18	38.58
MT-287	-	-	64.79	57.5	58.31	**66.60**	57.24	60.61
MT-771	-	-	**60.86**	55.89	54.12	60.82	57.26	56.43
RW	-	-	28.78	32.34	33.16	28.62	31.64	**35.27**

Table 2: Spearman correlation for word similarity datasets. The models `sg`, `w2g`, `w2gm` denote `word2vec` skip-gram, Gaussian embedding, and Gaussian mixture embedding (K=2). The measures `mc`, `el`, `me` denote maximum cosine similarity, expected likelihood kernel, and minimum Euclidean distance. For each of `w2g` and `w2gm`, we underline the similarity metric with the best score. For each dataset, we boldface the score with the best performance across all models. The correlation scores for `sg*`, `w2g*` are taken from Vilnis and McCallum (2014) and correspond to cosine distance.

Model	$\rho \times 100$
Huang	64.2
Huang*	71.3
MSSG 50D	63.2
MSSG 300D	71.2
w2g	70.9
w2gm	**73.5**

Table 3: Spearman's correlation (ρ) on WordSim-353 datasets for our Word to Gaussian Mixture embeddings, as well as the multi-prototype embedding by Huang et al. (2012) and the MSSG model by Neelakantan et al. (2014). `Huang*` is trained using data with all stop words removed. All models have dimension $D = 50$ except for MSSG 300D with $D = 300$ which is still outperformed by our `w2gm` model.

Model	Dimension	$\rho \times 100$
word2vec skip-gram	50	61.7
Huang-S	50	58.6
w2g	50	**64.7**
Chen-S	200	64.2
w2g	200	**66.2**
Huang-M AvgSim	50	62.8
Tian-M MaxSim	50	63.6
w2gm MaxSim	50	62.7
MSSG AvgSim	50	**64.2**
Chen-M AvgSim	200	**66.2**
w2gm MaxSim	200	65.5

Table 4: Spearman's correlation ρ on dataset SCWS. We show the results for single prototype (top) and multi-prototype (bottom) The suffix `-(S,M)` refers to single and multiple prototype models, respectively.

probabilistic representation.

4.6 Reduction in Variance of Polysemous Words

One motivation for our Gaussian mixture embedding is to model word uncertainty more accurately than Gaussian embeddings, which can have overly large variances for polysemous words (in order to assign some mass to all of the distinct meanings). We see that our Gaussian mixture model does indeed reduce the variances of each component for such words. For instance, we observe that the word `rock` in `w2g` has much higher variance per dimension ($e^{-1.8} \approx 1.65$) compared to that of Gaussian components of `rock` in `w2gm` (which has variance of roughly $e^{-2.5} \approx 0.82$). We also see, in the next section, that the Gaussian mixture model has desirable quantitative behavior for word entailment.

4.7 Word Entailment

We evaluate our embeddings on the word entailment dataset from Baroni et al. (2012). The lexical entailment between words is denoted by $w_1 \models w_2$ which means that all instances of w_1 are w_2. The entailment dataset contains positive pairs such as *aircraft* $\models$ *vehicle* and negative pairs such as *aircraft* $\not\models$ *insect*.

We generate entailment scores of word pairs and find the best threshold, measured by Average Precision (AP) or F1 score, which identifies negative versus positive entailment. We use the max-

Model	Score	Best AP	Best F1
w2g (5)	cos	73.1	76.4
w2g (5)	KL	73.7	76.0
w2gm (5)	cos	73.6	76.3
w2gm (5)	KL	75.7	77.9
w2g (10)	cos	73.0	76.1
w2g (10)	KL	74.2	76.1
w2gm (10)	cos	72.9	75.6
w2gm (10)	KL	74.7	76.3

Table 5: Entailment results for models `w2g` and `w2gm` with window size 5 and 10. The metrics used are the maximum cosine similarity, or the maximum negative KL divergence. We calculate the best average precision as well as the best F1 score. In most cases, `w2gm` outperforms `w2g` for describing entailment.

imum cosine similarity and the minimum KL divergence, $d(f, g) = \min_{i,j=1,\ldots,K} KL(f||g)$, for entailment scores. The minimum KL divergence is similar to the maximum cosine similarity, but also incorporates the embedding uncertainty. In addition, KL divergence is an asymmetric measure, which is more suitable for certain tasks such as word entailment where a relationship is unidirectional. For instance, $w_1 \models w_2$ does not imply $w_2 \models w_1$. Indeed, *aircraft* $\models$ *vehicle* does not imply *vehicle* $\models$ *aircraft*, since all aircraft are vehicles but not all vehicles are aircraft. The difference between $KL(w_1||w_2)$ versus $KL(w_2||w_1)$ distinguishes which word distribution encompasses another distribution, as demonstrated in Figure 1.

Table 5 shows the results of our `w2gm` model versus the Gaussian embedding model `w2g`. We observe a trend for both models with window size 5 and 10 that the KL metric yields improvement (both AP and F1) over cosine similarity. In addition, `w2gm` has a better performance compared to `w2g`. The multi-prototype model estimates the meaning uncertainty better since it is no longer constrained to be unimodal, leading to better characterizations of entailment. On the other hand, the Gaussian embedding model suffers from large variance problem for polysemous words, which results in less informative word distribution and inferior entailment scores.

5 Discussion

We introduced a model that represents words with expressive multimodal distributions formed from Gaussian mixtures. To learn the properties of each mixture, we proposed an analytic energy function for combination with a maximum margin objective. The resulting embeddings capture different semantics of polysemous words, uncertainty, and entailment, and also perform favorably on word similarity benchmarks.

Elsewhere, latent probabilistic representations are proving to be exceptionally valuable, able to capture nuances such as face angles with variational autoencoders (Kingma and Welling, 2013) or subtleties in painting strokes with the InfoGAN (Chen et al., 2016). Moreover, classically deterministic deep learning architectures are now being generalized to probabilistic deep models, for full predictive distributions instead of point estimates, and significantly more expressive representations (Wilson et al., 2016b,a; Al-Shedivat et al., 2016; Gan et al., 2016; Fortunato et al., 2017).

Similarly, probabilistic word embeddings can capture a range of subtle meanings, and advance the state of the art in predictive tasks. Multimodal word distributions naturally represent our belief that words do not have single precise meanings: indeed, the shape of a word distribution can express much more semantic information than any point representation.

In the future, multimodal word distributions could open the doors to a new suite of applications in language modelling, where whole word distributions are used as inputs to new probabilistic LSTMs, or in decision functions where uncertainty matters. As part of this effort, we can explore different metrics between distributions, such as KL divergences, which would be a natural choice for order embeddings that model entailment properties. It would also be informative to explore inference over the number of components in mixture models for word distributions. Such an approach could potentially discover an unbounded number of distinct meanings for words, but also distribute the support of each word distribution to express highly nuanced meanings. Alternatively, we could imagine a dependent mixture model where the distributions over words are evolving with time and other covariates. One could also build new types of supervised language models, constructed to more fully leverage the rich information provided by word distributions.

Acknowledgements

We thank NSF IIS-1563887 for support.

References

Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. 2016. Learning scalable deep kernels with recurrent structure. *arXiv preprint arXiv:1610.08936* .

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*. pages 23–32. http://aclweb.org/anthology-new/E/E12/E12-1004.pdf.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation* 43(3):209–226. https://doi.org/10.1007/s10579-009-9081-4.

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

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.* 49(1):1–47.

Xi Chen, Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. pages 2172–2180.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1025–1035. http://aclweb.org/anthology/D/D14/D14-1110.pdf.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. pages 160–167.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Martín Abadi et al. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.* 20(1):116–131.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. 2017. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798* .

Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin. 2016. Scalable bayesian learning of recurrent neural networks for language modeling. *arXiv preprint arXiv:1611.08034* .

Michael Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research* 13:307–361.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*. pages 1406–1414.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR* abs/1408.3456.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*. pages 873–882. http://www.aclweb.org/anthology/P12-1092.

Tony Jebara, Risi Kondor, and Andrew Howard. 2004. Probability product kernels. *Journal of Machine Learning Research* 5:819–844.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. pages 133–142.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR* abs/1312.6114. http://arxiv.org/abs/1312.6114.

Y. LeCun, L. Bottou, G. Orr, and K. Muller. 1998. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 2177–2185.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*. pages 2418–2424. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9314.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*. Sofia, Bulgaria.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukás Burget, and Jan Cernocký. 2011a. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU 2011, Waikoloa, HI, USA, December 11-15, 2011*. pages 196–201. https://doi.org/10.1109/ASRU.2011.6163930.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*. pages 1045–1048.

Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011b. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*. pages 5528–5531. https://doi.org/10.1109/ICASSP.2011.5947611.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*. pages 3111–3119.

George A. Miller and Walter G. Charles. 1991. Contextual Correlates of Semantic Similarity. *Language & Cognitive Processes* 6(1):1–28. https://doi.org/10.1080/01690969108406936.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*. pages 1081–1088.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*.

Eric T. Nalisnick and Sachin Ravi. 2015. Infinite dimensional word embeddings. *CoRR* abs/1511.05392.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1059–1069. http://aclweb.org/anthology/D/D14/D14-1113.pdf.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1532–1543. http://aclweb.org/anthology/D/D14/D14-1162.pdf.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*. WWW '11, pages 337–346.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM* 8(10):627–633.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. pages 880–887. https://doi.org/10.1145/1390156.1390267.

C. Spearman. 1904. The proof and measurement of association between two things. *American Journal of Psychology* 15:88–103.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*. pages 151–160. http://aclweb.org/anthology/C/C14/C14-1016.pdf.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *CoRR* abs/1412.6623.

Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. 2016a. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*. pages 2586–2594.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. 2016b. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. pages 370–378.

Dongqiang Yang and David M. W. Powers. 2006. Verb similarity on the taxonomy of wordnet. In *In the 3rd International WordNet Conference (GWC-06), Jeju Island, Korea*.

A Supplementary Material

A.1 Derivation of Expected Likelihood Kernel

We derive the form of expected likelihood kernel for Gaussian mixtures. Let f, g be Gaussian mixture distributions representing the words w_f, w_g. That is, $f(x) = \sum_{i=1}^{K} p_i \mathcal{N}(x; \mu_{f,i}, \Sigma_{f,i})$ and $g(x) = \sum_{i=1}^{K} q_i \mathcal{N}(x; \mu_{g,i}, \Sigma_{g,i})$, $\sum_{i=1}^{K} p_i = 1$, and $\sum_{i=1}^{K} q_i = 1$.

The expected likelihood kernel is given by

$$
\begin{aligned}
E_\theta(f,g) &= \int \left(\sum_{i=1}^{K} p_i \mathcal{N}(x;\mu_{f,i},\Sigma_{f,i})\right) \cdot \\
&\quad \left(\sum_{j=1}^{K} q_j \mathcal{N}(x;\mu_{g,j},\Sigma_{g,j})\right)\, dx \\
&= \sum_{i=1}^{K}\sum_{j=1}^{K} p_i q_j \int \mathcal{N}(x;\mu_{f,i},\Sigma_{f,i}) \cdot \mathcal{N}(x;\mu_{g,j},\Sigma_{g,j})\, dx \\
&= \sum_{i=1}^{K}\sum_{j=1}^{K} p_i q_j \mathcal{N}(0;\mu_{f,i}-\mu_{g,j},\Sigma_{f,i}+\Sigma_{g,j}) \\
&= \sum_{i=1}^{K}\sum_{j=1}^{K} p_i q_j e^{\xi_{i,j}}
\end{aligned}
$$

where we note that $\int \mathcal{N}(x;\mu_i,\Sigma_i)\mathcal{N}(x;\mu_j,\Sigma_j)\, dx = \mathcal{N}(0,\mu_i-\mu_j,\Sigma_i+\Sigma_j)$ (Vilnis and McCallum, 2014) and $\xi_{i,j}$ is the log partial energy, given by equation 3.

A.2 Implementation

In this section we discuss practical details for training the proposed model.

Reduction to Diagonal Covariance

We use a diagonal Σ, in which case inverting the covariance matrix is trivial and computations are particularly efficient.

Let $\boldsymbol{d}^f, \boldsymbol{d}^g$ denote the diagonal vectors of Σ_f, Σ_g The expression for $\xi_{i,j}$ reduces to

$$
\begin{aligned}
\xi_{i,j} &= -\frac{1}{2}\sum_{r=1}^{D}\log(d_r^p + d_r^q) \\
-\frac{1}{2}\sum &\left[(\boldsymbol{\mu}_{p,i}-\boldsymbol{\mu}_{q,j}) \circ \frac{1}{\boldsymbol{d^p}+\boldsymbol{d^q}} \circ (\boldsymbol{\mu}_{p,i}-\boldsymbol{\mu}_{q,j})\right]
\end{aligned}
$$

where $\circ$ denotes element-wise multiplication. The spherical case which we use in all our experiments is similar since we simply replace a vector $\boldsymbol{d}$ with a single value.

Optimization Constraint and Stability

We optimize $\log \boldsymbol{d}$ since each component of diagonal vector $\boldsymbol{d}$ is constrained to be positive. Similarly, we constrain the probability p_i to be in $[0,1]$ and sum to 1 by optimizing over unconstrained scores $s_i \in (-\infty,\infty)$ and using a softmax function to convert the scores to probability $p_i = \frac{e^{s_i}}{\sum_{j=1}^{K} e^{s_j}}$.

The loss computation can be numerically unstable if elements of the diagonal covariances are very small, due to the term $\log(d_r^f + d_r^g)$ and $\frac{1}{\boldsymbol{d^q}+\boldsymbol{d^p}}$. Therefore, we add a small constant $\epsilon = 10^{-4}$ so that $d_r^f + d_r^g$ and $\boldsymbol{d}^q + \boldsymbol{d}^p$ becomes $d_r^f + d_r^g + \epsilon$ and $\boldsymbol{d^q} + \boldsymbol{d^p} + \epsilon$.

In addition, we observe that $\xi_{i,j}$ can be very small which would result in $e^{\xi_{i,j}} \approx 0$ up to machine precision. In order to stabilize the computation in eq. 2, we compute its equivalent form

$$
\log E(f,g) = \xi_{i',j'} + \log \sum_{j=1}^{K}\sum_{i=1}^{K} p_i q_j e^{\xi_{i,j}-\xi_{i',j'}}
$$

where $\xi_{i',j'} = \max_{i,j} \xi_{i,j}$.

Model Hyperparameters and Training Details

In the loss function L_θ, we use a margin $m = 1$ and a batch size of 128. We initialize the word embeddings with a uniform distribution over $[-\sqrt{\frac{3}{D}}, \sqrt{\frac{3}{D}}]$ so that the expectation of variance is 1 and the mean is zero (LeCun et al., 1998). We initialize each dimension of the diagonal matrix (or a single value for spherical case) with a constant value $v = 0.05$. We also initialize the mixture scores s_i to be 0 so that the initial probabilities are equal among all K components. We use the threshold $t = 10^{-5}$ for negative sampling, which is the recommended value for `word2vec` skip-gram on large datasets.

We also use a separate output embeddings in addition to input embeddings, similar to `word2vec` implementation (Mikolov et al., 2013a,b). That is, each word has two sets of distributions q_I and q_O, each of which is a Gaussian mixture. For a given pair of word and context (w,c), we use the input distribution q_I for w (input word) and the output distribution q_O for context c (output word). We optimize the parameters of both q_I and q_O and use the trained input distributions q_I as our final word representations.

We use mini-batch asynchronous gradient descent with Adagrad (Duchi et al., 2011) which performs adaptive learning rate for each parameter. We also experiment with Adam (Kingma and Ba, 2014) which corrects the bias in adaptive gradient update of Adagrad and is proven very popular for most recent neural network models. However, we found that it is much slower than Adagrad ($\approx$ 10 times). This is because the gradient computation of the model is relatively fast, so a complex gradient update algorithm such as Adam becomes the bottleneck in the optimization. Therefore, we choose to use Adagrad which allows us to better scale to large datasets. We use a linearly decreasing learning rate from 0.05 to 0.00001.

Enhanced LSTM for Natural Language Inference

Qian Chen
University of Science and Technology of China
cq1231@mail.ustc.edu.cn

Xiaodan Zhu
National Research Council Canada
xiaodan.zhu@nrc-cnrc.gc.ca

Zhenhua Ling
University of Science and Technology of China
zhling@ustc.edu.cn

Si Wei
iFLYTEK Research
siwei@iflytek.com

Hui Jiang
York University
hj@cse.yorku.ca

Diana Inkpen
University of Ottawa
diana@site.uottawa.ca

Abstract

Reasoning and inference are central to human and artificial intelligence. Modeling inference in human language is very challenging. With the availability of large annotated data (Bowman et al., 2015), it has recently become feasible to train neural network based inference models, which have shown to be very effective. In this paper, we present a new state-of-the-art result, achieving the accuracy of 88.6% on the Stanford Natural Language Inference Dataset. Unlike the previous top models that use very complicated network architectures, we first demonstrate that carefully designing sequential inference models based on chain LSTMs can outperform all previous models. Based on this, we further show that by explicitly considering recursive architectures in both local inference modeling and inference composition, we achieve additional improvement. Particularly, incorporating syntactic parsing information contributes to our best result—it further improves the performance even when added to the already very strong model.

1 Introduction

Reasoning and inference are central to both human and artificial intelligence. Modeling inference in human language is notoriously challenging but is a basic problem towards true natural language understanding, as pointed out by MacCartney and Manning (2008), *"a necessary (if not sufficient) condition for true natural language understanding is a mastery of open-domain natural language inference."* The previous work has included extensive research on recognizing textual entailment.

Specifically, natural language inference (NLI) is concerned with determining whether a natural-language hypothesis h can be inferred from a premise p, as depicted in the following example from MacCartney (2009), where the hypothesis is regarded to be entailed from the premise.

p: *Several airlines polled saw costs grow more than expected, even after adjusting for inflation.*

h: *Some of the companies in the poll reported cost increases.*

The most recent years have seen advances in modeling natural language inference. An important contribution is the creation of a much larger annotated dataset, the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). The corpus has 570,000 human-written English sentence pairs manually labeled by multiple human subjects. This makes it feasible to train more complex inference models. Neural network models, which often need relatively large annotated data to estimate their parameters, have shown to achieve the state of the art on SNLI (Bowman et al., 2015, 2016; Munkhdalai and Yu, 2016b; Parikh et al., 2016; Sha et al., 2016; Paria et al., 2016).

While some previous top-performing models use rather complicated network architectures to achieve the state-of-the-art results (Munkhdalai and Yu, 2016b), we demonstrate in this paper that enhancing sequential inference models based on chain

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1657–1668
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1152

models can outperform all previous results, suggesting that the potentials of such sequential inference approaches have not been fully exploited yet. More specifically, we show that our sequential inference model achieves an accuracy of 88.0% on the SNLI benchmark.

Exploring syntax for NLI is very attractive to us. In many problems, syntax and semantics interact closely, including in semantic composition (Partee, 1995), among others. Complicated tasks such as natural language inference could well involve both, which has been discussed in the context of recognizing textual entailment (RTE) (Mehdad et al., 2010; Ferrone and Zanzotto, 2014). In this paper, we are interested in exploring this within the neural network frameworks, with the presence of relatively large training data. We show that by explicitly encoding parsing information with recursive networks in both local inference modeling and inference composition and by incorporating it into our framework, we achieve additional improvement, increasing the performance to a new state of the art with an 88.6% accuracy.

2 Related Work

Early work on natural language inference has been performed on rather small datasets with more conventional methods (refer to MacCartney (2009) for a good literature survey), which includes a large bulk of work on recognizing textual entailment, such as (Dagan et al., 2005; Iftene and Balahur-Dobrescu, 2007), among others. More recently, Bowman et al. (2015) made available the SNLI dataset with 570,000 human annotated sentence pairs. They also experimented with simple classification models as well as simple neural networks that encode the premise and hypothesis independently. Rocktäschel et al. (2015) proposed neural attention-based models for NLI, which captured the attention information. In general, attention based models have been shown to be effective in a wide range of tasks, including machine translation (Bahdanau et al., 2014), speech recognition (Chorowski et al., 2015; Chan et al., 2016), image caption (Xu et al., 2015), and text summarization (Rush et al., 2015; Chen et al., 2016), among others. For NLI, the idea allows neural models to pay attention to specific areas of the sentences.

A variety of more advanced networks have been developed since then (Bowman et al., 2016; Vendrov et al., 2015; Mou et al., 2016; Liu et al., 2016; Munkhdalai and Yu, 2016a; Rocktäschel et al., 2015; Wang and Jiang, 2016; Cheng et al., 2016; Parikh et al., 2016; Munkhdalai and Yu, 2016b; Sha et al., 2016; Paria et al., 2016). Among them, more relevant to ours are the approaches proposed by Parikh et al. (2016) and Munkhdalai and Yu (2016b), which are among the best performing models.

Parikh et al. (2016) propose a relatively simple but very effective decomposable model. The model decomposes the NLI problem into subproblems that can be solved separately. On the other hand, Munkhdalai and Yu (2016b) propose much more complicated networks that consider sequential LSTM-based encoding, recursive networks, and complicated combinations of attention models, which provide about 0.5% gain over the results reported by Parikh et al. (2016).

It is, however, not very clear if the potential of the sequential inference networks has been well exploited for NLI. In this paper, we first revisit this problem and show that enhancing sequential inference models based on chain networks can actually outperform all previous results. We further show that explicitly considering recursive architectures to encode syntactic parsing information for NLI could further improve the performance.

3 Hybrid Neural Inference Models

We present here our natural language inference networks which are composed of the following major components: input encoding, local inference modeling, and inference composition. Figure 1 shows a high-level view of the architecture. Vertically, the figure depicts the three major components, and horizontally, the left side of the figure represents our sequential NLI model named ESIM, and the right side represents networks that incorporate syntactic parsing information in tree LSTMs.

In our notation, we have two sentences $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_{\ell_a})$ and $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_{\ell_b})$, where $\mathbf{a}$ is a premise and $\mathbf{b}$ a hypothesis. The $\mathbf{a}_i$ or $\mathbf{b}_j \in \mathbb{R}^l$ is an embedding of l-dimensional vector, which can be initialized with some pre-trained word embeddings and organized with parse trees. The goal is to predict a label y that indicates the logic relationship between $\mathbf{a}$ and $\mathbf{b}$.

3.1 Input Encoding

We employ bidirectional LSTM (BiLSTM) as one of our basic building blocks for NLI. We first use it

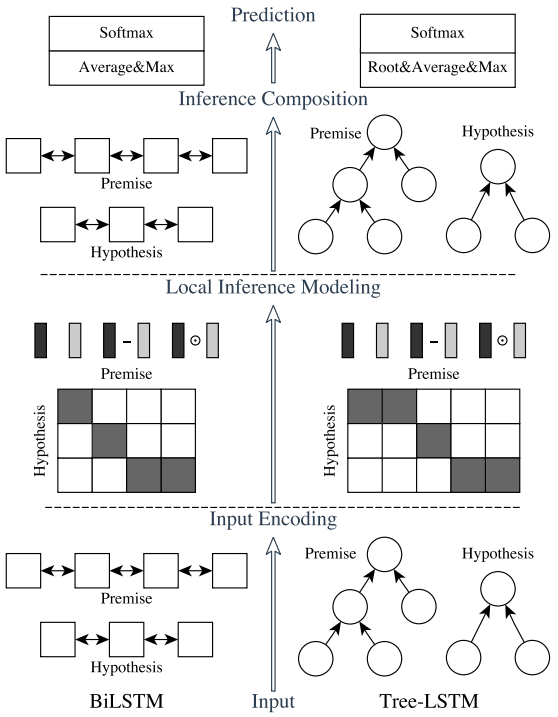


Figure 1: A high-level view of our hybrid neural inference networks.

to encode the input premise and hypothesis (Equation (1) and (2)). Here BiLSTM learns to represent a word (e.g., $\mathbf{a}_i$) and its context. Later we will also use BiLSTM to perform *inference composition* to construct the final prediction, where BiLSTM encodes local inference information and its interaction. To bookkeep the notations for later use, we write as $\bar{\mathbf{a}}_i$ the hidden (output) state generated by the BiLSTM at time i over the input sequence $\mathbf{a}$. The same is applied to $\bar{\mathbf{b}}_j$:

$$\bar{\mathbf{a}}_i = \text{BiLSTM}(\mathbf{a}, i), \forall i \in [1, \ldots, \ell_a], \quad (1)$$

$$\bar{\mathbf{b}}_j = \text{BiLSTM}(\mathbf{b}, j), \forall j \in [1, \ldots, \ell_b]. \quad (2)$$

Due to the space limit, we will skip the description of the basic chain LSTM and readers can refer to Hochreiter and Schmidhuber (1997) for details. Briefly, when modeling a sequence, an LSTM employs a set of soft gates together with a memory cell to control message flows, resulting in an effective modeling of tracking long-distance information/dependencies in a sequence.

A bidirectional LSTM runs a forward and backward LSTM on a sequence starting from the left and the right end, respectively. The hidden states generated by these two LSTMs at each time step are concatenated to represent that time step and its context. Note that we used LSTM memory blocks in our models. We examined other recurrent memory blocks such as GRUs (Gated Recurrent Units) (Cho et al., 2014) and they are inferior to LSTMs on the heldout set for our NLI task.

As discussed above, it is intriguing to explore the effectiveness of syntax for natural language inference; for example, whether it is useful even when incorporated into the best-performing models. To this end, we will also encode syntactic parse trees of a premise and hypothesis through tree-LSTM (Zhu et al., 2015; Tai et al., 2015; Le and Zuidema, 2015), which extends the chain LSTM to a recursive network (Socher et al., 2011).

Specifically, given the parse of a premise or hypothesis, a tree node is deployed with a tree-LSTM memory block depicted as in Figure 2 and computed with Equations (3–10). In short, at each node, an input vector $\mathbf{x}_t$ and the hidden vectors of its two children (the left child $\mathbf{h}^L_{t-1}$ and the right $\mathbf{h}^R_{t-1}$) are taken in as the input to calculate the current node's hidden vector $\mathbf{h}_t$.

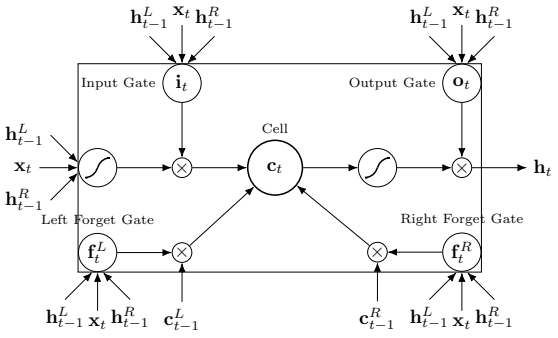


Figure 2: A tree-LSTM memory block.

We describe the updating of a node at a high level with Equation (3) to facilitate references later in the paper, and the detailed computation is described in (4–10). Specifically, the input of a node is used to configure four gates: the input gate $\mathbf{i}_t$, output gate $\mathbf{o}_t$, and the two forget gates $\mathbf{f}^L_t$ and $\mathbf{f}^R_t$. The memory cell $\mathbf{c}_t$ considers each child's cell vector, $\mathbf{c}^L_{t-1}$ and $\mathbf{c}^R_{t-1}$, which are gated by the left forget

gate $\mathbf{f}_t^L$ and right forget gate $\mathbf{f}_t^R$, respectively.

$$\mathbf{h}_t = \text{TrLSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}^L, \mathbf{h}_{t-1}^R), \quad (3)$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (4)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o^L \mathbf{h}_{t-1}^L + \mathbf{U}_o^R \mathbf{h}_{t-1}^R), \quad (5)$$
$$\mathbf{c}_t = \mathbf{f}_t^L \odot \mathbf{c}_{t-1}^L + \mathbf{f}_t^R \odot \mathbf{c}_{t-1}^R + \mathbf{i}_t \odot \mathbf{u}_t, \quad (6)$$
$$\mathbf{f}_t^L = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f^{LL} \mathbf{h}_{t-1}^L + \mathbf{U}_f^{LR} \mathbf{h}_{t-1}^R), \quad (7)$$
$$\mathbf{f}_t^R = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f^{RL} \mathbf{h}_{t-1}^L + \mathbf{U}_f^{RR} \mathbf{h}_{t-1}^R), \quad (8)$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i^L \mathbf{h}_{t-1}^L + \mathbf{U}_i^R \mathbf{h}_{t-1}^R), \quad (9)$$
$$\mathbf{u}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c^L \mathbf{h}_{t-1}^L + \mathbf{U}_c^R \mathbf{h}_{t-1}^R), \quad (10)$$

where σ is the sigmoid function, $\odot$ is the element-wise multiplication of two vectors, and all $\mathbf{W} \in \mathbb{R}^{d \times l}$, $\mathbf{U} \in \mathbb{R}^{d \times d}$ are weight matrices to be learned.

In the current *input encoding* layer, $\mathbf{x}_t$ is used to encode a word embedding for a leaf node. Since a non-leaf node does not correspond to a specific word, we use a special vector $\mathbf{x}'_t$ as its input, which is like an unknown word. However, in the *inference composition* layer that we discuss later, the goal of using tree-LSTM is very different; the input $\mathbf{x}_t$ will be very different as well—it will encode local inference information and will have values at all tree nodes.

3.2 Local Inference Modeling

Modeling local subsentential inference between a premise and hypothesis is the basic component for determining the overall inference between these two statements. To closely examine local inference, we explore both the sequential and syntactic tree models that have been discussed above. The former helps collect local inference for words and their context, and the tree LSTM helps collect local information between (linguistic) phrases and clauses.

Locality of inference Modeling local inference needs to employ some forms of hard or soft alignment to associate the relevant subcomponents between a premise and a hypothesis. This includes early methods motivated from the alignment in conventional automatic machine translation (MacCartney, 2009). In neural network models, this is often achieved with soft attention.

Parikh et al. (2016) decomposed this process: the word sequence of the premise (or hypothesis) is regarded as a bag-of-word embedding vector and inter-sentence "alignment" (or attention) is computed individually to softly align each word to the content of hypothesis (or premise, respectively). While their basic framework is very effective, achieving one of the previous best results, using a pre-trained word embedding by itself does not automatically consider the context around a word in NLI. Parikh et al. (2016) did take into account the word order and context information through an optional distance-sensitive intra-sentence attention.

In this paper, we argue for leveraging attention over the bidirectional sequential encoding of the input, as discussed above. We will show that this plays an important role in achieving our best results, and the intra-sentence attention used by Parikh et al. (2016) actually does not further improve over our model, while the overall framework they proposed is very effective.

Our soft alignment layer computes the attention weights as the similarity of a hidden state tuple $<\bar{\mathbf{a}}_i, \bar{\mathbf{b}}_j>$ between a premise and a hypothesis with Equation (11). We did study more complicated relationships between $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{b}}_j$ with multilayer perceptrons, but observed no further improvement on the heldout data.

$$e_{ij} = \bar{\mathbf{a}}_i^T \bar{\mathbf{b}}_j. \quad (11)$$

In the formula, $\bar{\mathbf{a}}_i$ and $\bar{\mathbf{b}}_j$ are computed earlier in Equations (1) and (2), or with Equation (3) when tree-LSTM is used. Again, as discussed above, we will use bidirectional LSTM and tree-LSTM to encode the premise and hypothesis, respectively. In our sequential inference model, unlike in Parikh et al. (2016) which proposed to use a function $F(\bar{\mathbf{a}}_i)$, i.e., a feedforward neural network, to map the original word representation for calculating e_{ij}, we instead advocate to use BiLSTM, which encodes the information in premise and hypothesis very well and achieves better performance shown in the experiment section. We tried to apply the $F(.)$ function on our hidden states before computing e_{ij} and it did not further help our models.

Local inference collected over sequences Local inference is determined by the attention weight e_{ij} computed above, which is used to obtain the local relevance between a premise and hypothesis. For the hidden state of a word in a premise, i.e., $\bar{\mathbf{a}}_i$ (already encoding the word itself and its context), the relevant semantics in the hypothesis is identified and composed using e_{ij}, more specifically

with Equation (12).

$$\tilde{\mathbf{a}}_i = \sum_{j=1}^{\ell_b} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_b} \exp(e_{ik})} \bar{\mathbf{b}}_j, \forall i \in [1, \ldots, \ell_a], \quad (12)$$

$$\tilde{\mathbf{b}}_j = \sum_{i=1}^{\ell_a} \frac{\exp(e_{ij})}{\sum_{k=1}^{\ell_a} \exp(e_{kj})} \bar{\mathbf{a}}_i, \forall j \in [1, \ldots, \ell_b], \quad (13)$$

where $\tilde{\mathbf{a}}_i$ is a weighted summation of $\{\bar{\mathbf{b}}_j\}_{j=1}^{\ell_b}$. Intuitively, the content in $\{\bar{\mathbf{b}}_j\}_{j=1}^{\ell_b}$ that is relevant to $\bar{\mathbf{a}}_i$ will be selected and represented as $\tilde{\mathbf{a}}_i$. The same is performed for each word in the hypothesis with Equation (13).

Local inference collected over parse trees We use tree models to help collect local inference information over linguistic phrases and clauses in this layer. The tree structures of the premise and hypothesis are produced by a constituency parser.

Once the hidden states of a tree are all computed with Equation (3), we treat all tree nodes equally as we do not have further heuristics to discriminate them, but leave the attention weights to figure out their relationship. So, we use Equation (11) to compute the attention weights for all node pairs between a premise and hypothesis. This connects all words, constituent phrases, and clauses between the premise and hypothesis. We then collect the information between all the pairs with Equations (12) and (13) and feed them into the next layer.

Enhancement of local inference information In our models, we further enhance the local inference information collected. We compute the difference and the element-wise product for the tuple $<\bar{\mathbf{a}}, \tilde{\mathbf{a}}>$ as well as for $<\bar{\mathbf{b}}, \tilde{\mathbf{b}}>$. We expect that such operations could help sharpen local inference information between elements in the tuples and capture inference relationships such as contradiction. The difference and element-wise product are then concatenated with the original vectors, $\bar{\mathbf{a}}$ and $\tilde{\mathbf{a}}$, or $\bar{\mathbf{b}}$ and $\tilde{\mathbf{b}}$, respectively (Mou et al., 2016; Zhang et al., 2017). The enhancement is performed for both the sequential and the tree models.

$$\mathbf{m}_a = [\bar{\mathbf{a}}; \tilde{\mathbf{a}}; \bar{\mathbf{a}} - \tilde{\mathbf{a}}; \bar{\mathbf{a}} \odot \tilde{\mathbf{a}}], \quad (14)$$

$$\mathbf{m}_b = [\bar{\mathbf{b}}; \tilde{\mathbf{b}}; \bar{\mathbf{b}} - \tilde{\mathbf{b}}; \bar{\mathbf{b}} \odot \tilde{\mathbf{b}}]. \quad (15)$$

This process could be regarded as a special case of modeling some high-order interaction between the tuple elements. Along this direction, we have also further modeled the interaction by feeding the tuples into feedforward neural networks and added the top layer hidden states to the above concatenation. We found that it does not further help the inference accuracy on the heldout dataset.

3.3 Inference Composition

To determine the overall inference relationship between a premise and hypothesis, we explore a composition layer to compose the enhanced local inference information $\mathbf{m}_a$ and $\mathbf{m}_b$. We perform the composition sequentially or in its parse context using BiLSTM and tree-LSTM, respectively.

The composition layer In our sequential inference model, we keep using BiLSTM to compose local inference information sequentially. The formulas for BiLSTM are similar to those in Equations (1) and (2) in their forms so we skip the details, but the aim is very different here—they are used to capture local inference information $\mathbf{m}_a$ and $\mathbf{m}_b$ and their context here for inference composition.

In the tree composition, the high-level formulas of how a tree node is updated to compose local inference is as follows:

$$\mathbf{v}_{a,t} = \text{TrLSTM}(F(\mathbf{m}_{a,t}), \mathbf{h}_{t-1}^L, \mathbf{h}_{t-1}^R), \quad (16)$$

$$\mathbf{v}_{b,t} = \text{TrLSTM}(F(\mathbf{m}_{b,t}), \mathbf{h}_{t-1}^L, \mathbf{h}_{t-1}^R). \quad (17)$$

We propose to control model complexity in this layer, since the concatenation we described above to compute $\mathbf{m}_a$ and $\mathbf{m}_b$ can significantly increase the overall parameter size to potentially overfit the models. We propose to use a mapping F as in Equation (16) and (17). More specifically, we use a 1-layer feedforward neural network with the ReLU activation. This function is also applied to BiLSTM in our sequential inference composition.

Pooling Our inference model converts the resulting vectors obtained above to a fixed-length vector with pooling and feeds it to the final classifier to determine the overall inference relationship.

We consider that summation (Parikh et al., 2016) could be sensitive to the sequence length and hence less robust. We instead suggest the following strategy: compute both average and max pooling, and concatenate all these vectors to form the final fixed length vector $\mathbf{v}$. Our experiments show that this leads to significantly better results than summation. The final fixed length vector $\mathbf{v}$ is calculated

as follows:

$$\mathbf{v}_{a,\text{ave}} = \sum_{i=1}^{\ell_a} \frac{\mathbf{v}_{a,i}}{\ell_a}, \quad \mathbf{v}_{a,\max} = \max_{i=1}^{\ell_a} \mathbf{v}_{a,i}, \tag{18}$$

$$\mathbf{v}_{b,\text{ave}} = \sum_{j=1}^{\ell_b} \frac{\mathbf{v}_{b,j}}{\ell_b}, \quad \mathbf{v}_{b,\max} = \max_{j=1}^{\ell_b} \mathbf{v}_{b,j}, \tag{19}$$

$$\mathbf{v} = [\mathbf{v}_{a,\text{ave}}; \mathbf{v}_{a,\max}; \mathbf{v}_{b,\text{ave}}; \mathbf{v}_{b,\max}]. \tag{20}$$

Note that for tree composition, Equation (20) is slightly different from that in sequential composition. Our tree composition will concatenate also the hidden states computed for the roots with Equations (16) and (17), which are not shown here.

We then put $\mathbf{v}$ into a final multilayer perceptron (MLP) classifier. The MLP has a hidden layer with *tanh* activation and *softmax* output layer in our experiments. The entire model (all three components described above) is trained end-to-end. For training, we use multi-class cross-entropy loss.

Overall inference models Our model can be based only on the sequential networks by removing all tree components and we call it Enhanced Sequential Inference Model (**ESIM**) (see the left part of Figure 1). We will show that ESIM outperforms all previous results. We will also encode parse information with tree LSTMs in multiple layers as described (see the right side of Figure 1). We train this model and incorporate it into ESIM by averaging the predicted probabilities to get the final label for a premise-hypothesis pair. We will show that parsing information complements very well with ESIM and further improves the performance, and we call the final model Hybrid Inference Model (**HIM**).

4 Experimental Setup

Data The Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) focuses on three basic relationships between a premise and a potential hypothesis: the premise entails the hypothesis (*entailment*), they contradict each other (*contradiction*), or they are not related (*neutral*). The original SNLI corpus contains also "*the other*" category, which includes the sentence pairs lacking consensus among multiple human annotators. As in the related work, we remove this category. We used the same split as in Bowman et al. (2015) and other previous work.

The parse trees used in this paper are produced by the Stanford PCFG Parser 3.5.3 (Klein and Manning, 2003) and they are delivered as part of the SNLI corpus. We use classification accuracy as the evaluation metric, as in related work.

Training We use the development set to select models for testing. To help replicate our results, we publish our code[1]. Below, we list our training details. We use the Adam method (Kingma and Ba, 2014) for optimization. The first momentum is set to be 0.9 and the second 0.999. The initial learning rate is 0.0004 and the batch size is 32. All hidden states of LSTMs, tree-LSTMs, and word embeddings have 300 dimensions.

We use dropout with a rate of 0.5, which is applied to all feedforward connections. We use pre-trained *300-D Glove 840B* vectors (Pennington et al., 2014) to initialize our word embeddings. Out-of-vocabulary (OOV) words are initialized randomly with Gaussian samples. All vectors including word embedding are updated during training.

5 Results

Overall performance Table 1 shows the results of different models. The first row is a baseline classifier presented by Bowman et al. (2015) that considers handcrafted features such as BLEU score of the hypothesis with respect to the premise, the overlapped words, and the length difference between them, etc.

The next group of models (2)-(7) are based on sentence encoding. The model of Bowman et al. (2016) encodes the premise and hypothesis with two different LSTMs. The model in Vendrov et al. (2015) uses unsupervised "skip-thoughts" pre-training in GRU encoders. The approach proposed by Mou et al. (2016) considers tree-based CNN to capture sentence-level semantics, while the model of Bowman et al. (2016) introduces a stack-augmented parser-interpreter neural network (SPINN) which combines parsing and interpretation within a single tree-sequence hybrid model. The work by Liu et al. (2016) uses BiLSTM to generate sentence representations, and then replaces average pooling with intra-attention. The approach proposed by Munkhdalai and Yu (2016a) presents a memory augmented neural network, neural semantic encoders (NSE), to encode sentences.

The next group of methods in the table, models

[1] https://github.com/lukecq1231/nli

Model	#Para.	Train	Test
(1) Handcrafted features (Bowman et al., 2015)	-	99.7	78.2
(2) 300D LSTM encoders (Bowman et al., 2016)	3.0M	83.9	80.6
(3) 1024D pretrained GRU encoders (Vendrov et al., 2015)	15M	98.8	81.4
(4) 300D tree-based CNN encoders (Mou et al., 2016)	3.5M	83.3	82.1
(5) 300D SPINN-PI encoders (Bowman et al., 2016)	3.7M	89.2	83.2
(6) 600D BiLSTM intra-attention encoders (Liu et al., 2016)	2.8M	84.5	84.2
(7) 300D NSE encoders (Munkhdalai and Yu, 2016a)	3.0M	86.2	84.6
(8) 100D LSTM with attention (Rocktäschel et al., 2015)	250K	85.3	83.5
(9) 300D mLSTM (Wang and Jiang, 2016)	1.9M	92.0	86.1
(10) 450D LSTMN with deep attention fusion (Cheng et al., 2016)	3.4M	88.5	86.3
(11) 200D decomposable attention model (Parikh et al., 2016)	380K	89.5	86.3
(12) Intra-sentence attention + (11) (Parikh et al., 2016)	580K	90.5	86.8
(13) 300D NTI-SLSTM-LSTM (Munkhdalai and Yu, 2016b)	3.2M	88.5	87.3
(14) 300D re-read LSTM (Sha et al., 2016)	2.0M	90.7	87.5
(15) 300D btree-LSTM encoders (Paria et al., 2016)	2.0M	88.6	87.6
(16) 600D ESIM	4.3M	92.6	88.0
(17) HIM (600D ESIM + 300D Syntactic tree-LSTM)	7.7M	93.5	**88.6**

Table 1: Accuracies of the models on SNLI. Our final model achieves the accuracy of 88.6%, the best result observed on SNLI, while our enhanced sequential encoding model attains an accuracy of 88.0%, which also outperform the previous models.

(8)-(15), are inter-sentence attention-based model. The model marked with Rocktäschel et al. (2015) is LSTMs enforcing the so called word-by-word attention. The model of Wang and Jiang (2016) extends this idea to explicitly enforce word-by-word matching between the hypothesis and the premise. Long short-term memory-networks (LSTMN) with deep attention fusion (Cheng et al., 2016) link the current word to previous words stored in memory. Parikh et al. (2016) proposed a decomposable attention model without relying on any word-order information. In general, adding intra-sentence attention yields further improvement, which is not very surprising as it could help align the relevant text spans between premise and hypothesis. The model of Munkhdalai and Yu (2016b) extends the framework of Wang and Jiang (2016) to a full n-ary tree model and achieves further improvement. Sha et al. (2016) proposes a special LSTM variant which considers the attention vector of another sentence as an inner state of LSTM. Paria et al. (2016) use a neural architecture with a complete binary tree-LSTM encoders without syntactic information.

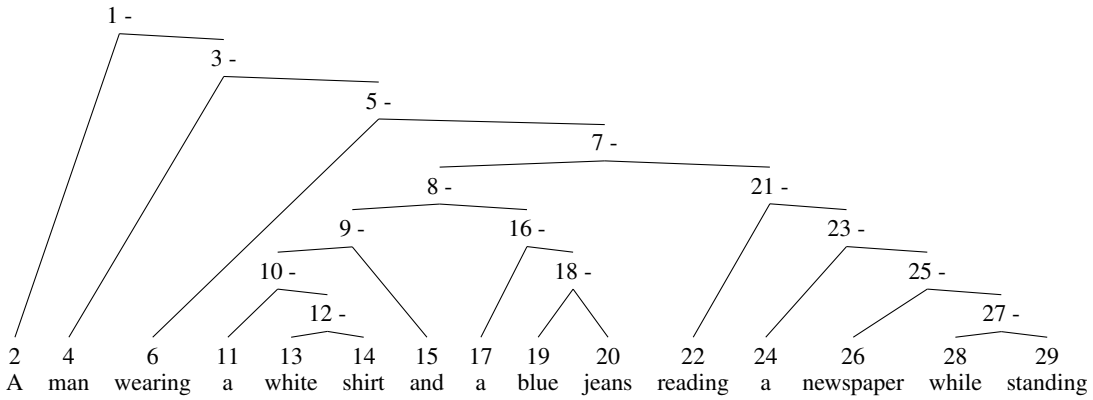
The table shows that our ESIM model achieves an accuracy of 88.0%, which has already outperformed all the previous models, including those using much more complicated network architectures (Munkhdalai and Yu, 2016b).

We ensemble our ESIM model with syntactic tree-LSTMs (Zhu et al., 2015) based on syntactic parse trees and achieve significant improvement over our best sequential encoding model ESIM, attaining an accuracy of 88.6%. This shows that syntactic tree-LSTMs complement well with ESIM.

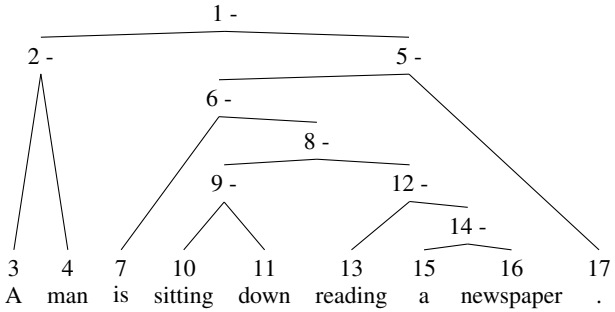
Model	Train	Test
(17) HIM (ESIM + syn.tree)	93.5	88.6
(18) ESIM + tree	91.9	88.2
(16) ESIM	92.6	88.0
(19) ESIM - ave./max	92.9	87.1
(20) ESIM - diff./prod.	91.5	87.0
(21) ESIM - inference BiLSTM	91.3	87.3
(22) ESIM - encoding BiLSTM	88.7	86.3
(23) ESIM - P-based attention	91.6	87.2
(24) ESIM - H-based attention	91.4	86.5
(25) syn.tree	92.9	87.8

Table 2: Ablation performance of the models.

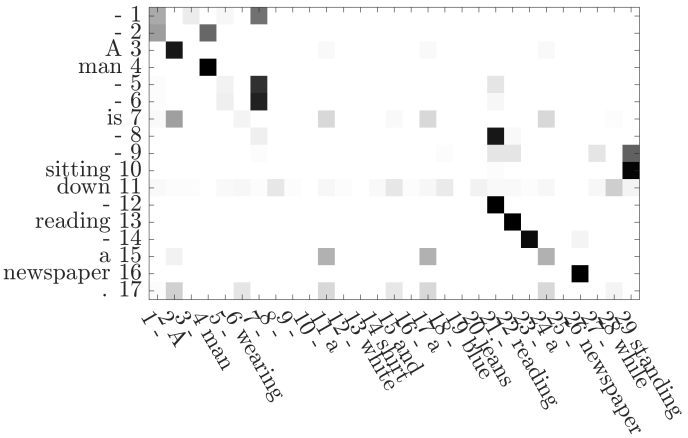
Ablation analysis We further analyze the major components that are of importance to help us achieve good performance. From the best model, we first replace the syntactic tree-LSTM with the full tree-LSTM without encoding syntactic parse information. More specifically, two adjacent words in a sentence are merged to form a parent node, and



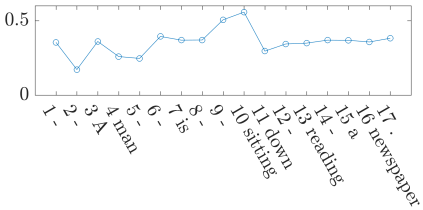
(a) Binarized constituency tree of premise



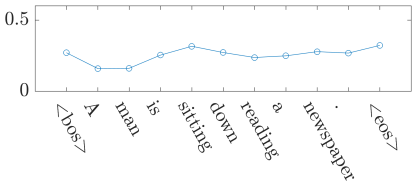
(b) Binarized constituency tree of hypothesis



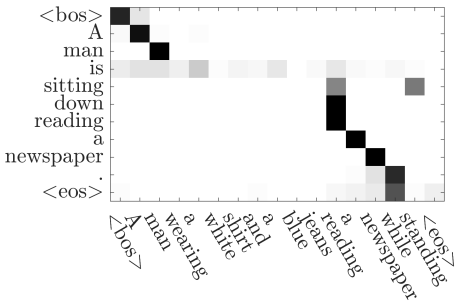
(c) Normalized attention weights of tree-LSTM



(d) Input gate of tree-LSTM in *inference composition* (l^2-norm)



(e) Input gate of BiLSTM in *inference composition* (l^2-norm)



(f) Normalized attention weights of BiLSTM

Figure 3: An example for analysis. Subfigures (a) and (b) are the constituency parse trees of the premise and hypothesis, respectively. "-" means a non-leaf or a null node. Subfigures (c) and (f) are attention visualization of the tree model and ESIM, respectively. The darker the color, the greater the value. The premise is on the x-axis and the hypothesis is on y-axis. Subfigures (d) and (e) are input gates' l^2-norm of tree-LSTM and BiLSTM in *inference composition*, respectively.

this process continues and results in a full binary tree, where padding nodes are inserted when there are no enough leaves to form a full tree. Each tree node is implemented with a tree-LSTM block (Zhu et al., 2015) same as in model (17). Table 2 shows that with this replacement, the performance drops to 88.2%.

Furthermore, we note the importance of the layer performing the enhancement for local inference information in Section 3.2 and the pooling layer in inference composition in Section 3.3. Table 2 suggests that the NLI task seems very sensitive to the

layers. If we remove the pooling layer in inference composition and replace it with summation as in Parikh et al. (2016), the accuracy drops to 87.1%. If we remove the difference and element-wise product from the local inference enhancement layer, the accuracy drops to 87.0%. To provide some detailed comparison with Parikh et al. (2016), replacing bidirectional LSTMs in *inference composition* and also *input encoding* with feedforward neural network reduces the accuracy to 87.3% and 86.3% respectively.

The difference between ESIM and each of the other models listed in Table 2 is statistically significant under the one-tailed paired t-test at the 99% significance level. The difference between model (17) and (18) is also significant at the same level. Note that we cannot perform significance test between our models with the other models listed in Table 1 since we do not have the output of the other models.

If we remove the *premise-based* attention from ESIM (model 23), the accuracy drops to 87.2% on the test set. The *premise-based* attention means when the system reads a word in a premise, it uses soft attention to consider all relevant words in hypothesis. Removing the *hypothesis-based* attention (model 24) decrease the accuracy to 86.5%, where *hypothesis-based* attention is the attention performed on the other direction for the sentence pairs. The results show that removing *hypothesis-based* attention affects the performance of our model more, but removing the attention from the other direction impairs the performance too.

The stand-alone syntactic tree-LSTM model achieves an accuracy of 87.8%, which is comparable to that of ESIM. We also computed the oracle score of merging syntactic tree-LSTM and ESIM, which picks the right answer if either is right. Such an oracle/upper-bound accuracy on test set is 91.7%, which suggests how much tree-LSTM and ESIM could ideally complement each other. As far as the speed is concerned, training tree-LSTM takes about 40 hours on Nvidia-Tesla K40M and ESIM takes about 6 hours, which is easily extended to larger scale of data.

Further analysis We showed that encoding syntactic parsing information helps recognize natural language inference—it additionally improves the strong system. Figure 3 shows an example where tree-LSTM makes a different and correct decision. In subfigure (d), the larger values at the input gates on nodes 9 and 10 indicate that those nodes are important in making the final decision. We observe that in subfigure (c), nodes 9 and 10 are aligned to node 29 in the premise. Such information helps the system decide that this pair is a contradiction. Accordingly, in subfigure (e) of sequential BiLSTM, the words *sitting* and *down* do not play an important role for making the final decision. Subfigure (f) shows that *sitting* is equally aligned with *reading* and *standing* and the alignment for word *down* is not that useful.

6 Conclusions and Future Work

We propose neural network models for natural language inference, which achieve the best results reported on the SNLI benchmark. The results are first achieved through our enhanced sequential inference model, which outperformed the previous models, including those employing more complicated network architectures, suggesting that the potential of sequential inference models have not been fully exploited yet. Based on this, we further show that by explicitly considering recursive architectures in both local inference modeling and inference composition, we achieve additional improvement. Particularly, incorporating syntactic parsing information contributes to our best result: it further improves the performance even when added to the already very strong model.

Future work interesting to us includes exploring the usefulness of external resources such as WordNet and contrasting-meaning embedding (Chen et al., 2015) to help increase the coverage of word-level inference relations. Modeling negation more closely within neural network frameworks (Socher et al., 2013; Zhu et al., 2014) may help contradiction detection.

Acknowledgments

The first and the third author of this paper were supported in part by the Science and Technology Development of Anhui Province, China (Grants No. 2014z02006), the Fundamental Research Funds for the Central Universities (Grant No. WK2350000001) and the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB02070006).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Samuel Bowman, Gabor Angeli, Christopher Potts, and D. Christopher Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 632–642. https://doi.org/10.18653/v1/D15-1075.

Samuel Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, D. Christopher Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1466–1477. https://doi.org/10.18653/v1/P16-1139.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*. IEEE, pages 4960–4964. https://doi.org/10.1109/ICASSP.2016.7472621.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling document. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. IJCAI/AAAI Press, pages 2754–2760. http://www.ijcai.org/Abstract/16/391.

Zhigang Chen, Wei Lin, Qian Chen, Xiaoping Chen, Si Wei, Hui Jiang, and Xiaodan Zhu. 2015. Revisiting word embedding for contrasting meaning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 106–115. https://doi.org/10.3115/v1/P15-1011.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 551–561. http://aclweb.org/anthology/D16-1053.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi, editors, *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*. Association for Computational Linguistics, pages 103–111. http://aclweb.org/anthology/W/W14/W14-4012.pdf.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 577–585. http://papers.nips.cc/paper/5847-attention-based-models-for-speech-recognition.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*. pages 177–190.

Lorenzo Ferrone and Massimo Fabio Zanzotto. 2014. Towards syntax-aware compositional distributional semantic models. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, pages 721–730. http://aclweb.org/anthology/C14-1068.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Adrian Iftene and Alexandra Balahur-Dobrescu. 2007. *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Association for Computational Linguistics, chapter Hypothesis Transformation and Semantic Variability Rules Used in Recognizing Textual Entailment, pages 125–130. http://aclweb.org/anthology/W07-1421.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P03-1054.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, pages 10–19. https://doi.org/10.18653/v1/S15-1002.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR* abs/1605.09090. http://arxiv.org/abs/1605.09090.

Bill MacCartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford University.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '08, pages 521–528. http://dl.acm.org/citation.cfm?id=1599081.1599147.

Yashar Mehdad, Alessandro Moschitti, and Massimo Fabio Zanzotto. 2010. Syntactic/semantic structures for textual entailment recognition. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1020–1028. http://aclweb.org/anthology/N10-1146.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 130–136. https://doi.org/10.18653/v1/P16-2022.

Tsendsuren Munkhdalai and Hong Yu. 2016a. Neural semantic encoders. *CoRR* abs/1607.04315. http://arxiv.org/abs/1607.04315.

Tsendsuren Munkhdalai and Hong Yu. 2016b. Neural tree indexers for text understanding. *CoRR* abs/1607.04492. http://arxiv.org/abs/1607.04492.

Biswajit Paria, K. M. Annervaz, Ambedkar Dukkipati, Ankush Chatterjee, and Sanjay Podder. 2016. A neural architecture mimicking humans end-to-end for natural language inference. *CoRR* abs/1611.04741. http://arxiv.org/abs/1611.04741.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2249–2255. http://aclweb.org/anthology/D16-1244.

Barbara Partee. 1995. Lexical semantics and compositionality. *Invitation to Cognitive Science* 1:311–360.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1532–1543. https://doi.org/10.3115/v1/D14-1162.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR* abs/1509.06664. http://arxiv.org/abs/1509.06664.

Alexander Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 379–389. https://doi.org/10.18653/v1/D15-1044.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read LSTM unit for textual entailment recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 2870–2879. http://aclweb.org/anthology/C16-1270.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. Omnipress, pages 129–136.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, D. Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1631–1642. http://aclweb.org/anthology/D13-1170.

Sheng Kai Tai, Richard Socher, and D. Christopher Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1556–1566. https://doi.org/10.3115/v1/P15-1150.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *CoRR* abs/1511.06361. http://arxiv.org/abs/1511.06361.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1442–1451. https://doi.org/10.18653/v1/N16-1170.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. pages 2048–2057. http://jmlr.org/proceedings/papers/v37/xuc15.html.

Junbei Zhang, Xiaodan Zhu, Qian Chen, Lirong Dai, Si Wei, and Hui Jiang. 2017. Exploring question understanding and adaptation in neural-network-based question answering. *CoRR* abs/arXiv:1703.04617v2. https://arxiv.org/abs/1703.04617.

Xiaodan Zhu, Hongyu Guo, Saif Mohammad, and Svetlana Kiritchenko. 2014. An empirical study on the effect of negation words on sentiment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 304–313. https://doi.org/10.3115/v1/P14-1029.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. pages 1604–1612. http://jmlr.org/proceedings/papers/v37/zhub15.html.

Linguistic analysis of differences in portrayal of movie characters

Anil Ramakrishna[1], Victor R. Martínez[1], Nikolaos Malandrakis[1], Karan Singla[1], and Shrikanth Narayanan[1,2]

[1]Department of Computer Science
[2]Department of Electrical Engineering
University of Southern California, Los Angeles, USA
{*akramakr, victorrm, malandra, singlak*}*@usc.edu, shri@sipi.usc.edu*

Abstract

We examine differences in portrayal of characters in movies using psycholinguistic and graph theoretic measures computed directly from screenplays. Differences are examined with respect to characters' gender, race, age and other metadata. Psycholinguistic metrics are extrapolated to dialogues in movies using a linear regression model built on a set of manually annotated seed words. Interesting patterns are revealed about relationships between genders of production team and the gender ratio of characters. Several correlations are noted between gender, race, age of characters and the linguistic metrics.

1 Introduction

Movies are often described as having the power to influence individual beliefs and values. In (Cape, 2003), the authors assert movies' influence in both creating new thinking patterns in previously unexplored social phenomena, especially in children, as well as their ability to update an individual's existing social boundaries based on what is shown on screen as the "norm". Some authors claim the inverse (Wedding and Boyd, 1999): that movies reflect existing cultural values of the society, adding weight to their ability in influencing individual beliefs of what is accepted as the norm. As a result, they are studied in multiple disciplines to analyze their influence.

Movies are particularly scrutinized in aspects involving negative stereotyping (Cape, 2003; Dimnik and Felton, 2006; Ter Bogt et al., 2010; Hedley, 1994) since this may introduce questionable beliefs in viewers. Negative stereotyping is believed to impact society in multiple aspects such as self-induced undermining of ability (Davies et al., 2005) as well as causing forms of prejudice that can impact leadership or employment prospects (Eagly and Karau, 2002; Niven, 2006). Studies in analyzing stereotyping in movies typically rely on collecting manual annotations on a small set of movies on which hypotheses tests are conducted (Behm-Morawitz and Mastro, 2008; Benshoff and Griffin, 2011; Hooks, 2009). In this work, we present large scale automated analyses of movie characters using language used in dialogs to study stereotyping along factors such as gender, race and age.

Language use has been long known as a strong indicator of the speaker's psychological and emotional state (Gottschalk and Gleser, 1969) and is well studied in a number of applications such as automatic personality detection (Mairesse et al., 2007) and psychotherapy (Xiao et al., 2015; Pennebaker et al., 2003). Computational analysis of language has been particularly popular thanks to advancements in computing and the ease of conducting large scale analysis of text on computers (Pennebaker et al., 2015).

To perform our analysis, we construct a new movie screenplay corpus [1] that includes nearly 1000 movie scripts obtained from the Internet. For each movie in the corpus, we obtain additional metadata such as cast, genre, writers and directors, and also collect actor level demographic information such as gender, race and age.

We use two kinds of measures in our analyses: (i) linguistic metrics that capture various psychological constructs and behaviors, estimated using dialogues from the screenplay; and (ii) graph theoretic metrics estimated from character network graphs, which are constructed to model inter-character interactions in the movie. The linguistic metrics include psycholinguistic normatives,

[1] http://sail.usc.edu/mica/text_corpus_release.php

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1669–1678
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1153

which provide word level scores on a numeric scale which are then aggregated at the dialog level, and metrics from the Linguistic Inquiry and Word Counts tool (LIWC) which capture usage of well studied stereotyping dimensions such as sexuality. We estimate centrality metrics from the character network graphs to measure relative importance of the different characters, which are analyzed with respect to the different factors of gender, race and age.

The main contributions of this work are as follows: (i) we present a scalable analysis of differences in portrayal of various character subgroups in movies using their language use, (ii) we construct a new corpus with detailed annotations for our analysis and (iii) we highlight several differences in the portrayal of characters along factors such as race, age and gender.

The rest of the paper is organized as follows: in section 2 we describe related work. We explain the data collection process in section 3 and experimental procedure in section 4. We explain results in section 5 and conclude in section 6.

2 Related work

Previous works in studying representation in movies largely focus on relative frequencies, particularly on character gender. In (Smith et al., 2014), the authors studied 120 movies from around the globe which were manually annotated to capture information about character gender, age, careers, writer gender and director gender. However, since the annotations are done manually, collecting information on new movies is a laborious process. We avoided this by estimating the metadata computationally, enabling us to scale up efficiently.

Automated analyses of movies using computational techniques to analyze representation has recently gained some attention. In (NYFA, 2013; Polygraph, 2016), the authors examine differences in relative frequency of female characters and note considerable disparities in gender ratio in these movies. However, the analyses there too are limited to comparing relative frequencies. Our work is closest to (Ramakrishna et al., 2015) where the authors study difference in language used in movies across genders, but their analysis is one dimensional. In our work we perform fine grained comparisons of character portrayal using multiple language based metrics along factors such as gender, race and age on a newly created corpus.

3 Data

3.1 Raw screenplay

We fetch movie screenplay files from two primary sources: imsdb (IMSDb, 2017) and daily scripts (DailyScript, 2017). In total, we retrieved 1547 movies. After removing duplicates we retain 1434 raw screenplay files, of which 489 were corrupted or empty leaving us with 945 usable screenplays. Tables 1, 3 and 4 list statistics about the corpus.

3.2 Script parser

The screenplay files are formatted in human readable format and include dialogues tagged with character names along with auxiliary information of the scene such as shot location (interior/exterior), character placement and scene context. The screenplays are from a diverse set of writers and include a significant amount of noise and inconsistencies in their structure. To extract the relevant information, we developed a text parser [2] that accepts raw script files and outputs utterances along with character names. We ignore scene context information and primarily focus on spoken dialogues to study language usage in the movies.

3.3 Movie and character meta-data

For each parsed movie, we fetch relevant metadata such as year of release, directors, writers, and producers from the Internet Movie Database (IMDb, 2017).

Since most screenplays are drafts and subject to revisions such as changes in character names, matching them to an entry from IMDb is non-trivial. We first start with a list of all movies that have a close match with the screenplay name; given this list of potential matches we compute name alignment scores for each entry as the percentage of character names from the script found online. The character names are mapped using term frequency-inverse document frequency (TFIDF) to compute the name alignment score following (Cohen et al., 2003). Finally, the entry with highest alignment score is chosen. For all actors listed in the aligned result, we collect their age, gender and race as detailed below.

[2] https://bitbucket.org/anil_ramakrishna/scriptparser

3.3.1 Gender

Given the names of actors and other members of production team found in a movie, we use a name based gender classifier to predict their gender information. Table 4 lists statistics on gender ratios for the production team in the corpus. Female-to-male ratios were found in close agreement with previous works (Smith et al., 2014).

As mentioned above, several screenplays get revised during production. In particular character names get changed, sometimes even gender. As a result, some characters may not be aligned to the correct entry from IMDb. In addition, digitized screenplays sometime include significant noise thanks to optical character recognition errors, leading to character names failing to align with entries from IMDb. To correct these, we perform manual cleanup of all the movie alignments, fix incorrect gender maps, and manually force match movies if they're mapped to the wrong IMDb entry.

3.3.2 Age

We also extract age for each actor to study possible age related biases in movies. We include age in our analysis since studies report preferential biases with age in employment particularly when combined with gender (Lincoln and Allen, 2004). In addition, there may be biases in portrayal of specific age groups when combined with gender and race.

For each actor in the mapped IMDb entry, we collect his/her birthday information. We subtract the movie production year obtained also from IMDb from the actor's birthday to get an estimate of the actor's age during the movie's production. We note however that the age obtained in this manner may be different from the portrayed age of the character. To account for this we bin the actors into fifteen year age groups before our analysis, since its generally unlikely to have actors further than fifteen years from their portrayed age.

3.3.3 Race

We parse ethnicity information from the website (ethnicelebs.com, 2017), which includes ethnicity for approximately 8000 different actors. The information obtained from this site is primarily submitted by independent users, and exhibits significant amount of variation among the possible ethnicities with about 750 different unique ethnicity types. Since we are more specifically interested in racial representations, we map the ethnicity types to race using Amazon Mechanical Turk (MTurk). We use a modified version of the racial categories from the US census which are listed in Table 1 along with frequency of actors from each racial category in our corpus.

Race	*# Actors*	*Percentage*
African	585	7.44%
Caucasian	6539	83.24%
East Asian	73	0.93%
Latino/Hispanic	161	2.05%
Native American	15	0.19%
Pacific Islander	5	0.063%
South Asian	43	0.547%
Mixed	434	5.52%

Table 1: Racial categories

The ethnicities obtained from the site above primarily cover major actors with a fan base with no information for several actors who play minor roles. We annotate racial information for nearly 2000 such actors using MTurk with two annotations for each actor, manually correcting nearly 400 cases in which the annotators disagreed.

4 Experiments

4.1 Character portrayal using language

To study differences in portrayal of characters, we use two different metrics: psycholinguistic normatives, which are designed to capture the underlying emotional state of the speaker; and LIWC metrics, which provide a measure of the speaker's affinity to different social and physical constructs such as religion and death. We explain these two metrics in detail below.

4.1.1 Psycholinguistic normatives

Psycholinguistic normatives provide a measure of various emotional and psychological constructs of the speaker, such as arousal, valence, concreteness, intelligibility, etc. and are computed entirely from language usage. They are relatively easy to compute, provide reliable indicators of the above constructs, and have been used in a variety of tasks in natural language processing such as information retrieval (Tanaka et al., 2013), sentiment analysis (Nielsen, 2011), text based personality prediction (Mairesse et al., 2007) and opinion mining.

The numeric ratings are typically extrapolated from a small set of keywords which are annotated

by psychologists. Manual annotations of word ratings is a laborious process and is hence limited to a few thousand words (Clark and Paivio, 2004). Automatic extrapolation of these ratings to words not covered by the manual annotations can be done using structured databases which provide relationships between words such as synonymy and hyponymy (Liu et al., 2014), or using context based semantic similarity.

In this work, we use the model described in (Malandrakis and Narayanan, 2015) where the authors use linear regression to compute normative scores for an input word w based on its similarity to a set of concept words s_i.

$$r(w) = \theta_0 + \sum_i \theta_i \cdot sim(w, s_i) \qquad (1)$$

where, $r(w)$ is the computed normative score for word w, θ_0 and θ_i are regression coefficients and sim is similarity between the given word w and concept words s_i.

The concept words can either be hand crafted suitably for the domain or chosen automatically from data. Similar to (Malandrakis and Narayanan, 2015), we create training data by posing queries on the Yahoo search engine from words of the aspell spell checker of which top 500 previews are collected from each query. From this corpus, the top 10000 most frequent words with atleast 3 characters were were used as concept words in extrapolation of all the norms. The linear regression model is trained using normative ratings for the manually annotated words by computing their similarity to the concept words. The similarity function sim is the cosine of binary context vectors with window size 1. The computed normatives are in the range $[-1, 1]$.

The psycholinguistic normatives used in this work are listed in Table 2. *Valence* is the degree of positive or negative emotion evoked by the word. *Arousal* is a measure of excitement in the speaker. Valence and arousal combined are common indicators used to map emotions. *Age of Acquisition* refers to the average age at which the word is learned and it denotes sophistication of language use. *Gender Ladenness* is a measure of masculine or feminine association of a word. 10 fold Cross Validation tests are performed on the normative scores predicted by the regression model given by equation 1. Correlation coefficients of the selected normatives with the manual annotations are as follows: Arousal (0.7), Valence (0.88), Age of Acquisition (0.86) and Gender Ladenness (0.8). The high correlations render confidence in the psycholinguistic models.

In our experiments, the normative scores are computed on content words from each dialog. We filter out all words other than nouns, verbs, adjectives and adverbs. Word level scores are aggregated at the dialog level using arithmetic mean.

4.1.2 Linguistic inquiry and word counts (LIWC)

LIWC is a text processing application that processes raw text and outputs percentage of words from the text that belong to linguistic, affective, perceptual and other dimensions. It operates by maintaining a diverse set of dictionaries of words each belonging to a unique dimension. Input texts are processed word by word; each word is searched in the internal dictionaries and the corresponding counter is incremented if a word is found in that dictionary. Finally, percentage of words from the input text belonging to the different dimensions are returned.

For our experiments, we treat each utterance in the movie as a unique document and obtain values for the LIWC metrics. Table 2 lists the metrics used in our experiments.

4.2 Character network analytics

In order to study representation of the different subgroups as major characters in movies, we construct a network of interaction between characters using which we compute importance measures for each character. From each movie script, we construct an undirected and unweighted graph where nodes represent characters. We place an edge e_{ab} if two characters A and B interact at least once in the movie. For our experiments we assume interaction between A and B if there is at least one scene in which one speaks right after another. This graph creation method based on scene co-occurrence is similar to the approach used in (Beveridge and Shan, 2016).

We estimate different measures of a node's importance within the character network and use it as proxy for the character's importance. We employ two types of centralities: *betweenness centrality*, the number of shortest paths that go through the node, and *degree centrality*, which is the number of edges incident on a node. These centrality measurements have been previously used in the con-

Psycholinguistic norms	Valence, Arousal, Age of Acquisition, Gender Ladenness
LIWC metrics	Achievement, Religion, Death, Sexual, Swear

Table 2: Psycholinguistic Normatives and LIWC metrics used in analysis

	male	*female*	*total*
# Characters	4899	2008	6907
# Dialogues	375711	154897	530608
Number of movies			945

Table 3: Character statistics

role	*male*	*female*	*total*
Writers	1326	169	1495
Directors	544	46	590
Producers	2866	870	3736
Casting Directors	135	275	410
Distributing Companies			2701

Table 4: Production team statistics

	character genders	
	f (28.9%)	*m* (71.1%)
f	249 (41.2%)	356 (58.8%)
m	1541 (27.6%)	4040 (72.4%)

(a) writers gender

f	114 (39.3%)	176 (60.7%)
m	1676 (28.4%)	4220 (71.6%)

(b) directors gender

f	1374 (29.1%)	3350 (70.9%)
m	416 (28.5%)	1046 (71.5%)

(c) casting directors gender

Table 5: Contingency tables for character gender v/s writers, directors and casting directors' gender; f: female and m: male; each cell gives frequency of character gender for that column and production member gender for that row, numbers in braces indicate row wise proportion of character gender

text of books, films and comics (Beveridge and Shan, 2016; Bonato et al., 2016; Alberich et al., 2002; Ribeiro et al., 2016).

5 Results

We study differences in various subgroups along multiple facets. We first report results on differences in character ratios from each subgroup since this has implications on employment and can have social-economic effects (Niven, 2006). We next use psycholinguistic normatives and LIWC metrics described in the previous section to study differences in character portrayal along the primary markers: age, gender and race. We finally use the graph theoretic centrality measures to estimate characters' importance and analyze differences among the different subgroups.

Since we are interested in character level analytics, we treat all utterances from the character as a single document to compute the aggregate language metrics. We perform all our experiments using non-parametric statistical tests since the data fails to satisfy preconditions such as normality and homoscedasticity required for parametric tests such as ANOVA.

5.1 Difference in relative frequency of subgroups

We first filter our characters with unknown gender/race/age leaving us with 6907 characters in total. Table 3 lists the number of characters and dialogues from each gender. As noted in previous studies, the ratio is considerably skewed with male actors having nearly twice as many roles and dialogues compared to female actors. Table 4 lists relative frequency among male and female members of the production team. Table 1 lists the percentage of actors belonging to different racial categories in the corpus.

We perform chi-squared tests between character gender and gender of production team members who are most likely to influence characters gender: writers, directors and casting directors. Table 5 shows contingency tables with gender frequencies for each of these cases along with percentages. Note we filter out nearly 100 movies for this test in which the gender of the production team members was unknown. Of the three tests we perform, character gender distributions for writer and director genders are significantly different from the overall character gender distribution ($p < 10^{-10}$ and $p < 10^{-4}$ respectively; $\alpha = 0.05$). In particular, female writers and directors appear to produce movies with relatively balanced gender proportions (still slightly skewed towards the male side) compared to male writers

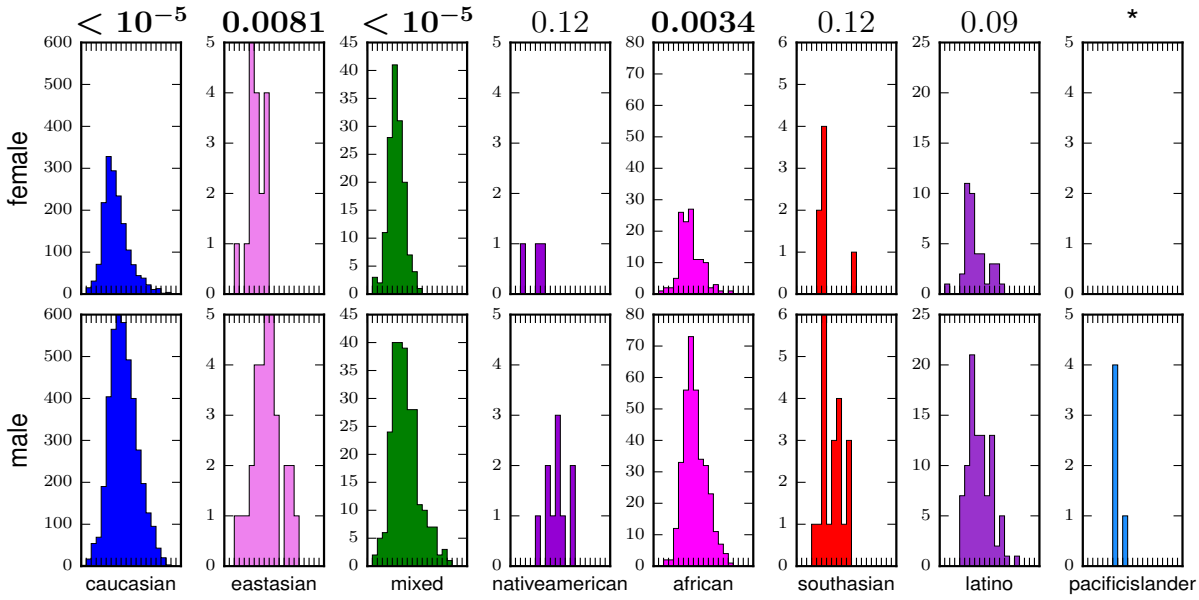


Figure 1: Histogram of age for actors belonging to different gender and racial categories with p-values on top; significant values at $\alpha = 0.05$ are highlighted; *: no test performed since the female group is empty

and directors. Casting directors however appear to have no influence on gender of the characters.

Studies report potential biases in actor employment with age (Lincoln and Allen, 2004), particularly in female actors. To evaluate this, we plot histograms of age for male and female characters for each of the racial categories in Figure 1. The distribution of age for each category appears approximately normal, except for the *nativeamerican* and *pacificislander* character groups which have a small sample size. For most categories of race, the mode of the distribution for female actors appears to be at least five years less than the mode for male actors. To check for significance in this difference we conduct Mann-Whitney U tests on male and female age groups for each race with the resulting p-values shown in the figure. We ignore characters belonging to the *pacificislander* racial group since there are no female actors from this race in our corpus. The difference in age groups is significant in most categories with large sample sizes, suggesting possible preferences towards casting younger people when casting female actors.

5.2 Character portrayal using language

To analyze differences in portrayal of subgroups, we compute psycholinguistic normatives and LIWC metrics as described before. For each of the metrics listed in Table 2, we conduct non-parametric hypothesis tests to look for differences in samples from the subgroups. We treat the different metrics independently, performing statistical tests along each separately. We avoid statistical tests combining two or more factors since some of the resulting groups would be empty due to the skewed group sizes along race. We defer such analyses to future work.

	m (4894)	f (2008)	p
age of acq.	**−0.1590**	**−0.1715**	**$< 10^{-5}$**
arousal	0.0253	0.0246	0.41
gender	**−0.0312**	**−0.0055**	**$< 10^{-5}$**
valence	**0.2284**	**0.2421**	**$< 10^{-5}$**
sex	0.00015	0.0000	0.08
achieve	**0.0087**	**0.0080**	**$< 10^{-5}$**
religion	0.0025	0.0022	0.10
death	**0.0025**	**0.0016**	**$< 10^{-5}$**
swear	**0.0037**	**0.0015**	**$< 10^{-5}$**

Table 6: Median values for male and female characters along with p values obtained by comparing the two groups using Mann-Whitney U test; highlighted differences are significant at $\alpha = 0.05$

5.2.1 Gender

We perform Mann-Whitney U tests between male and female characters along the nine dimensions and the results are shown in Table 6. In all of

the cases, higher values imply higher degree of the corresponding dimension, except for valence in which higher values imply positive valence (attractiveness) and lower values imply negative valence (averseness). The difference between male and female characters are statistically significant along six of the nine dimensions. The results indicate slightly higher age of acquisition scores for male characters. Regarding gender ladenness, male characters appear to be closer to the masculine side than female characters on average, agreeing with previous results.

Our results also indicate that female character utterances tend to be more positive in valence compared to male characters while male characters seem to have higher percentage of words related to achievement. In addition, male characters appear to be more frequent in using words related to death as well as swear words compared to female characters.

5.2.2 Race

To study differences in portrayal of the racial categories, we perform Kruskal-Wallis test (a generalization of Mann-Whitney U test for more than two groups) on each of the nine metrics with race as the independent variable. We found significant differences in distribution of samples for gender ladenness, sexuality, religion and swear words. For gender ladenness, *caucasian* and *mixed* race characters have significantly higher medians than *african* and *nativeamerican* characters. In sexuality, *latino* and *mixed* race characters were found to have higher median than at least one other racial group with significance indicating a higher degree of sexualization in these characters. *Eastasian* characters were found to be significantly lower than medians of three other races (*caucasian*, *african* and *mixed*) in using words with religious connotations. In swear word usage, the only significant difference found is between *caucasian* and *african* characters with *african* characters using higher percentage of swear words. In all of the above cases, significance was tested at $\alpha = 0.05$.

5.2.3 Age

To examine the relationship between age and the different metrics, we build separate linear regression models with each dimension as the dependent variable and character age as the independent variable. Table 7 reports regression coefficients for age along with p values for each dimension. The positive coefficient for age of acquisition indicates an increase in sophistication of word usage with age. Arousal, on the other hand, has a significant negative coefficient indicating a decrease in activation, on average, as character age increases. Gender ladenness also has a significant negative coefficient indicating that as age increases, the average gender ladenness value decreases. Similar trends are observed for sexuality and swear word usage. Usage of words related to achievement and religion however, seem to increase with age.

	$\beta_1(\times 10^{-3})$	*p-value*
age of acq.	**3.9**	$< 10^{-10}$
arousal	**-1.1**	$< 10^{-10}$
gender	**-2.5**	$< 10^{-10}$
valence	0.078	0.7
sex	**-0.25**	$< 10^{-5}$
achieve	**0.26**	$< 10^{-10}$
religion	**0.12**	0.001
death	-0.039	0.2
swear	**-0.34**	$< 10^{-5}$

Table 7: Coefficients of age for linear regression models along each dimension along with p-values; highlighted cells are significant at $\alpha = 0.05$

5.3 Character network analytics

To study differences in major roles assigned to the different subgroups, we compute two centrality metrics from the character network graph constructed for each movie: *degree* centrality measures the number of unique characters that interact with a given character, *betweenness* centrality measures how much would the plot be disrupted if said character was to disappear completely, i.e., how important is a character to the overall plot. Similar to the language analyses from previous section, we test differences in these metrics along the three factors of gender, race and age. All statistical tests reported below are conducted at $\alpha = 0.05$.

5.3.1 Gender

Male characters were found to have higher values in the two metrics compared to female characters but the differences were not statistically significant. Motivated by studies (Sapolsky et al., 2003; Linz et al., 1984) which report interactions between genre and gender, we performed Mann-Whitney U tests between male and female char-

acters given different genres. To avoid type I errors we corrected for multiple comparisons using the Holm-Bonferroni correction. Significant differences were found only in horror movies where the median *degree* centrality for females (0.221) was higher than the median *degree* centrality of males (0.166). This is in agreement with prior studies which report female characters to have a more prominent presence in horror movies, particularly as victims of violent scenes (Welsh and Brantford, 2009).

5.3.2 Race

To examine differences in major roles across the racial categories, we perform Kruskal-Wallis tests similar to previous subsection. Significant differences were found with both *degree* and *betweenness* centrality measures ($p < 0.001$; $\alpha = 0.05$).

Latino characters were found to have significantly lower *degree* centralities compared to *caucasian* and *southasian* races suggesting non-central roles in these characters. *Caucasian* characters were found to have median *betweenness* centralities significantly higher than at least one other race. Characters from the *nativeamerican* race exhibit significantly lower medians in both *degree* and *betweenness* centralities than *caucasian*, *african* and *mixed* characters, which agrees with (Rosenthal, 2012).

5.3.3 Age

We investigate the effects of age on importance of character roles by building a linear regression model on the two centralities with age as the independent variable. In both cases, age was found to be significant ($p < 0.001$; $\alpha = 0.05$). With *degree* centrality, the regression coefficient β was found to be equal to 0.003. In *betweenness* centrality, the regression coefficient was also positive, given by $\beta = 8.41 \times 10^{-4}$. Both these metrics indicate a positive correlation for character importance with age, i.e. as characters age, there is an increased interaction with other characters in the movie as well as higher prominence in the movie plot.

6 Conclusion

We present a scalable automated analyses of differences in character portrayal along multiple factors such as gender, race and age using word usage, psycholinguistic and graph theoretic measures. Several interesting patterns are revealed in the analysis. In particular, movies with female writers and directors in the production team are observed to have balanced gender ratios in characters compared to male writers/directors. Across several races, female actors are found to be younger than male actors on average.

Female characters appear to be more positive in language use with fewer references to death and fewer swear words compared to male characters. Female characters also appear to be more prominent in horror movies compared to male characters. Latino and mixed race characters appear to have higher usage of sexual words. Eastasian characters seem to use significantly fewer religious words. As characters aged, their word sophistication seems to increase along with usage of words related to achievement and religion; there was also a significant reduction in word activation, usage of sexual and swear words as character age increases.

Future work includes expanding the analyses to non-English movies and combining the linguistic metrics with character networks. Specifically, character network edges can be weighted using the psycholinguistic metrics to analyze the emotional patterns in inter-character interactions.

7 Acknowledgments

We acknowledge support from NSF and our partnership with Google and the Geena Davis Institute on Gender in Media.

We thank Naveen Kumar for all the helpful discussions and feedback during this work.

References

Ricardo Alberich, Joe Miro-Julia, and Francesc Rosselló. 2002. Marvel universe looks almost like a real social network. *arXiv preprint cond-mat/0202174* .

Elizabeth Behm-Morawitz and Dana E Mastro. 2008. Mean girls? the influence of gender portrayals in teen movies on emerging adults' gender-based attitudes and beliefs. *Journalism & Mass Communication Quarterly* 85(1):131–146.

Harry M Benshoff and Sean Griffin. 2011. *America on film: Representing race, class, gender, and sexuality at the movies*. John Wiley & Sons.

Andrew Beveridge and Jie Shan. 2016. Network of thrones. *Math Horizons* 23(4):18–22.

Anthony Bonato, David Ryan D'Angelo, Ethan R Elenberg, David F Gleich, and Yangyang Hou. 2016.

Mining and modeling character networks. In *Algorithms and Models for the Web Graph: 13th International Workshop, WAW 2016, Montreal, QC, Canada, December 14–15, 2016, Proceedings 13*. Springer, pages 100–114.

Gavin S Cape. 2003. Addiction, stigma and movies. *Acta Psychiatrica Scandinavica* 107(3):163–169.

James M Clark and Allan Paivio. 2004. Extensions of the paivio, yuille, and madigan (1968) norms. *Behavior Research Methods, Instruments, & Computers* 36(3):371–383.

William Cohen, Pradeep Ravikumar, and Stephen Fienberg. 2003. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*. volume 3, pages 73–78.

DailyScript. 2017. The daily script. [Online; accessed 1-February-2017]. http://dailyscript.com/.

Paul G Davies, Steven J Spencer, and Claude M Steele. 2005. Clearing the air: identity safety moderates the effects of stereotype threat on women's leadership aspirations. *Journal of personality and social psychology* 88(2):276.

Tony Dimnik and Sandra Felton. 2006. Accountant stereotypes in movies distributed in north america in the twentieth century. *Accounting, Organizations and Society* 31(2):129–155.

Alice H Eagly and Steven J Karau. 2002. Role congruity theory of prejudice toward female leaders. *Psychological review* 109(3):573.

ethnicelebs.com. 2017. Celebrity ethnicity. [Online; accessed 1-February-2017]. http://ethnicelebs.com.

Louis August Gottschalk and Goldine C Gleser. 1969. *The measurement of psychological states through the content analysis of verbal behavior*. Univ of California Press.

Mark Hedley. 1994. The presentation of gendered conflict in popular movies: Affective stereotypes, cultural sentiments, and men's motivation. *Sex Roles* 31(11-12):721–740.

Bell Hooks. 2009. *Reel to real: race, class and sex at the movies*. Routledge.

IMDb. 2017. Internet movie database. [Online; accessed 1-February-2017]. http://www.imdb.com/.

IMSDb. 2017. Internet movie script database. [Online; accessed 1-February-2017]. http://www.imsdb.com/.

Anne E Lincoln and Michael Patrick Allen. 2004. Double jeopardy in hollywood: Age and gender in the careers of film actors, 1926–1999. In *Sociological Forum*. Springer, volume 19, pages 611–631.

Daniel Linz, Edward Donnerstein, and Steven Penrod. 1984. The effects of multiple exposures to filmed violence against women. *Journal of Communication* 34(3):130–147.

Ting Liu, Kit Cho, George Aaron Broadwell, Samira Shaikh, Tomek Strzalkowski, John Lien, Sarah M Taylor, Laurie Feldman, Boris Yamrom, Nick Webb, et al. 2014. Automatic expansion of the mrc psycholinguistic database imageability ratings. In *LREC*. pages 2800–2805.

François Mairesse, Marilyn A Walker, Matthias R Mehl, and Roger K Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of artificial intelligence research* 30:457–500.

Nikolaos Malandrakis and Shrikanth S Narayanan. 2015. Therapy language analysis using automatically generated psycholinguistic norms. In *INTERSPEECH*. pages 1952–1956.

Finn Årup Nielsen. 2011. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903* .

David Niven. 2006. Throwing your hat out of the ring: Negative recruitment and the gender imbalance in state legislative candidacy. *Politics & Gender* 2(04):473–489.

NYFA. 2013. Gender inequality in film. [Online; accessed 1-February-2017]. https://www.nyfa.edu/film-school-blog/gender-inequality-in-film/.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. Technical report.

James W Pennebaker, Matthias R Mehl, and Kate G Niederhoffer. 2003. Psychological aspects of natural language use: Our words, our selves. *Annual review of psychology* 54(1):547–577.

Polygraph. 2016. Film dialogue from 2,000 screenplays, broken down by gender and age. [Online; accessed 1-February-2017]. http://polygraph.cool/films/.

Anil Ramakrishna, Nikolaos Malandrakis, Elizabeth Staruk, and Shrikanth S Narayanan. 2015. A quantitative analysis of gender differences in movies using psycholinguistic normatives. In *EMNLP*. pages 1996–2001.

Mauricio Aparecido Ribeiro, Roberto Antonio Vosgerau, Maria Larissa Pereira Andruchiw, and Sandro Ely de Souza Pinto. 2016. The complex social network of the lord of rings. *Revista Brasileira de Ensino de Física* 38(1).

Nicolas G Rosenthal. 2012. *Reimagining Indian country: native American migration and identity in twentieth-century Los Angeles*. Univ of North Carolina Press.

Burry S Sapolsky, Fred Molitor, and Sarah Luque. 2003. Sex and violence in slasher films: Re-examining the assumptions. *Journalism & Mass Communication Quarterly* 80(1):28–38.

Stacy L Smith, Marc Choueiti, and Katherine Pieper. 2014. Gender bias without borders: An investigation of female characters in popular films across 11 countries. *USC Annenberg* 5.

Shinya Tanaka, Adam Jatowt, Makoto P Kato, and Katsumi Tanaka. 2013. Estimating content concreteness for finding comprehensible documents. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, pages 475–484.

Tom FM Ter Bogt, Rutger CME Engels, Sanne Bogers, and Monique Kloosterman. 2010. "shake it baby, shake it": Media preferences, sexual attitudes and gender stereotypes among adolescents. *Sex Roles* 63(11-12):844–859.

Danny Wedding and Mary Ann Boyd. 1999. Movies & mental illness: Using films to understand psychopathology. .

Andrew Welsh and Laurier Brantford. 2009. Sex and violence in the slasher horror film: A content analysis of gender differences in the depiction of violence. *Journal of Criminal Justice and Popular Culture* 16(1):1–25.

Bo Xiao, Zac E Imel, Panayiotis G Georgiou, David C Atkins, and Shrikanth S Narayanan. 2015. "rate my therapist": Automated detection of empathy in drug and alcohol counseling via speech and language processing. *PloS one* 10(12):e0143055.

Linguistically Regularized LSTM for Sentiment Classification

Qiao Qian[1], **Minlie Huang**[1]*, **Jinhao Lei**[2], **Xiaoyan Zhu**[1]
[1]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China
[2]Dept. of Thermal Engineering, Tsinghua University, Beijing 100084, PR China
qianqiaodecember29@126.com, aihuang@tsinghua.edu.cn
leijh14@gmail.com, zxy-dcs@tsinghua.edu.cn

Abstract

This paper deals with sentence-level sentiment classification. Though a variety of neural network models have been proposed recently, however, previous models either depend on expensive phrase-level annotation, most of which has remarkably degraded performance when trained with only sentence-level annotation; or do not fully employ linguistic resources (e.g., sentiment lexicons, negation words, intensity words). In this paper, we propose simple models trained with sentence-level annotation, but also attempt to model the linguistic role of sentiment lexicons, negation words, and intensity words. Results show that our models are able to capture the linguistic role of sentiment words, negation words, and intensity words in sentiment expression.

1 Introduction

Sentiment classification aims to classify text to sentiment classes such as *positive or negative*, or more fine-grained classes such as *very positive, positive, neutral, etc*. There has been a variety of approaches for this purpose such as lexicon-based classification (Turney, 2002; Taboada et al., 2011), and early machine learning based methods (Pang et al., 2002; Pang and Lee, 2005), and recently neural network models such as convolutional neural network (CNN) (Kim, 2014; Kalchbrenner et al., 2014; Lei et al., 2015), recursive autoencoders (Socher et al., 2011, 2013), Long Short-Term Memory (LSTM) (Mikolov, 2012; Chung et al., 2014; Tai et al., 2015; Zhu et al., 2015), and many more.

In spite of the great success of these neural models, there are some defects in previous studies. First, tree-structured models such as recursive autoencoders and Tree-LSTM (Tai et al., 2015; Zhu et al., 2015), depend on parsing tree structures and expensive phrase-level annotation, whose performance drops substantially when only trained with sentence-level annotation. Second, linguistic knowledge such as sentiment lexicon, negation words or *negators* (e.g., *not, never*), and intensity words or *intensifiers* (e.g., *very, absolutely*), has not been fully employed in neural models.

The goal of this research is to developing simple sequence models but also attempts to fully employing linguistic resources to benefit sentiment classification. Firstly, we attempts to develop simple models that do not depend on parsing trees and do not require phrase-level annotation which is too expensive in real-world applications. Secondly, in order to obtain competitive performance, simple models can benefit from linguistic resources. Three types of resources will be addressed in this paper: sentiment lexicon, negation words, and intensity words. Sentiment lexicon offers the prior polarity of a word which can be useful in determining the sentiment polarity of longer texts such as phrases and sentences. Negators are typical sentiment shifters (Zhu et al., 2014), which constantly change the polarity of sentiment expression. Intensifiers change the valence degree of the modified text, which is important for fine-grained sentiment classification.

In order to model the linguistic role of sentiment, negation, and intensity words, our central idea is to regularize the difference between the predicted sentiment distribution of the current position [1], and that of the previous or next positions, in a sequence model. For instance, if the cur-

*Corresponding Author: Minlie Huang

[1]Note that in sequence models, the hidden state of the current position also encodes forward or backward contexts.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1679–1689
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1154

rent position is a negator *not*, the negator should change the sentiment distribution of the next position accordingly. To summarize, our contributions lie in two folds:

- We discover that modeling the linguistic role of sentiment, negation, and intensity words can enhance sentence-level sentiment classification. We address the issue by imposing linguistic-inspired regularizers on sequence LSTM models.

- Unlike previous models that depend on parsing structures and expensive phrase-level annotation, our models are simple and efficient, but the performance is on a par with the state-of-the-art.

The rest of the paper is organized as follows: In the following section, we survey related work. In Section 3, we briefly introduce the background of LSTM and bidirectional LSTM, and then describe in detail the lingistic regularizers for sentiment/negation/intensity words in Section 4. Experiments are presented in Section 5, and Conclusion follows in Section 6.

2 Related Work

2.1 Neural Networks for Sentiment Classification

There are many neural networks proposed for sentiment classification. The most noticeable models may be the recursive autoencoder neural network which builds the representation of a sentence from subphrases recursively (Socher et al., 2011, 2013; Dong et al., 2014; Qian et al., 2015). Such recursive models usually depend on a tree structure of input text, and in order to obtain competitive results, usually require annotation of all subphrases. Sequence models, for instance, convolutional neural network (CNN), do not require tree-structured data, which are widely adopted for sentiment classification (Kim, 2014; Kalchbrenner et al., 2014; Lei et al., 2015). Long short-term memory models are also common for learning sentence-level representation due to its capability of modeling the prefix or suffix context (Hochreiter and Schmidhuber, 1997). LSTM can be commonly applied to sequential data but also tree-structured data (Zhu et al., 2015; Tai et al., 2015).

2.2 Applying Linguistic Knowledge for Sentiment Classification

Linguistic knowledge and sentiment resources, such as sentiment lexicons, negation words (*not, never, neither, etc.*) or *negators*, and intensity words (*very, extremely, etc.*) or *intensifiers*, are useful for sentiment analysis in general.

Sentiment lexicon (Hu and Liu, 2004; Wilson et al., 2005) usually defines prior polarity of a lexical entry, and is valuable for lexicon-based models (Turney, 2002; Taboada et al., 2011), and machine learning approaches (Pang and Lee, 2008). There are recent works for automatic construction of sentiment lexicons from social data (Vo and Zhang, 2016) and for multiple languages (Chen and Skiena, 2014). A noticeable work that ultilizes sentiment lexicons can be seen in (Teng et al., 2016) which treats the sentiment score of a sentence as a weighted sum of prior sentiment scores of negation words and sentiment words, where the weights are learned by a neural network.

Negation words play a critical role in modifying sentiment of textual expressions. Some early negation models adopt the *reversing assumption* that a negator reverses the sign of the sentiment value of the modified text (Polanyi and Zaenen, 2006; Kennedy and Inkpen, 2006). The *shifting hyothesis* assumes that negators change the sentiment values by a constant amount (Taboada et al., 2011; Liu and Seneff, 2009). Since each negator can affect the modified text in different ways, the constant amount can be extended to be negator-specific (Zhu et al., 2014), and further, the effect of negators could also depend on the syntax and semantics of the modified text (Zhu et al., 2014). Other approaches to negation modeling can be seen in (Jia et al., 2009; Wiegand et al., 2010; Benamara et al., 2012; Lapponi et al., 2012).

Sentiment intensity of a phrase indicates the strength of associated sentiment, which is quite important for fine-grained sentiment classification or rating. Intensity words can change the valence degree (i.e., sentiment intensity) of the modified text. In (Wei et al., 2011) the authors propose a linear regression model to predict the valence value for content words. In (Malandrakis et al., 2013), a kernel-based model is proposed to combine semantic information for predicting sentiment score. In the SemEval-2016 task 7 subtask A, a learning-to-rank model with a pair-wise strategy is proposed to predict sentiment intensity scores (Wang

et al., 2016). Linguistic intensity is not limited to sentiment or intensity words, and there are works that assign low/medium/high intensity scales to adjectives such as *okay, good, great* (Sharma et al., 2015) or to gradable terms (e.g. *large, huge, gigantic*) (Shivade et al., 2015).

In (Dong et al., 2015), a sentiment parser is proposed, and the authors studied how sentiment changes when a phrase is modified by negators or intensifiers.

Applying linguistic regularization to text classification can be seen in (Yogatama and Smith, 2014) which introduces three linguistically motivated structured regularizers based on parse trees, topics, and hierarchical word clusters for text categorization. Our work differs in that (Yogatama and Smith, 2014) applies group lasso regularizers to logistic regression on model parameters while our regularizers are applied on intermediate outputs with KL divergence.

3 Long Short-term Memory Network

3.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory has been widely adopted for text processing. Briefly speaking, in LSTM, the hidden states h_t and memory cell c_t is a function of their previous c_{t-1} and h_{t-1} and input vector x_t, or formally as follows:

$$c_t, h_t = g^{(LSTM)}(c_{t-1}, h_{t-1}, x_t) \quad (1)$$

The hidden state $h_t \in R^d$ denotes the representation of position t while also encoding the preceding contexts of the position. For more details about LSTM, we refer readers to (Hochreiter and Schmidhuber, 1997).

3.2 Bidirectional LSTM

In LSTM, the hidden state of each position (h_t) only encodes the prefix context in a forward direction while the backward context is not considered. Bidirectional LSTM (Graves et al., 2013) exploited two parallel passes (forward and backward) and concatenated hidden states of the two LSTMs as the representation of each position. The forward and backward LSTMs are respectively formulated as follows:

$$\overrightarrow{c}_t, \overrightarrow{h}_t = g^{(LSTM)}(\overrightarrow{c}_{t-1}, \overrightarrow{h}_{t-1}, x_t) \quad (2)$$

$$\overleftarrow{c}_t, \overleftarrow{h}_t = g^{(LSTM)}(\overleftarrow{c}_{t+1}, \overleftarrow{h}_{t+1}, x_t) \quad (3)$$

where $g^{(LSTM)}$ is the same as that in Eq (1). Particularly, parameters in the two LSTMs are shared. The representation of the entire sentence is $[\overrightarrow{h}_n, \overleftarrow{h}_1]$, where n is the length of the sentence. At each position t, the new representation is $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$, which is the concatenation of hidden states of the forward LSTM and backward LSTM. In this way, the forward and backward contexts can be considered simultaneously.

4 Linguistically Regularized LSTM

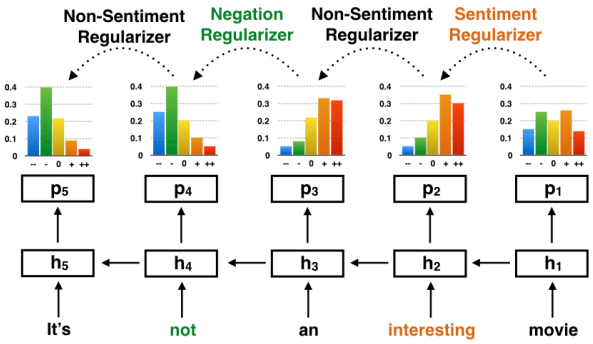


Figure 1: The overview of Linguistically Regularized LSTM. Note that we apply a backward LSTM (from right to left) to encode sentence since most negators and intensifiers are modifying their following words.

The central idea of the paper is to model the linguistic role of sentiment, negation, and intensity words in sentence-level sentiment classification by regularizing the outputs at adjacent positions of a sentence. For example in Fig 1, in sentence "*It's not an interesting movie*", the predicted sentiment distributions at "**an interesting movie*[2]" and "**interesting movie*" should be close to each other, while the predicted sentiment distribution at "**interesting movie*" should be quite different from the preceding positions (in the backward direction) ("**movie*") since a sentiment word ("*interesting*") is seen.

We propose a generic regularizer and three special regularizers based on the following linguistic observations:

- **Non-Sentiment Regularizer**: if the two adjacent positions are all non-opinion words, the sentiment distributions of the two positions should be close to each other. Though

[2]The asterisk denotes the current position.

this is not always true (e.g., *soap movie*), this assumption holds at most cases.

- **Sentiment Regularizer**: if the word is a sentiment word found in a lexicon, the sentiment distribution of the current position should be significantly different from that of the next or previous positions. We approach this phenomenon with a *sentiment class specific shifting distribution*.

- **Negation Regularizer**: Negation words such as "not" and "never" are critical sentiment shifter or converter: in general they shift sentiment polarity from the positive end to the negative end, but sometimes depend on the negation word and the words they modify. The negation regularizer models this linguistic phenomena with a *negator-specific transformation matrix*.

- **Intensity Regularizer**: Intensity words such as "very" and "extremely" change the valence degree of a sentiment expression: for instance, from *positive* to *very positive*. Modeling this effect is quite important for fine-grained sentiment classification, and the intensity regularizer is designed to formulate this effect by a *word-specific transformation matrix*.

More formally, the predicted sentiment distribution (p_t, based on h_t, see Eq. 5) at position t should be linguistically regularized with respect to that of the preceding $(t-1)$ or following $(t+1)$ positions. In order to enforce the model to produce coherent predictions, we plug a new loss term into the original cross entropy loss:

$$\mathcal{L}(\theta) = -\sum_i \hat{y}_i \log y_i + \alpha \sum_i \sum_t L_{t,i} + \beta ||\theta||^2 \quad (4)$$

where $\hat{y}_i$ is the gold distribution for sentence i, y_i is the predicted distribution, $L_{t,i}$ is one of the above regularizers or combination of these regularizers on sentence i, α is the weight for the regularization term, and t is the word position in a sentence.

Note that we do not consider the modification span of negation and intensity words to preserve the simplicity of the proposed models. Negation scope resolution is another complex problem which has been extensively studied (Zou et al., 2013; Packard et al., 2014; Fancellu et al., 2016), which is beyond the scope of this work. Instead, we resort to sequence LSTMs for encoding surrounding contexts at a given position.

4.1 Non-Sentiment Regularizer (NSR)

This regularizer constrains that the sentiment distributions of adjacent positions should not vary much if the additional input word x_t is not a sentiment word, formally as follows:

$$L_t^{(NSR)} = max(0, D_{KL}(p_t||p_{t-1}) - M) \quad (5)$$

where M is a hyperparameter for margin, p_t is the predicted distribution at state of position t, (i.e., h_t), and $D_{KL}(p||q)$ is a symmetric KL divergence defined as follows:

$$D_{KL}(p||q) = \frac{1}{2}\sum_{l=1}^{C} p(l)\log q(l) + q(l)\log p(l) \quad (6)$$

where p, q are distributions over sentiment labels l and C is the number of labels.

4.2 Sentiment Regularizer (SR)

The sentiment regularizer constrains that the sentiment distributions of adjacent positions should drift accordingly if the input word is a sentiment word. Let's revisit the example "*It's not an interesting movie*" again. At position $t = 2$ (in the backward direction) we see a positive word "*interesting*" so the predicted distribution would be more positive than that at position $t = 1$ (*movie*). This is the issue of *sentiment drift*.

In order to address the sentiment drift issue, we propose a polarity shifting distribution $s_c \in R^C$ for each sentiment class defined in a lexicon. For instance, a sentiment lexicon may have class labels like *strong positive*, *weakly positive*, *weakly negative*, and *strong negative*, and for each class, there is a shifting distribution which will be learned by the model. The sentiment regularizer states that if the current word is a sentiment word, the sentiment distribution drift should be observed in comparison to the previous position, in more details:

$$p_{t-1}^{(SR)} = p_{t-1} + s_{c(x_t)} \quad (7)$$

$$L_t^{(SR)} = max(0, D_{KL}(p_t||p_{t-1}^{(SR)}) - M) \quad (8)$$

where $p_{t-1}^{(SR)}$ is the drifted sentiment distribution after considering the shifting sentiment distribution corresponding to the state at position t, $c(x_t)$

is the prior sentiment class of word x_t, and $s_c \in \theta$ is a parameter to be optimized but could also be set fixed with prior knowledge. Note that in this way all words of the same sentiment class share the same drifting distribution, but in a refined setting, we can learn a shifting distribution for each sentiment word if large-scale datasets are available.

4.3 Negation Regularizer (NR)

The negation regularizer approaches how negation words shift the sentiment distribution of the modified text. When the input x_t is a negation word, the sentiment distribution should be shifted/reversed accordingly. However, the negation role is more complex than that by sentiment words, for example, the word *"not"* in *"not good"* and *"not bad"* have different roles in polarity change. The former changes the polarity to *negative*, while the latter changes to *neutral* instead of *positive*.

To respect such complex negation effects, we propose a transformation matrix $T_m \in R^{C \times C}$ for each negation word m, and the matrix will be learned by the model. The regularizer assumes that if the current position is a negation word, the sentiment distribution of the current position should be close to that of the next or previous position with the transformation.

$$p_{t-1}^{(NR)} = softmax(T_{x_j} \times p_{t-1}) \tag{9}$$

$$p_{t+1}^{(NR)} = softmax(T_{x_j} \times p_{t+1}) \tag{10}$$

$$L_t^{(NR)} = min \begin{cases} max(0, D_{KL}(p_t || p_{t-1}^{(NR)}) - M) \\ max(0, D_{KL}(p_t || p_{t+1}^{(NR)}) - M) \end{cases} \tag{11}$$

where $p_{t-1}^{(NR)}$ and $p_{t+1}^{(NR)}$ is the sentiment distuibution after transformation, $T_{x_j} \in \theta$ is the transformation matrix for a negation word x_j, a parameter to be learned during training. In total, we train m transformation matrixs for m negation words. Such negator-specific transformation is in accordance with the finding that each negator has its individual negation effect (Zhu et al., 2014).

4.4 Intensity Regularizer (IR)

Sentiment intensity of a phrase indicates the strength of associated sentiment, which is quite important for fine-grained sentiment classification or rating. Intensifier can change the valence degree of the content word. The intensity regularizer models how intensity words influence the sentiment valence of a phrase or a sentence.

The formulation of the intensity effect is quite the same as that in the negation regularizer, but with different parameters of course. For each intensity word, there is a transform matrix to favor the different roles of various intensifiers on sentiment drift. For brevity, we will not repeat the formulas here.

4.5 Applying Linguistic Regularizers to Bidirectional LSTM

To preserve the simplicity of our proposals, we do not consider the modification span of negation and intensity words, which is a quite challenging problem in the NLP community (Zou et al., 2013; Packard et al., 2014; Fancellu et al., 2016). However, we can alleviate the problem by leveraging bidirectional LSTM.

For a single LSTM, we employ a backward LSTM from the end to the beginning of a sentence. This is because, at most times, the modified words of negation and intensity words are usually at the right side of the modified text. But sometimes, the modified words are at the left side of negation and intensity words. To better address this issue, we employ bidirectional LSTM and let the model determine which side should be chosen.

More formally, in Bi-LSTM, we compute a transformed sentiment distribution on $\overrightarrow{p}_{t-1}$ of the forward LSTM and also that on $\overleftarrow{p}_{t+1}$ of the backward LSTM, and compute the minimum distance of the distribution of the current position to the two distributions. This could be formulated as follows:

$$\overrightarrow{p}_{t-1}^{(R)} = softmax(T_{x_j} \times \overrightarrow{p}_{t-1}) \tag{12}$$

$$\overleftarrow{p}_{t+1}^{(R)} = softmax(T_{x_j} \times \overleftarrow{p}_{t+1}) \tag{13}$$

$$L_t^{(R)} = min \begin{cases} max(0, D_{KL}(\overrightarrow{p}_t || \overrightarrow{p}_{t-1}^{(R)}) - M) \\ max(0, D_{KL}(\overleftarrow{p}_t || \overleftarrow{p}_{t+1}^{(R)}) - M) \end{cases} \tag{14}$$

where $\overrightarrow{p}_{t-1}^{(R)}$ and $\overleftarrow{p}_{t+1}^{(R)}$ are the sentiment distributions transformed from the previous distribution $\overrightarrow{p}_{t-1}$ and next distribution $\overleftarrow{p}_{t+1}$ respectively. Note that $R \in \{NR, IR\}$ indicating the formulation works for both negation and intensity regularizers.

Due to the same consideration, we redefine $L_t^{(NSR)}$ and $L_t^{(SR)}$ with bidirectional LSTM similarly. The formulation is the same and omitted for brevity.

4.6 Discussion

Our models address these linguistic factors with mathematical operations, parameterized with shifting distribution vectors or transformation matrices. In the sentiment regularizer, the sentiment shifting effect is parameterized with a *class-specific* distribution (but could also be word-specific if with more data). In the negation and intensity regularizers, the effect is parameterized with *word-specific* transformation matrices. This is to respect the fact that the mechanism of how negation and intensity words shift sentiment expression is quite complex and highly dependent on individual words. Negation/Intensity effect also depends on the syntax and semantics of the modified text, however, for simplicity we resort to sequence LSTM for encoding surrounding contexts in this paper. We partially address the modification scope issue by applying the minimization operator in Eq. 11 and Eq. 14, and the bidirectional LSTM.

5 Experiment

5.1 Dataset and Sentiment Lexicon

Two datasets are used for evaluating the proposed models: Movie Review (MR) (Pang and Lee, 2005) where each sentence is annotated with two classes as *negative, positive* and Stanford Sentiment Treebank (SST) (Socher et al., 2013) with five classes { *very negative, negative, neutral, positive, very positive*}. Note that SST has provided phrase-level annotation on all inner nodes, but we only use the sentence-level annotation since one of our goals is to avoid expensive phrase-level annotation.

The sentiment lexicon contains two parts. The first part comes from MPQA (Wilson et al., 2005), which contains $5,153$ sentiment words, each with polarity rating. The second part consists of the leaf nodes of the SST dataset (i.e., all sentiment words) and there are $6,886$ polar words except *neural* ones. We combine the two parts and ignore those words that have conflicting sentiment labels, and produce a lexicon of $9,750$ words with 4 sentiment labels. For negation and intensity words, we collect them manually since the number is small, some of which can be seen in Table 2.

Dataset	MR	SST
# sentences in total	10,662	11,885
#sen containing sentiment word	10,446	11,211
#sen containing negation word	1,644	1,832
#sen containing intensity word	2,687	2,472

Table 1: The data statistics.

5.2 The Details of Experiment Setting

In order to let others reproduce our results, we present all the details of our models. We adopt Glove vectors (Pennington et al., 2014) as the initial setting of word embeddings V. The shifting vector for each sentiment class (s_c), and the transformation matrices for negation and intensity (T_m) are initialized with a prior value. The other parameters for hidden layers ($W^{(*)}, U^{(*)}, S$) are initialized with $Uniform(0, 1/sqrt(d))$, where d is the dimension of hidden representation, and we set d=300. We adopt adaGrad to train the models, and the learning rate is 0.1. It's worth noting that, we adopt stochastic gradient descent to update the word embeddings (V), with a learning rate of 0.2 but without momentum.

The optimal setting for α and β is 0.5 and 0.0001 respectively. During training, we adopt the dropout operation before the softmax layer, with a probability of 0.5. Mini-batch is taken to train the models, each batch containing 25 samples. After training with 3,000 mini-batch (about 9 epochs on MR and 10 epochs on SST), we choose the results of the model that performs best on the validation dataset as the final performance.

Negation word	no, nothing, never, neither, not, seldom, scarcely, etc.
Intensity word	terribly, greatly, absolutely, too, very, completely, etc.

Table 2: Examples of negation and intensity words.

5.3 Overall Comparison

We include several baselines, as listed below:

RNN/RNTN: Recursive Neural Network over parsing trees, proposed by (Socher et al., 2011) and Recursive Tensor Neural Network (Socher et al., 2013) employs tensors to model correlations between different dimensions of child nodes' vectors.

LSTM/Bi-LSTM: Long Short-Term Memory

(Cho et al., 2014) and the bidirectional variant as introduced previously.

Tree-LSTM: Tree-Structured Long Short-Term Memory (Tai et al., 2015) introduces memory cells and gates into tree-structured neural network.

CNN: Convolutional Neural Network (Kalchbrenner et al., 2014) generates sentence representation by convolution and pooling operations.

CNN-Tensor: In (Lei et al., 2015), the convolution operation is replaced by tensor product and a dynamic programming is applied to enumerate all skippable trigrams in a sentence. Very strong results are reported.

DAN: Deep Average Network (DAN) (Iyyer et al., 2015) averages all word vectors in a sentence and connects an MLP layer to the output layer.

Neural Context-Sensitive Lexicon: NCSL (Teng et al., 2016) treats the sentiment score of a sentence as a weighted sum of prior scores of words in the sentence where the weights are learned by a neural network.

Method	MR	SST Phrase-level	SST Sent.-level
RNN	77.7*	44.8#	43.2*
RNTN	75.9#	45.7*	43.4#
LSTM	77.4#	46.4*	45.6#
Bi-LSTM	79.3#	49.1*	46.5#
Tree-LSTM	80.7#	51.0*	48.1#
CNN	81.5*	48.0*	46.9#
CNN-Tensor	-	51.2*	50.6*
DAN	-	-	47.7*
NCSL	82.9	51.1*	47.1#
LR-Bi-LSTM	82.1	50.6	48.6
LR-LSTM	81.5	50.2	48.2

Table 3: The accuracy on MR and SST. *Phrase-level* means the models use phrase-level annotation for training. And *Sent.-level* means the models only use sentence-level annotation. Results marked with * are re-printed from the references, while those with # are obtained either by our own implementation or with the same codes shared by the original authors.

Firstly, we evaluate our model on the MR dataset and the results are shown in Table 3. We have the following observations:

First, both LR-LSTM and LR-Bi-LSTM outperforms their counterparts (81.5% vs. 77.4% and 82.1% vs. 79.3%, resp.), demonstrating the effectiveness of the linguistic regularizers. **Second**, LR-LSTM and LR-Bi-LSTM perform slightly better than Tree-LSTM but Tree-LSTM leverages a constituency tree structure while our model is a simple sequence model. As future work, we will apply such regularizers to tree-structured models.

Last, on the MR dataset, our model is comparable to or slightly better than CNN.

For fine-grained sentiment classification, we evaluate our model on the SST dataset which has five sentiment classes { *very negative, negative, neutral, positive, very positive*} so that we can evaluate the sentiment shifting effect of intensity words. The results are shown in Table 3. We have the following observations:

First, linguistically regularized LSTM and Bi-LSTM are better than their counterparts. It's worth noting that LR-Bi-LSTM (trained with just sentence-level annotation) is even comparable to Bi-LSTM trained with phrase-level annotation. That means, LR-Bi-LSTM can avoid the heavy phrase-level annotation but still obtain comparable results.

Second, our models are comparable to Tree-LSTM but our models are not dependent on a parsing tree and more simple, and hence more efficient. Further, for Tree-LSTM, the model is heavily dependent on phrase-level annotation, otherwise the performance drops substantially (from 51% to 48.1%).

Last, on the SST dataset, our model is better than CNN, DAN, and NCSL. We conjecture that the strong performance of CNN-Tensor may be due to the tensor product operation, the enumeration of all skippable trigrams, and the concatenated representations of all pooling layers for final classification.

5.4 The Effect of Different Regularizers

In order to reveal the effect of each individual regularizer, we conduct ablation experiments. Each time, we remove a regularizer and observe how the performance varies. First of all, we conduct this experiment on the entire datasets, and then we experiment on sub-datasets that only contain negation words or intensity words.

The experiment results are shown in Table 4 where we can see that the non-sentiment regularizer (NSR) and sentiment regularizer (SR) play a key role[3], and the negation regularizer and in-

[3]Kindly note that almost all sentences contain sentiment

Method	MR	SST
LR-Bi-LSTM	82.1	48.6
LR-Bi-LSTM (-NSR)	80.8	46.9
LR-Bi-LSTM (-SR)	80.6	46.9
LR-Bi-LSTM (-NR)	81.2	47.6
LR-Bi-LSTM (-IR)	81.7	47.9
LR-LSTM	81.5	48.2
LR-LSTM (-NSR)	80.2	46.4
LR-LSTM (-SR)	80.2	46.6
LR-LSTM (-NR)	80.8	47.4
LR-LSTM (-IR)	81.2	47.4

Table 4: The accuracy for LR-Bi-LSTM and LR-LSTM with regularizer ablation. *NSR*, *SR*, *NR* and *IR* denotes *Non-sentiment Regularizer*, *Sentiment Regularizer*, *Negation Regularizer*, and *Intensity Regularizer* respectively.

tensity regularizer are effective but less important than NSR and SR. This may be due to the fact that only 14% of sentences contains negation words in the test datasets, and 23% contains intensity words, and thus we further evaluate the models on two subsets, as shown in Table 5.

The experiments on the subsets show that: 1) With linguistic regularizers, LR-Bi-LSTM outperforms Bi-LSTM remarkably on these subsets; 2) When the negation regularizer is removed from the model, the performance drops significantly on both MR and SST subsets; 3) Similar observations can be found regarding the intensity regularizer.

Method	Neg. Sub.		Int. Sub.	
	MR	SST	MR	SST
BiLSTM	72.0	39.8	83.2	48.8
LR-Bi-LSTM (-NR)	74.2	41.6	-	-
LR-Bi-LSTM (-IR)	-	-	85.2	50.0
LR-Bi-LSTM	78.5	44.4	87.1	53.2

Table 5: The accuracy on the negation sub-dataset (Neg. Sub.) that only contains negators, and intensity sub-dataset (Int. Sub.) that only contains intensifiers.

5.5 The Effect of the Negation Regularizer

To further reveal the linguistic role of negation words, we compare the predicted sentiment distributions of a phrase pair with and without a negation word. The experimental results performed on MR are shown in Fig. 2. Each dot denotes a phrase pair (for example, <*interesting, not interesting*>), where the x-axis denotes the positive score[4] of a phrase without negators (e.g., *interesting*), and the y-axis indicates the positive score for the phrase with negators (e.g., *not interesting*). The curves in the figures show this function: $[1-y, y] = softmax(T_{nw} * [1-x, x])$ where $[1-x, x]$ is a sentiment distribution on $[negative, positive]$, x is the positive score of the phrase without negators (x-axis) and y that of the phrase with negators (y-axis), and T_{nw} is the transformation matrix for the negation word nw (see Eq. 9). By looking into the

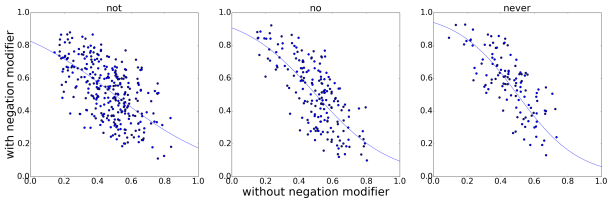


Figure 2: The sentiment shifts with negators. Each dot $< x, y >$ indicates that x is the sentiment score of a phrase without negator and y is that of the phrase with a negator.

detailed results of our model, we have the following statements:

First, there is no dot at the up-right and bottom-left blocks, indicating that negators generally shift/convert very positive or very negative phrases to other polarities. Typical phrases include *not very good, not too bad*.

Second, the dots at the up-left and bottom-right respectively indicates the negation effects: changing negative to positive and positive to negative. Typical phrases include *never seems hopelessly (up-left), no good scenes (bottom-right), not interesting (bottom-right)*, etc. There are also some positive/negative phrases shifting to neutral sentiment such as *not so good*, and *not too bad*.

Last, the dots located at the center indicate that neutral phrases maintain neutral sentiment with negators. Typical phrases include *not at home, not here*, where negators typically modify non-sentiment words.

5.6 The Effect of the Intensity Regularizer

To further reveal the linguistic role of intensity words, we perform experiments on the SST dataset, as illustrated in Figure 3. We show the

words, see Tab. 1.

[4] The score is obtained from the predicted distribution, where 1 means positive and 0 means negative.

matrix that indicates how the sentiment shifts after being modified by intensifiers. Each number in a cell (m_{ij}) indicates how many phrases are predicted with a sentiment label i but the prediction of the phrases with intensifiers changes to label j. For instance, the number 20 (m_{21}) in the second matrix , means that there are 20 phrases predicted with a class of *negative* (-) but the prediction changes to *very negative* (- -) after being modified by intensifier *"very"*. Results in the first

Figure 3: The sentiment shifting with intensifiers. The number in cell(m_{ij}) indicates how many phrases are predicted with sentiment label i but the prediction of phrases with intensifiers changes to label j.

matrix show that, for intensifier "*most*", there are 21/21/13/12 phrases whose sentiment is shifted after being modified by intensifiers, from negative to very negative (*eg. most irresponsible picture*), positive to very positive (*eg. most famous author*), neutral to negative (*eg. most plain*), and neutral to positive (*eg. most closely*), respectively.

There are also many phrases retaining the sentiment after being modified with intensifiers. Not surprisingly, for very positive/negative phrases, phrases modified by intensifiers still maintain the strong sentiment. For the left phrases, they fall into three categories: first, words modified by intensifiers are non-sentiment words, such as *most of us, most part*; second, intensifiers are not strong enough to shift sentiment, such as *most complex* (from neg. to neg.), *most traditional* (from *pos.* to *pos.*); third, our models fail to shift sentiment with intensifiers such as *most vital, most resonant film*.

6 Conclusion and Future Work

We present linguistically regularized LSTMs for sentence-level sentiment classification. The proposed models address the sentient shifting effect of sentiment, negation, and intensity words. Furthermore, our models are sequence LSTMs which do not depend on a parsing tree-structure and do not require expensive phrase-level annotation. Results show that our models are able to address the linguistic role of sentiment, negation, and intensity words.

To preserve the simplicity of the proposed models, we do not consider the modification scope of negation and intensity words, though we partially address this issue by applying a minimization operartor (see Eq. 11, Eq. 14) and bi-directional LSTM. As future work, we plan to apply the linguistic regularizers to tree-LSTM to address the scope issue since the parsing tree is easier to indicate the modification scope explicitly.

Acknowledgments

This work was partly supported by the National Basic Research Program (973 Program) under grant No. 2013CB329403, and the National Science Foundation of China under grant No.61272227/61332007.

References

Farah Benamara, Baptiste Chardon, Yannick Mathieu, Vladimir Popescu, and Nicholas Asher. 2012. How do negation and modality impact on opinions? In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*. pages 10–18.

Yanqing Chen and Steven Skiena. 2014. Building sentiment lexicons for all major languages. In *ACL*. pages 383–389.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. 2015. A statistical parsing framework for sentiment classification. *Computational Linguistics* .

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *AAAI*. AAAI.

Federico Fancellu, Adam Lopez, and Bonnie Webber. 2016. Neural networks for negation scope detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 495–504.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, pages 273–278.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 168–177.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*.

Lifeng Jia, Clement Yu, and Weiyi Meng. 2009. The effect of negation on sentiment analysis and retrieval effectiveness. In *Proceedings of the 18th ACM conference on Information and knowledge management*. pages 1827–1830.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*. pages 655–665.

Alistair Kennedy and Diana Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational intelligence* 22(2):110–125.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*. pages 1746–1751.

Emanuele Lapponi, Jonathon Read, and Lilja Øvrelid. 2012. Representing and resolving negation for sentiment analysis. In *2012 IEEE 12th International Conference on Data Mining Workshops*. pages 687–692.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. *ACL* .

Jingjing Liu and Stephanie Seneff. 2009. Review sentiment scoring via a parse-and-paraphrase paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. pages 161–169.

Nikolaos Malandrakis, Alexandros Potamianos, Elias Iosif, and Shrikanth Narayanan. 2013. Distributional semantic models for affective text analysis. *IEEE Transactions on Audio, Speech, and Language Processing* 21(11):2379–2392.

Tomáš Mikolov. 2012. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April* .

Woodley Packard, M. Emily Bender, Jonathon Read, Stephan Oepen, and Rebecca Dridan. 2014. Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 69–78.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*. pages 115–124.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval* 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *ACL*. pages 79–86.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *EMNLP* 12:1532–1543.

Livia Polanyi and Annie Zaenen. 2006. Contextual valence shifters. In *Computing attitude and affect in text: Theory and applications*, Springer, pages 1–10.

Qiao Qian, Bo Tian, Minlie Huang, Yang Liu, Xuan Zhu, and Xiaoyan Zhu. 2015. Learning tag embeddings and tag-specific composition functions in recursive neural network. In *ACL*. volume 1, pages 1365–1374.

Raksha Sharma, Mohit Gupta, Astha Agarwal, and Pushpak Bhattacharyya. 2015. Adjective intensity and sentiment analysis. *EMNLP2015* .

Chaitanya Shivade, Marie-Catherine de Marneffe, Eric Folser-Lussier, and Albert Lai. 2015. Corpus-based discovery of semantic intensity scales. *In Proceedings of NAACL-HTL* .

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*. pages 151–161.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*. pages 1631–1642.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics* 37(2):267–307.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Zhiyang Teng, Duy-Tin Vo, and Yue Zhang. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1629–1638.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *ACL*. pages 417–424.

Duy Tin Vo and Yue Zhang. 2016. Dont count, predict! an automatic approach to learning sentiment lexicons for short text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. volume 2, pages 219–224.

Feixiang Wang, Zhihua Zhang, and Man Lan. 2016. Ecnu at semeval-2016 task 7: An enhanced supervised learning method for lexicon sentiment intensity ranking. *Proceedings of SemEval* pages 491–496.

Wen-Li Wei, Chung-Hsien Wu, and Jen-Chun Lin. 2011. A regression approach to affective rating of chinese words from anew. In *Affective Computing and Intelligent Interaction*, Springer, pages 121–131.

Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. 2010. A survey on the role of negation in sentiment analysis. In *Proceedings of the workshop on negation and speculation in natural language processing*. Association for Computational Linguistics, pages 60–68.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *EMNLP*. pages 347–354.

Dani Yogatama and Noah A. Smith. 2014. Linguistic structured sparsity in text categorization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 786–796.

Xiaodan Zhu, Hongyu Guo, Saif Mohammad, and Svetlana Kiritchenko. 2014. An empirical study on the effect of negation words on sentiment. In *ACL*. pages 304–313.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *ICML*. pages 1604–1612.

Bowei Zou, Guodong Zhou, and Qiaoming Zhu. 2013. Tree kernel-based negation and speculation scope detection with structured syntactic parse features. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 968–976.

Sarcasm SIGN: Interpreting Sarcasm with Sentiment Based Monolingual Machine Translation

Lotem Peled and **Roi Reichart**
Faculty of Industrial Engineering and Management, Technion, IIT
splotem@campus.technion.ac.il, roiri@ie.technion.ac.il

Abstract

Sarcasm is a form of speech in which speakers say the opposite of what they truly mean in order to convey a strong sentiment. In other words, *"Sarcasm is the giant chasm between what I say, and the person who doesn't get it."*. In this paper we present the novel task of sarcasm interpretation, defined as the generation of a non-sarcastic utterance conveying the same message as the original sarcastic one. We introduce a novel dataset of 3000 sarcastic tweets, each interpreted by five human judges. Addressing the task as monolingual machine translation (MT), we experiment with MT algorithms and evaluation measures. We then present SIGN: an MT based sarcasm interpretation algorithm that targets sentiment words, a defining element of textual sarcasm. We show that while the scores of n-gram based automatic measures are similar for all interpretation models, SIGN's interpretations are scored higher by humans for adequacy and sentiment polarity. We conclude with a discussion on future research directions for our new task.[1]

1 Introduction

Sarcasm is a sophisticated form of communication in which speakers convey their message in an indirect way. It is defined in the Merriam-Webster dictionary (Merriam-Webster, 1983) as the use of words that mean the opposite of what one would really want to say in order to insult someone, to show irritation, or to be funny. Considering this definition, it is not surprising to find frequent use of sarcastic language in opinionated user generated content, in environments such as Twitter, Facebook, Reddit and many more.

In textual communication, knowledge about the speaker's intent is necessary in order to fully understand and interpret sarcasm. Consider, for example, the sentence *"what a wonderful day"*. A literal analysis of this sentence demonstrates a positive experience, due to the use of the word *wonderful*. However, if we knew that the sentence was meant sarcastically, *wonderful* would turn into a word of a strong negative sentiment. In spoken language, sarcastic utterances are often accompanied by a certain tone of voice which points out the intent of the speaker, whereas in textual communication, sarcasm is inherently ambiguous, and its identification and interpretation may be challenging even for humans.

In this paper we present the novel task of interpretation of sarcastic utterances. We define the purpose of the interpretation task as the capability to generate a non-sarcastic utterance that captures the meaning behind the original sarcastic text.

Our work currently targets the Twitter domain since it is a medium in which sarcasm is prevalent, and it allows us to focus on the interpretation of tweets marked with the content tag #sarcasm. And so, for example, given the tweet *"how I love Mondays. #sarcasm"* we would like our system to generate interpretations such as *"how I hate Mondays"* or *"I really hate Mondays"*. In order to learn such interpretations, we constructed a parallel corpus of 3000 sarcastic tweets, each of which has five non-sarcastic interpretations (Section 3).

Our task is complex since sarcasm can be expressed in many forms, it is ambiguous in nature and its understanding may require world knowl-

[1] Our dataset, consisting of 3000 sarcastic tweets each augmented with five interpretations, is available in the project page: https://github.com/Lotemp/SarcasmSIGN. The page also contains the sarcasm interpretation guidelines, the code of the SIGN algorithms and other materials related to this project.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1690–1700
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1155

edge. Following are several examples taken from our corpus:

1. *loving life so much right now. #sarcasm*
2. *Way to go California! #sarcasm*
3. *Great, a choice between two excellent candidates, Donald Trump or Hillary Clinton. #sarcasm*

In example (1) it is quite straightforward to see the exaggerated positive sentiment used in order to convey strong negative feelings. Examples (2) and (3), however, do not contain any excessive sentiment. Instead, previous knowledge is required if one wishes to fully understand and interpret what went wrong with California, or who Hillary Clinton and Donald Trump are.

Since sarcasm is a refined and indirect form of speech, its interpretation may be challenging for certain populations. For example, studies show that children with deafness, autism or Asperger's Syndrome struggle with non literal communication such as sarcastic language (Peterson et al., 2012; Kimhi, 2014). Moreover, since sarcasm transforms the polarity of an apparently positive or negative expression into its opposite, it poses a challenge for automatic systems for opinion mining, sentiment analysis and extractive summarization (Popescu et al., 2005; Pang and Lee, 2008; Wiebe et al., 2004). Extracting the honest meaning behind the sarcasm may alleviate such issues.

In order to design an automatic sarcasm interpretation system, we first rely on previous work in established similar tasks (section 2), particularly machine translation (MT), borrowing algorithms as well as evaluation measures. In section 4 we discuss the automatic evaluation measures we apply in our work and present human based measures for: (a) the *fluency* of a generated non-sarcastic utterance, (b) its *adequacy* as interpretation of the original sarcastic tweet's meaning, and (c) whether or not it captures the sentiment of the original tweet. Then, in section 5, we explore the performance of prominent phrase-based and neural MT systems on our task in development data experiments. We next present the *Sarcasm SIGN* (***S**arcasm **S**entimental **I**nterpretation **GeN**erator*, section 6), our novel MT based algorithm which puts a special emphasis on sentiment words. Lastly, in Section 7 we assess the performance of the various algorithms and show that while they perform similarly in terms of automatic MT evaluation, SIGN is superior according to the human measures. We conclude with a discussion on future research directions for our task, regarding both algorithms and evaluation.

2 Related Work

The use of irony and sarcasm has been well studied in the linguistics (Muecke, 1982; Stingfellow, 1994; Gibbs and Colston, 2007) and the psychology (Shamay-Tsoory et al., 2005; Peterson et al., 2012) literature. In computational work, the interest in sarcasm has dramatically increased over the past few years. This is probably due to factors such as the rapid growth in user generated content on the web, in which sarcasm is used excessively (Maynard et al., 2012; Kaplan and Haenlein, 2011; Bamman and Smith, 2015; Wang, 2013) and the challenge that sarcasm poses for opinion mining and sentiment analysis systems (Pang and Lee, 2008; Maynard and Greenwood, 2014). Despite this rising interest, and despite many works that deal with sarcasm identification (Tsur et al., 2010; Davidov et al., 2010; González-Ibánez et al., 2011; Riloff et al., 2013; Barbieri et al., 2014), to the best of our knowledge, *generation of sarcasm interpretations* has not been previously attempted.

Therefore, the following sections are dedicated to previous work from neighboring NLP fields which are relevant to our work: sarcasm detection, MT, paraphrasing and text summarization.

Sarcasm Detection Recent computational work on sarcasm revolves mainly around detection. Due to the large volume of detection work, we survey only several representative examples.

Tsur et al. (2010) and Davidov et al. (2010) presented a semi-supervised approach for detecting irony and sarcasm in product-reviews and tweets, where features are based on ironic speech patterns extracted from a labeled dataset. González-Ibánez et al. (2011) used lexical and pragmatic features, e.g. emojis and whether the utterance is a comment to another person, in order to train a classifier that distinguishes sarcastic utterances from tweets of positive and negative sentiment.

Riloff et al. (2013) observed that a certain type of sarcasm is characterized by a contrast between a positive sentiment and a negative situation. Consequently, they described a bootstrapping algorithm that learns distinctive phrases connected to negative situations along with a positive sentiment and used these phrases to train their classifier. Barbieri et al. (2014) avoided using word patterns and

instead employed features such as the length and sentiment of the tweet, and the use of rare words.

Despite the differences between detection and interpretation, this line of work is highly relevant to ours in terms of feature design. Moreover, it presents fundamental notions, such as the sentiment polarity of the sarcastic utterance and of its interpretation, that we adopt. Finally, when utterances are not marked for sarcasm as in the Twitter domain, or when these labels are not reliable, detection is a necessary step before interpretation.

Machine Translation We approach our task as one of monolingual MT, where we translate sarcastic English into non-sarcastic English. Therefore, our starting point is the application of MT techniques and evaluation measures. The three major approaches to MT are phrase based (Koehn et al., 2007), syntax based (Koehn et al., 2003) and the recent neural approach. For automatic MT evaluation, often an n-gram co-occurrence based scoring is performed in order to measure the lexical closeness between a candidate and a reference translations. Example measures are NIST (Doddington, 2002), METEOR (Denkowski and Lavie, 2011), and the widely used BLEU (Papineni et al., 2002), which represents *precision*: the fraction of n-grams from the machine generated translation that also appear in the human reference.

Here we employ the phrase based Moses system (Koehn et al., 2007) and an RNN-encoder-decoder architecture, based on Cho et al. (2014). Later we will show that these algorithms can be further improved and will explore the quality of the MT evaluation measures in the context of our task.

Paraphrasing and Summarization Tasks such as paraphrasing and summarization are often addressed as monolingual MT, and so they are close in nature to our task. Quirk et al. (2004) proposed a model of paraphrasing based on monolingual MT, and utilized alignment models used in the Moses translation system (Koehn et al., 2007; Wubben et al., 2010; Bannard and Callison-Burch, 2005). Xu et al. (2015) presented the task of paraphrase generation while targeting a particular writing style, specifically paraphrasing modern English into Shakespearean English, and approached it with phrase based MT.

Work on paraphrasing and summarization is often evaluated using MT evaluation measures such as BLEU. As BLEU is *precision-oriented*, complementary *recall-oriented* measures are often used as well. A prominent example is ROUGE (Lin, 2004), a family of measures used mostly for evaluation in automatic summarization: candidate summaries are scored according to the fraction of n-grams from the human references they contain.

We also utilize PINC (Chen and Dolan, 2011), a measure which rewards paraphrases for being different from their source, by introducing new n-grams. PINC is often combined with BLEU due to their complementary nature: while PINC rewards n-gram novelty, BLEU rewards similarity to the reference. The highest correlation with human judgments is achieved by the product of PINC with a sigmoid function of BLEU (Chen and Dolan, 2011).

3 A Parallel Sarcastic Tweets Corpus

To properly investigate our task, we collected a dataset, first of its kind, of sarcastic tweets and their non-sarcastic (honest) interpretations. This data, as well as the instructions provided for our human judges, will be made publicly available and will hopefully provide a basis for future work regarding sarcasm on Twitter. Despite the focus of the current work on the Twitter domain, we consider our task as a more general one, and hope that our discussion, observations and algorithms will be beneficial for other domains as well.

Using the Twitter API[2], we collected tweets marked with the content tag #sarcasm, posted between Januray and June of 2016. Following Tsur et al. (2010), González-Ibánez et al. (2011) and Bamman and Smith (2015), we address the problem of noisy tweets with automatic filtering: we remove all tweets not written in English, discard retweets (tweets that have been forwarded or shared) and remove tweets containing URLs or images, so that the sarcasm in the tweet regards to the text only and not to an image or a link. This results in 3000 sarcastic tweets containing text only, where the average sarcastic tweet length is 13.87 utterances, the average interpretation length is 12.10 words and the vocabulary size is 8788 unique words.

In order to obtain honest interpretations for our sarcastic tweets, we used Fiverr[3] – a platform for selling and purchasing services from independent suppliers (also referred to as workers). We em-

[2] http://apiwiki.twitter.com
[3] https://www.fiverr.com

Sarcastic Tweets	Honest Interpretations
What a great way to end my night. #sarcasm	1. Such a bad ending to my night 2. Oh what a great way to ruin my night 3. What a horrible way to end a night 4. Not a good way to end the night 5. Well that wasn't the night I was hoping for
Staying up till 2:30 was a brilliant idea, very productive #sarcasm	1. Bad idea staying up late, not very productive 2. It was not smart or productive for me to stay up so late 3. Staying up till 2:30 was not a brilliant idea, very non-productive 4. I need to go to bed on time 5. Staying up till 2:30 was completely useless

Table 1: Examples from our parallel sarcastic tweet corpus.

ployed ten Fiverr workers, half of them from the field of comedy writing, and half from the field of literature paraphrasing. The chosen workers were made sure to have an active Twitter account, in order to ensure their acquaintance with social networks and with Twitter's colorful language (hashtags, common acronyms such as LOL, etc.).

We then randomly divided our tweet corpus to two batches of size 1500 each, and randomly assigned five workers to each batch. We instructed the workers to translate each sarcastic tweet into a non sarcastic utterance, while maintaining the original meaning. We encouraged the workers to use external knowledge sources (such as Google) if they came across a subject they were not familiar with, or if the sarcasm was unclear to them.

Although our dataset consists only of tweets that were marked with the hashtag *#sarcasm*, some of these tweets were not identified as sarcastic by all or some of our Fiverr workers. In such cases the workers were instructed to keep the original tweet unchanged (i.e, uninterpreted). We keep such tweets in our dataset since we expect a sarcasm interpretation system to be able to recognize non-sarcastic utterances in its input, and to leave them in their original form.

Table 1 presents two examples from our corpus. The table demonstrates the tendency of the workers to generally agree on the core meaning of the sarcastic tweets. Yet, since sarcasm is inherently vague, it is not surprising that the interpretations differ from one worker to another. For example, some workers change only one or two words from the original sarcastic tweet, while others rephrase the entire utterance. We regard this as beneficial, since it brings a natural, human variance into the task. This variance makes the evaluation of automatic sarcasm interpretation algorithms challenging, as we further discuss in the next section.

4 Evaluation Measures

As mentioned above, in certain cases world knowledge is mandatory in order to correctly evaluate sarcasm interpretations. For example, in the case of the second sarcastic tweet in table 1, we need to know that 2:30 is considered a late hour so that *staying up till 2:30* and *staying up late* would be considered equivalent despite the lexical difference. Furthermore, we notice that transforming a sarcastic utterance into a non sarcastic one often requires to change a small number of words. For example, a single word change in the sarcastic tweet *"How I love Mondays. #sarcasm"* leads to the non-sarcastic utterance *How I hate Mondays*.

This is not typical for MT, where usually the entire source sentence is translated to a new sentence in the target language and we would expect lexical similarity between the machine generated translation and the human reference it is compared to. This raises a doubt as to whether n-gram based MT evaluation measures such as the aforementioned are suitable for our task. We hence asses the quality of an interpretation using *automatic evaluation measures* from the tasks of MT, paraphrasing, and summarization (Section 2), and compare these measures to *human-based measures*.

Automatic Measures We use BLEU and ROUGE as measures of n-gram precision and recall, respectively. We report scores of ROUGE-1, ROUGE-2 and ROUGE-L (recall based on unigrams, bigrams and longest common subsequence between candidate and reference, respectively). In order to asses the n-gram novelty of interpretations (i.e, difference from the source), we report PINC and PINC$*$sigmoid(BLEU) (see Section 2).

Human judgments We employed an additional group of five Fiverr workers and asked them to score each generated interpretations with two scores on a 1-7 scale, 7 being the best. The scores

Sarcastic Tweet	Moses Interpretation	Neural Interpretation
Boy , am I glad the rain's here #sarcasm	Boy, I'm so annoyed that the rain is here	I'm not glad to go today
Another night of work, Oh, the joy #sarcasm	Another night of work, Ugh, unbearable	Another night, I don't like it
Being stuck in an airport is fun #sarcasm	Be stuck in an airport is not fun	Yay, stuck at the office again
You're the best. #sarcasm	You're the best	You're my best friend

Table 2: Sarcasm interpretations generated by Moses and by the RNN.

	Evaluation Measure	Moses	RNN
Precision Oriented	BLEU	62.91	41.05
Novelty Oriented	PINC	51.81	76.45
	PINC∗sigmoid(BLEU)	33.79	45.96
Recall Oriented	ROUGE-1	66.44	42.20
	ROUGE-2	41.03	29.97
	ROUGE-l	65.31	40.87
Human Judgments	Fluency	6.46	5.12
	Adequacy	2.54	2.08
	% correct sentiment	28.84	17.93

Table 3: Development data results for MT models.

are: *adequacy*: the degree to which the interpretation captures the meaning of the original tweet; and *fluency*: how readable the interpretation is. In addition, reasoning that a high quality interpretation is one that captures the true intent of the sarcastic utterance by using words suitable to its sentiment, we ask the workers to assign the interpretation with a binary score indicating whether the sentiment presented in the interpretation agrees with the sentiment of the original sarcastic tweet.[4]

The human measures enjoy high agreement levels between the human judges. The averaged root mean squared error calculated on the test set across all pairs of judges and across the various algorithms we experiment with are: 1.44 for fluency and 1.15 for adequacy. For sentiment scores the averaged agreement at the same setup is 93.2%.

5 Sarcasm Interpretations as MT

As our task is about the generation of one English sentence given another, a natural starting point is treating it as monolingual MT. We hence begin with utilizing two widely used MT systems, representing two different approaches: Phrase Based MT vs. Neural MT. We then analyze the performance of these two systems, and based on our conclusions we design our SIGN model.

Phrase Based MT We employ Moses[5], using word alignments extracted by GIZA++ (Och and Ney, 2003) and symmetrized with the grow-diag-final strategy. We use phrases of up to 8 words to build our phrase table, and do not filter sentences according to length since tweets contain at most 140 characters. We employ the KenLM algorithm (Heafield, 2011) for language modeling, and train it on the non-sarcastic tweet interpretations (the target side of the parallel corpus).

Neural Machine Translation We use GroundHog, a publicly available implementation of an RNN encoder-decoder, with LSTM hidden states.[6] Our encoder and decoder contain 250 hidden units each. We use the minibatch stochastic gradient descent (SGD) algorithm together with Adadelta (Zeiler, 2012) to train each model, where each SGD update is computed using a minibatch of 16 utterances. Following Sutskever et al. (2014), we use beam search for test time decoding. Henceforth we refer to this system as *RNN*.

Performance Analysis We divide our corpus into training, development and test sets of sizes 2400, 300 and 300 respectively. We train Moses and the RNN on the training set and tune their parameters on the development set. Table 3 presents *development data* results, as these are preliminary experiments that aim to asses the compatibility of MT algorithms to our task.

Moses scores much higher in terms of BLEU and ROUGE, meaning that compared to the RNN its interpretations capture more n-grams appearing in the human references while maintaining high precision. The RNN outscores Moses in terms of PINC and PINC∗sigmoid(BLEU), meaning that its interpretations are more novel, in terms of n-grams. This alone might not be a negative trait; However, according to human judgments Moses performs better in terms of fluency, adequacy and sentiment, and so the novelty of the RNN's interpretations does not necessarily contribute to their

[4]For example, we consider *"Best day ever #sarcasm"* and its interpretation "Worst day ever" to agree on the sentiment, despite the use of opposite sentiment words.

[5]http://www.statmt.org/moses

[6]https://github.com/lisa-groundhog/GroundHog

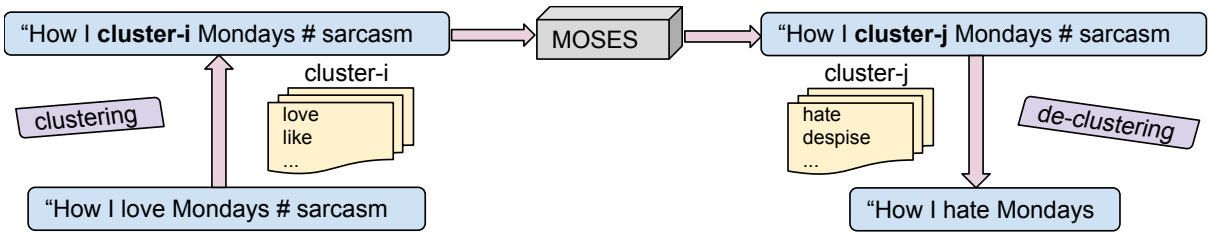


Figure 1: An illustration of the application of SIGN to the tweet *"How I love Mondays # sarcasm"*.

quality, and even possibly reduces it.

Table 2 illustrates several examples of the interpretations generated by both Moses and the RNN. While the interpretations generated by the RNN are readable, they generally do not maintain the meaning of the original tweet. We believe that this is the result of the neural network overfitting the training set, despite regularization and dropout layers, probably due to the relatively small training set size. In light of these results when we experiment with the SIGN algorithm (Section 7), we employ Moses as its MT component.

The final example of Table 2 is representative of cases where both Moses and the RNN fail to capture the sarcastic sense of the tweet, incorrectly interpreting it or leaving it unchanged. In order to deal with such cases, we wish to utilize a property typical of sarcastic language. Sarcasm is mostly used to convey a certain emotion by using strong sentiment words that express the exact opposite of their literal meaning. Hence, many sarcastic utterances can be correctly interpreted by keeping most of their words, replacing only sentiment words with expressions of the opposite sentiment. For example, the sarcasm in the utterance *"You're the best. #sarcasm"* is hidden in *best*, a word of a strong positive sentiment. If we transform this word into a word of the opposite sentiment, such as *worst*, then we get a non-sarcastic utterance with the correct sentiment.

We next present the *Sarcasm SIGN* (***S**arcasm **S**entimental **I**nterpretation **G**e**N**erator*), an algorithm which capitalizes on sentiment words in order to produce accurate interpretations.

6 The Sarcasm SIGN Algorithm

SIGN (Figure 1) targets sentiment words in sarcastic utterances. First, it clusters sentiment words according to semantic relatedness. Then, each sentiment word is replaced with its cluster [7] and the transformed data is fed into an MT system (Moses in this work), at both its training and test phases. Consequently, at test time the MT system outputs non-sarcastic utterances with clusters replacing sentiment words. Finally, SIGN performs a de-clustering process on these MT outputs, replacing sentiment clusters with suitable words.

Positive Clusters	merit, wonder, props, praise, congratulations..	patience, dignity, truth, chivalry, rationality...
Negative Clusters	hideous, horrible, nasty, obnoxious, scary, pathetic...	shame, sadness, sorrow, fear, disappointment, regret, danger...

Table 4: Examples of two positive and two negative clusters created by the SIGN algorithm.

In order to detect the sentiment of words, we turn to SentiWordNet (Esuli and Sebastiani, 2006), a lexical resource based on WordNet (Miller et al., 1990). Using SentiWordNet's positivity and negativity scores, we collect from our training data a set of distinctly positive words ($\sim$ 70) and a set of distinctly negative words ($\sim$ 160).[8] We then utilize the pre-trained dependency-based word embeddings of Levy and Goldberg (2014)[9] and cluster each set using the k-means algorithm with $L2$ distance. We aim to have ten words on average in each cluster, and so the positive set is clustered into 7 clusters, and the negative set into 16 clusters. Table 4 presents examples from our clusters.

Upon receiving a sarcastic tweet, at both training and test, SIGN searches it for sentiment words according to the positive and negative sets. If such

[7] This means that we replace a *word* with *cluster-j* where *j* is the number of the cluster to which the *word* belongs.

[8] The scores are in the [0,1] range. We set the threshold of 0.6 for both distinctly positive and distinctly negative words.

[9] https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/. We choose these embeddings since they are believed to better capture the relations between a word and its context, having been trained on dependency-parsed sentences.

	Evaluation Measure	Moses	SIGN-centroid	SIGN-context	SIGN-oracle
Precision Oriented	BLEU	65.24	63.52	**66.96**	67.49
Novelty Oriented	PINC	45.92	**47.11**	46.65	46.10
	PINC∗sigmoid(BLEU)	30.21	30.79	**31.13**	30.54
Recall Oriented	ROUGE-1	**70.26**	68.43	69.67	70.34
	ROUGE-2	**42.18**	40.34	40.96	42.81
	ROUGE-l	69.82	68.24	**69.98**	70.01

Table 5: Test data results with automatic evaluation measures.

a word is found, it is replaced with its cluster. For example, given the sentence *"How I love Mondays. #sarcasm"*, *love* will be recognized as a positive sentiment word, and the sarcastic tweet will become: *"How I **cluster-i** Mondays. #sarcasm"* where i is the cluster number of the word *love*.

During training, this process is also applied to the non-sarcastic references. And so, if one such reference is *"I dislike Mondays."*, then *dislike* will be identified and the reference will become *"I **cluster-j** Mondays."*, where j is the cluster number of the word *dislike*. Moses is then trained on these new representations of the corpus, using the exact same setup as before. This training process produces a mapping between positive and negative clusters, and outputs sarcastic interpretations with clustered sentiment words (e.g, *"I **cluster-j** Mondays."*). At test time, after Moses generates an utterance containing clusters, a de-clustering process takes place: the clusters are replaced with the appropriate sentiment words.

We experiment with several de-clustering approaches: **(1) SIGN-centroid:** the chosen sentiment word will be the one closest to the centroid of cluster j. For example in the tweet *"I **cluster-j** Mondays."*, the sentiment word closest to the centroid of cluster j will be chosen; **(2) SIGN-context:** the cluster is replaced with its word that has the highest average Pointwise Mutual Information (PMI) with the words in a symmetric context window of size 3 around the cluster's location in the output. For example, for *"I **cluster-j** Mondays."*, the sentiment word from cluster j which has the highest average PMI with the words in {'I','Mondays'} will be chosen. The PMI values are computed on the training data; and **(3) SIGN-Oracle:** an upper bound where a person manually chooses the most suitable word from the cluster.

We expect this process to improve the quality of sarcasm interpretations in two aspects. First, as mentioned earlier, sarcastic tweets often differ from their non sarcastic interpretations in a small number of sentiment words (sometimes even in a single word). SIGN should help highlight the sentiment words most in need of interpretation. Second, under the pre-processing SIGN performs to the input examples of Moses, the latter is inclined to learn a mapping from positive to negative clusters, and vice versa. This is likely to encourage the Moses output to generate outputs of the same sentiment as the original sarcastic tweet, but with honest sentiment words. For example, if the sarcastic tweet expresses a negative sentiment with strong positive words, the non-sarcastic interpretation will express this negative sentiment with negative words, thus stripping away the sarcasm.

	Fluency	Adequacy	% correct sentiment	% changed
Moses	**6.67**	2.55	25.7	42.3
SIGN-Centroid	6.38	3.23*	42.2*	67.4
SIGN-Context	6.66	**3.61***	**46.2***	**68.5**
SIGN-Oracle	6.69	3.67*	46.8*	68.8

Table 6: Test set results with human measures. *%changed* provides the fraction of tweets that were changed during interpretation (i.e. the tweet and its interpretation are not identical). In cases where one of our models presents significant improvement over Moses, the results are decorated with a star. Statistical significance is tested with the paired t-test for fluency and adequacy, and with the McNemar paired test for labeling disagreements (Gillick and Cox, 1989) for *% correct sentiment*, in both cases with $p < 0.05$.

7 Experiments and Results

We experiment with SIGN and the Moses and RNN baselines at the same setup of section 5. We report test set results for automatic and human measures, in Tables 5 and 6 respectively. As in the development data experiments (Table 3), the RNN presents critically low adequacy scores of 2.11 across the entire test set and of 1.89 in cases where the interpretation and the tweet differ. This, along with its low fluency scores (5.74 and 5.43

respectively) and its very low BLEU and ROUGE scores make us deem this model immature for our task and dataset, hence we exclude it from this section's tables and do not discuss it further.

In terms of automatic evaluation (Table 5), SIGN and Moses do not perform significantly different. When it comes to human evaluation (Table 6) however, SIGN-context presents substantial gains. While for fluency Moses and SIGN-context perform similarly, SIGN-context performs much better in terms of adequacy and the percentage of tweets with the correct sentiment. The differences are substantial as well as statistically significant: adequacy of 3.61 for SIGN-context compared to 2.55 of Moses, and correct sentiment for 46.2% of the SIGN interpretations, compared to only 25.7% of the Moses interpretations.

Table 6 further provides an initial explanation to the improvement of SIGN over Moses: Moses tends to keep interpretations identical to the original sarcastic tweet, altering them in only 42.3% of the cases, [10] while SIGN-context's interpretations differ from the original sarcastic tweet in 68.5% of the cases, which comes closer to the 73.8% in the gold standard human interpretations. If for each of the algorithms we only regard to interpretations that differ from the original sarcastic tweet, the differences between the models are less substantial. Nonetheless, SIGN-context still presents improvement by correctly changing sentiment in 67.5% of the cases compared to 60.8% for Moses.

Both tables consistently show that the context-based selection strategy of SIGN outperforms the centroid alternative. This makes sense as, being context-ignorant, SIGN-centroid might produce non-fluent or inadequate interpretations for a given context. For example, the tweet *"Also gotta move a piano as well. joy #sarcasm"* is changed to *"Also gotta move a piano as well. bummer"* by SIGN-context, while SIGN-centroid changes it to the less appropriate *"Also gotta move a piano as well. boring"*. Nonetheless, even this naive de-clustering approach substantially improves adequacy and sentiment accuracy over Moses.

Finally, comparison to SIGN-oracle reveals that the context selection strategy is not far from human performance with respect to both automatic and human evaluation measures. Still, some gain can be achieved, especially for the human measures on tweets that were changed at interpretation. This indicates that SIGN can improve mostly through a better clustering of sentiment words, rather than through a better selection strategy.

[10] We elaborate on this in section 8.

8 Discussion and Future Work

Automatic vs. Human Measures The performance gap between Moses and SIGN may stem from the difference in their optimization criteria. Moses aims to optimize the BLEU score and given the overall lexical similarity between the original tweets and their interpretations, it therefore tends to keep them identical. SIGN, in contrast, targets sentiment words and changes them frequently. Consequently, we do not observe substantial differences between the algorithms in the automatic measures that are mostly based on n-gram differences between the source and the interpretation. Likewise, the human fluency measure that accounts for the readability of the interpretation is not seriously affected by the translation process. When it comes to the human adequacy and sentiment measures, which account for the understanding of the tweet's meaning, SIGN reveals its power and demonstrates much better performance compared to Moses.

To further understand the relationship between the automatic and the human based measures we computed the Pearson correlations for each pair of (automatic, human) measures. We observe that all correlation values are low (up to 0.12 for fluency, 0.13-0.18 for sentiment and 0.19-0.24 for adequacy). Moreover, for fluency the correlation values are insignificant (using a correlation significance t-test with $p = 0.05$). We believe this indicates that these automatic measures do not provide appropriate evaluation for our task. Designing automatic measures is hence left for future research.

Sarcasm Interpretation as Sentiment Based Monolingual MT: Strengths and Weaknesses The SIGN models' strength is revealed when interpreting sarcastic tweets with strong sentiment words, transforming expressions such as *"Audits are a blast to do #sarcasm"* and *"Being stuck in an airport is fun #sarcasm"* into *"Audits are a bummer to do"* and *"Being stuck in an airport is boring"*, respectively. Even when there are no words of strong sentiment, the MT component of SIGN still performs well, interpreting tweets such as *"the Cavs aren't getting any calls, this is new #sarcasm"* into *"the Cavs aren't getting any calls, as usuall"*.

The SIGN models perform well even in cases where there are several sentiment words but not all of them require change. For example, for the sarcastic tweet *"Constantly being irritated, anxious and depressed is a great feeling! #sarcasm"*, SIGN-context produces the adequate interpretation: *"Constantly being irritated, anxious and depressed is a terrible feeling"*.

Future research directions rise from cases in which the SIGN models left the tweet unchanged. One prominent set of examples consists of tweets that require world knowledge for correct interpretation. Consider the tweet *"Can you imagine if Lebron had help? #sarcasm"*. The model requires knowledge of who Lebron is and what kind of help he needs in order to fully understand and interpret the sarcasm. In practice the SIGN models leave this tweet untouched.

Another set of examples consists of tweets that lack an explicit sentiment word, for example, the tweet *"Clear example they made of Sharapova then, ey? #sarcasm"*. While for a human reader it is apparent that the author means a clear example *was not* made of Sharapova, the lack of strong sentiment words results in all SIGN models leaving this tweet uninterpreted.

Finally, tweets that present sentiment in phrases or slang words are particularly challenging for our approach which relies on the identification and clustering of sentiment words. Consider, for example, the following two cases: (a) the sarcastic tweet *"Can't wait until tomorrow #sarcasm"*, where the positive sentiment is expressed in the phrase *can't wait*; and (b) the sarcastic tweet *"another shooting? yeah we totally need to make guns easier for people to get #sarcasm"*, where the word *totally* receives a strong sentiment despite its normal use in language. While we believe that identifying the role of *can't wait* and of *totally* in the sentiment of the above tweets can be a key to properly interpreting them, our approach that relies on a sentiment word lexicon is challenged by such cases.

Summary We presented a first attempt to approach the problem of sarcasm interpretation. Our major contributions are:

- Construction of a dataset, first of its kind, that consists of 3000 tweets each augmented with five non-sarcastic interpretations generated by human experts.
- Discussion of the proper evaluation in our task. We proposed a battery of human measures and compared their performance to the accepted measures in related fields such as machine translation.
- An algorithmic approach: sentiment based monolingual machine translation. We demonstrated the strength of our approach and pointed on cases that are currently beyond its reach.

Several challenges are still to be addressed in future research so that sarcasm interpretation can be performed in a fully automatic manner. These include the design of appropriate automatic evaluation measures as well as improving the algorithmic approach so that it can take world knowledge into account and deal with cases where the sentiment of the input tweet is not expressed with a clear sentiment words.

We are releasing our dataset with its sarcasm interpretation guidelines, the code of the SIGN algorithms, and the output of the various algorithms considered in this paper (`https://github.com/Lotemp/SarcasmSIGN`). We hope this new resource will help researchers make further progress on this new task.

References

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *Ninth International AAAI Conference on Web and Social Media*. http://dblp.uni-trier.de/rec/bib/conf/icwsm/BammanS15.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 597–604. www.aclweb.org/anthology/P05-1074.

Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Proceedings of the 5th workshop on computational approaches to subjectivity, sentiment and social media analysis. pages 50–58. https://doi.org/10.3115/v1/W14-2609.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 190–200. www.aclweb.org/anthology/P11-1020.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. of EMNLP*. https://doi.org/10.3115/v1/d14-1179.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*. Association for Computational Linguistics, pages 107–116. https://www.aclweb.org/anthology/W/W10/W10-2914.pdf.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 85–91. www.aclweb.org/anthology/W11-2107.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., pages 138–145. https://doi.org/10.3115/1289189.1289273.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*. http://aclweb.org/anthology/L06-1225.

Raymond W Gibbs and Herbert L Colston. 2007. *Irony in language and thought: A cognitive science reader*. Psychology Press.

Laurence Gillick and Stephen J Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *Proc. of ICASSP*. IEEE. https://doi.org/10.1109/ICASSP.1989.266481.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 581–586. http://www.aclweb.org/anthology/P11-2102.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 187–197. www.aclweb.org/anthology/W11-2123.

Andreas M Kaplan and Michael Haenlein. 2011. Two hearts in three-quarter time: How to waltz the social media/viral marketing dance. *Business Horizons* 54(3):253–263. https://doi.org/10.1016/j.bushor.2011.01.006.

Yael Kimhi. 2014. Theory of mind abilities and deficits in autism spectrum disorders. *Topics in Language Disorders* 34(4):329–343. https://doi.org/10.1097/tld.0000000000000033.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180. https://www.aclweb.org/anthology/P/P07/P07-2.pdf.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 48–54. www.aclweb.org/anthology/N/N03/N03-1017.ps.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 302–308. https://doi.org/10.3115/v1/P14-2050.

Chin-Yew Lin. 2004. Text summarization branches out (acl-04 workshop). http://aclweb.org/anthology/W04-1013.

Diana Maynard, Kalina Bontcheva, and Dominic Rout. 2012. Challenges in developing opinion mining tools for social media. *Proceedings of the@ NLP can u tag# usergeneratedcontent (LREC-12 workshop)* pages 15–22.

Diana Maynard and Mark A Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *LREC*. pages 4238–4243. http://dblp.uni-trier.de/rec/bib/conf/lrec/MaynardG14.

Inc Merriam-Webster. 1983. *Webster's ninth new collegiate dictionary*. Merriam-Webster. https://doi.org/10.1353/dic.1984.0017.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography* 3(4):235–244. https://doi.org/10.1093/ijl/3.4.235.

Douglas Colin Muecke. 1982. *Irony and the Ironic*. Methuen.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics* 29(1):19–51. http://aclweb.org/anthology/J03-1002.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval* 2(1-2):1–135. http://dblp.uni-trier.de/rec/bib/journals/ftir/PangL07.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318. www.aclweb.org/anthology/P02-1040.pdf.

Candida C Peterson, Henry M Wellman, and Virginia Slaughter. 2012. The mind behind the message: Advancing theory-of-mind scales for typically developing children, and those with deafness, autism, or asperger syndrome. *Child development* 83(2):469–485. https://doi.org/10.1111/j.1467-8624.2011.01728.x.

Ana-Maria Popescu, Bao Nguyen, and Oren Etzioni. 2005. Opine: Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP interactive demonstrations*. Association for Computational Linguistics, pages 32–33. https://doi.org/10.3115/1225733.1225750.

Chris Quirk, Chris Brockett, and William B Dolan. 2004. Proceedings of the 2004 conference on empirical methods in natural language processing. pages 142–149. http://aclweb.org/anthology/W04-3219.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 704–714. http://aclweb.org/anthology/D13-1066.

SG Shamay-Tsoory, Rachel Tomer, and Judith Aharon-Peretz. 2005. The neuroanatomical basis of understanding sarcasm and its relationship to social cognition. *Neuropsychology* 19(3):288. https://doi.org/10.1037/0894-4105.19.3.288.

FJ Stingfellow. 1994. *The Meaning of Irony*. New York: State University of NY.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *ICWSM*. http://dblp.uni-trier.de/rec/bib/conf/icwsm/TsurDR10.

Po-Ya Angela Wang. 2013. # irony or# sarcasma quantitative and qualitative study based on twitter. In *Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27)*. https://aclweb.org/anthology/Y/Y13/Y13-1035.pdf.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational linguistics* 30(3):277–308. http://aclweb.org/anthology/J04-3002.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2010. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics, pages 203–207. http://dblp.uni-trier.de/rec/bib/conf/inlg/WubbenBK10.

Wei Xu, Chris Callison-Burch, and William B Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* https://doi.org/10.18653/v1/s15-2001.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* http://dblp2.uni-trier.de/rec/bib/journals/corr/abs-1212-5701.

Active Sentiment Domain Adaptation

Fangzhao Wu[†], **Yongfeng Huang**[†,*], **and Jun Yan**[‡]
[†]Department of Electronic Engineering, Tsinghua University
[‡]Microsoft Research Asia, Beijing, China
wufangzhao@gmail.com, yfhuang@tsinghua.edu.cn, junyan@microsoft.com

Abstract

Domain adaptation is an important technology to handle domain dependence problem in sentiment analysis field. Existing methods usually rely on sentiment classifiers trained in source domains. However, their performance may heavily decline if the distributions of sentiment features in source and target domains have significant difference. In this paper, we propose an active sentiment domain adaptation approach to handle this problem. Instead of the source domain sentiment classifiers, our approach adapts the general-purpose sentiment lexicons to target domain with the help of a small number of labeled samples which are selected and annotated in an active learning mode, as well as the domain-specific sentiment similarities among words mined from unlabeled samples of target domain. A unified model is proposed to fuse different types of sentiment information and train sentiment classifier for target domain. Extensive experiments on benchmark datasets show that our approach can train accurate sentiment classifier with less labeled samples.

1 Introduction

Sentiment classification is widely known as a domain-dependent problem (Liu, 2012; Pang and Lee, 2008; Blitzer et al., 2007; Pan et al., 2010). This is because different domains usually have many different sentiment expressions. For example, "lengthy" and "boring" are popularly used in *Book* domain to express negative sentiment. However, they are rare in *Kitchen appliance* domain. Moreover, the same word or phrase may convey different sentiments in different domains. For instance, "unpredictable" is frequently used to express positive sentiment in *Movie* domain (e.g., "The plot of this movie is fun and unpredictable"). However, it tends to be used as a negative word in *Kitchen appliance* domain (e.g., "Even holding heat is unpredictable. It is just terrible!"). Thus, every domain has many domain-specific sentiment expressions, which cannot be captured by other domains. The performance of directly applying a general sentiment classifier or a sentiment classifier trained in other domains to target domain is usually suboptimal.

Since there are a large number of domains in user-generated content, it is impractical to manually annotate enough samples for each domain to train an accurate domain-specific sentiment classifier. Thus, sentiment domain adaptation, which transfers the sentiment classifier trained in a source domain with sufficient labeled data to a target domain with no or scarce labeled data, has been widely studied (Blitzer et al., 2007; Pan et al., 2010; He et al., 2011; Glorot et al., 2011). Existing sentiment domain adaptation methods are mainly based on transfer learning techniques. Many of them try to learn a new feature representation to augment or replace the original feature space in order to reduce the gap of sentiment feature distributions between source and target domains (Pan et al., 2010; Glorot et al., 2011). For example, Blitzer et al. (2007) proposed to learn a latent representation for domain-specific words from both source and target domains by using pivot features as bridge. The advantage of these methods is that no labeled data in target domain is needed. However, when the distributions of sentiment features in source and target domains have significant difference, the performance of domain adaptation will heavily decline (Li et al., 2013). In some cases, the performance of adaptation is even lower than

*Corresponding author.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1701–1711
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1156

that without adaptation, which is usually known as *negative transfer* (Pan and Yang, 2010).

In this paper, we propose an active sentiment domain adaptation approach to handle this problem by incorporating both general sentiment information and a small number of actively selected labeled samples from target domain. More specifically, in our approach the general sentiment information extracted from sentiment lexicons is adapted to target domain using domain-specific sentiment similarities among words. The general sentiment information is regarded as a "background" domain to transfer. The word similarities are extracted from unlabeled samples of target domain using both syntactic rules and co-occurrence patterns. Then we actively select and annotate a small number of informative samples from target domain in an active learning manner. These labeled samples are incorporated into our approach to improve the performance of sentiment domain adaptation. A unified model is proposed to incorporate different types of sentiment information to train sentiment classifier for target domain. Extensive experiments were conducted on benchmark datasets. The experimental results show that our approach can train accurate sentiment classifiers and reduce the manual annotation effort.

2 Related Work

2.1 Sentiment Domain Adaptation

Sentiment classification is well known as a highly domain-dependent task, and domain adaptation is widely studied in sentiment analysis field to handle this problem (Blitzer et al., 2007; Pan et al., 2010; He et al., 2011; Glorot et al., 2011). Existing sentiment domain adaptation methods are mainly based on transfer learning technique (Pan and Yang, 2010), where sentiment classifiers are trained in one or multiple source domains with sufficient labeled samples, and then applied to target domain where there is no or only scarce labeled samples. In order to reduce the gap of sentiment feature distributions between source and target domains, many sentiment domain adaptation methods try to learn a new feature representation to augment or replace the original feature space. For example, Pan et al. (2010) proposed a sentiment domain adaptation method based on spectral feature alignment (*SFA*) algorithm. They first manually selected several domain-independent features and computed the associations between domain-specific features and domain-independent features. After that they built a bipartite graph where domain-independent and domain-specific features were regarded as two types of nodes. Then domain-specific features were grouped into several clusters using spectral clustering algorithm. These clusters were used to augment the original feature representations. Glorot et al. (2011) proposed a sentiment domain adaptation method based on a deep learning technique, i.e., Stacked Denoising Autoencoders. They learned the parameters of neural networks using unlabeled samples from both source and target domains, and used the hidden nodes of the neural networks as the latent feature representations of both domains. Then they trained sentiment classifiers using source domain labeled data in this new feature space and applied it to target domain. The advantage of these sentiment domain adaptation methods is that they do not rely on the labeled data in target domain. However, they have a common shortcoming, i.e., when the distributions of sentiment features in source and target domains have significant difference, the performance of domain adaptation will heavily decline (Li et al., 2013). In some cases, *negative transfer* may happen (Blitzer et al., 2007; Li et al., 2013), which means the performance of adaptation is worse than that without adaptation (Pan and Yang, 2010). Different from many existing sentiment domain adaptation methods, in our approach we adapt the general sentiment information in sentiment lexicons to target domain with the help of a small number of labeled samples which are selected and annotated in an active learning mode. Since the sentiment words in general-purpose sentiment lexicons usually convey consistent sentiment polarities in different domains, and the actively selected labeled samples contain rich domain-specific sentiment information of target domain, our approach can effectively reduce the risk of negative transfer.

The usefulness of labeled samples from target domain in sentiment domain adaptation has been observed by previous research works (Choi and Cardie, 2009; Chen et al., 2011; Li et al., 2013; Wu et al., 2016). For example, Choi and Cardie (2009) proposed to adapt a sentiment lexicon to a specific domain by exploiting both the relations among words which co-occur in the same sentiment expressions and the relations between words and labeled sentiment expressions. However, the

labeled samples used in these methods are randomly selected, while in our approach we actively select informative samples from target domain to annotate. Thus, our approach has the potential to reduce the manual annotation effort.

2.2 Active Learning

Active learning is a useful technique in scenarios where unlabeled data is abundant but their labels are difficult or expensive to obtain (Tong and Koller, 2002; Settles, 2010). By actively selecting informative samples to label, active learning can effectively reduce the annotation effort, and improve the classification performance with limited budget (Li et al., 2012). An important problem in active learning is how to evaluate the informativeness of unlabeled samples (Fu et al., 2013). Different methods have been applied to select informative samples, such as uncertainty sampling (Zhu et al., 2010; Yang et al., 2015), query-by-committee (Freund et al., 1997; Li et al., 2013) and so on. In our approach, uncertainty combined with density is used to measure the informativeness of samples. A major difference between our approach and existing active learning methods is that in existing methods the parameters of the initial classifier are either initialized as zero (Cesa-Bianchi et al., 2006) or learned from a set of randomly selected samples (Settles, 2010). In contrast, the initial sentiment classifier in our approach is constructed by adapting the general sentiment information to target domain via the domain-specific sentiment similarities among words.

There are a few works that apply active learning methods to sentiment domain adaptation task (Rai et al., 2010; Li et al., 2013). For example, Rai et al. (2010) proposed an online active learning algorithm for sentiment domain adaptation. They started with a sentiment classifier trained on the labeled samples of a source domain. Then they sequentially selected informative samples in target domain to annotate with a probability positively related to classification uncertainty. The newly annotated samples were used to update the sentiment classifier in an online learning manner. Li et al. (2013) proposed another active learning method for cross-domain sentiment classification. In their method they trained two sentiment classifiers, one on the labeled samples of source domain, and the other one on the labeled samples of target domain. Then query-by-committee strategy was used to select the informative instances from target domain. Different from these methods, our approach does not rely on the labeled data of source domains. Instead, in our approach the general sentiment information in sentiment lexicons is actively adapted to target domain, which usually has better generalization ability in various domains than the sentiment classifier trained in a source domain. In addition, our approach can incorporate the domain-specific sentiment similarities among words mined from unlabeled samples of target domain, which are not considered in these methods.

3 Active Sentiment Domain Adaptation

3.1 Notations

First we introduce several notations that will be used in remaining part of this paper. Denote the general sentiment information extracted from a general-purpose sentiment lexicon as $\mathbf{p} \in \mathbb{R}^{D\times 1}$, where D is the vocabulary size. If the i_{th} word is labeled as positive (or negative) in the sentiment lexicon, then $p_i = +1$ (or $p_i = -1$). Otherwise, $p_i = 0$. Following many previous works in sentiment classification field (Blitzer et al., 2007; Pan et al., 2010), here we select linear classifier as sentiment classifier, and denote the linear classification model as $\mathbf{w} \in \mathbb{R}^{D\times 1}$. We use $f(\mathbf{x}_i, y_i, \mathbf{w})$ to represent the loss of classifying the i_{th} labeled sample in target domain under the classification model $\mathbf{w}$, where f is the classification loss function, $\mathbf{x}_i \in \mathbb{R}^{D\times 1}$ is the feature vector of this sample and y_i is its sentiment label. In this paper we focus on binary sentiment classification and $y_i \in \{+1, -1\}$. In addition, we select log loss for f. Thus, $f(\mathbf{x}_i, y_i, \mathbf{w}) = \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$. Besides, we use $\mathbf{S} \in \mathbb{R}^{D\times D}$ to represent the sentiment similarities among words extracted from unlabeled samples of target domain.

3.2 Domain-Specific Sentiment Similarities

Next we introduce the extraction of domain-specific sentiment similarities among words from unlabeled samples of target domain. Two types of similarities are extracted in this paper. The first one is based on syntactic rules, which is inspired by (Hatzivassiloglou and McKeown, 1997; Huang et al., 2014; Wu and Huang, 2016). If two words have the same POS-tag such as adjective, verb, and adverb, and they are connected by coordinating conjunction "and" in the same sentence, then we regard they convey the same sentiment polarity. In

addition, if two words are connected by adversative conjunction "but" and have the same POS-tag, then they are assumed to have opposite sentiment polarities. Denote $\mathbf{S}^r \in \mathbb{R}^{D\times D}$ as the sentiment similarities extracted from unlabeled samples according to syntactic rules, and the similarity score between words i and j is defined as:

$$S^r_{i,j} = \frac{N^s_{i,j} - N^o_{i,j}}{N^s_{i,j} + N^o_{i,j} + \alpha_1}, \tag{1}$$

where $N^s_{i,j}$ and $N^o_{i,j}$ are the frequencies of words i and j having the same or opposite sentiments respectively according to the syntactic rules, and α_1 is a positive smoothing factor. If two words have much higher frequency of sharing the same sentiment than opposite sentiments, then they will have a larger positive sentiment similarity score. Note that $S^r_{i,j}$ can be negative according to Eq. (1). Here we focus on sentiment similarity rather than dissimilarity, and set all the negative values in $\mathbf{S}^r$ to zero. The range of $S^r_{i,j}$ is $[0, 1]$.

The second type of sentiment similarities are extracted according to the co-occurrence patterns among words. It is inspired by the observation that words frequently co-occurring with each other not only have a high probability to have similar semantics, but also tend to share similar sentiments (Turney, 2002; Velikovich et al., 2010; Yogatama and Smith, 2014; Tang et al., 2015; Hamilton et al., 2016). In this paper, we compute the co-occurrence between words in the context of document. Denote $\mathcal{D}$ as the set of all documents, and N^i_d as the frequency of word i appearing in document d. Then, the sentiment similarity score between words i and j based on their co-occurrence patterns is defined as:

$$S^c_{i,j} = \frac{\sum_{d\in\mathcal{D}} \min\{N^i_d, N^j_d\}}{\sum_{d\in\mathcal{D}} \max\{N^i_d, N^j_d\} + \alpha_2}, \tag{2}$$

where α_2 is a positive smoothing parameter. If two words frequently co-occur with each other in many documents, then they will have a high sentiment similarity score according to Eq. (2). The range of $S^c_{i,j}$ is also $[0, 1]$. Denote $\mathbf{S}^c \in \mathbb{R}^{D\times D}$ as the set of all sentiment similarities extracted according to co-occurrence patterns.

The sentiment similarities extracted according to syntactic rules are usually of high accuracy. However, their coverage is limited, because the word pairs detected by these syntactic rules are sparse. In contrast, the coverage of sentiment similarities extracted from co-occurrence patterns is quite wide because document is a long context, while their accuracies are not as high as the similarities based on syntactic rules. Thus, we propose to combine these two types of sentiment similarities to obtain a balance between accuracy and coverage. Denote $\mathbf{S} \in \mathbb{R}^{D\times D}$ as the final sentiment similarities among words, and $S_{i,j} = \theta S^r_{i,j} + (1-\theta) S^c_{i,j}$, where $\theta \in [0, 1]$ is the combination coefficient. In this paper we set θ to 0.5, which means that we regard these two types of sentiment similarities as equally important.

3.3 Initial Sentiment Classifier Construction

In this section, we introduce the construction of the initial sentiment classifier to start the active learning process. Existing active learning methods usually randomly select a set of unlabeled samples to annotate and then train the initial classifier on them (Settles, 2010). However, these randomly selected samples may be redundant and not informative enough. In this paper, we propose to build the initial sentiment classifier by adapting the general sentiment information to target domain via domain-specific sentiment similarities as follows:

$$\mathbf{w}_0 = \arg\min_{\mathbf{w}} -\sum_{i=1}^{D} p_i w_i + \alpha \sum_{i=1}^{D}\sum_{j\neq i} S_{i,j}(w_i - w_j)^2, \tag{3}$$

where $\mathbf{w}_0 \in \mathbb{R}^{D\times 1}$ is the initial sentiment classifier, α is a positive regularization coefficient, p_i is the prior sentiment polarity of word i in sentiment lexicons, and $S_{i,j}$ is the sentiment similarity score between words i and j. Eq. (3) is motivated by (Bengio et al., 2006), and the quadratic cost criterion is equivalent to label propagation. In Eq. (3), $-\sum_{i=1}^{D} p_i w_i$ means that if a word i is labeled as a positive (or negative) word in a general-purpose sentiment lexicon, i.e., $p_i > 0$ (or $p_i < 0$), then we constrain that its sentiment weight in the sentiment classifier is also positive (or negative). Otherwise, a penalty will be added to the objective function. In addition, $\sum_{i=1}^{D}\sum_{j\neq i} S_{i,j}(w_i - w_j)^2$ represents that if two words share high sentiment similarity, then we constrain they have similar sentiment weights in sentiment classifier. For example, if we find that "great" and "easy" have high sentiment similarities in *Kitchen appliances* domain (e.g., "This is a great pan and easy to wash"), and "great" is a positive sentiment word in many sentiment lexicons, then we can infer that "easy" may also be a positive sentiment word in this domain by propagating the sentiment information from

"great" to "easy". In this way, the general sentiment information can be adapted to many domain-specific sentiment expressions in target domain.

3.4 Query Strategy

Active learning methods iteratively select the most informative instances to label and add them to the training set (Settles, 2010). Thus, an important issue in these methods is how to measure the informativeness of unlabeled samples. In this paper, we select classification uncertainty as the informativeness measure, which has been proven effective in many active learning methods (Zhu et al., 2010; Yang et al., 2015). Since we focus on binary sentiment classification and the classification loss function is log loss, the classification uncertainty of an unlabeled instance $\mathbf{x}$ is defined as:

$$U(\mathbf{x}) = 1 - \left| 1 - \frac{2}{1 + \exp(-\mathbf{w}^T \mathbf{x})} \right|, \tag{4}$$

where $\mathbf{w}$ is the linear sentiment classification model. The range of $U(\mathbf{x})$ is $[0, 1]$. If $|\mathbf{w}^T \mathbf{x}|$ is large, which means that current sentiment classifier is confident in classifying this instance, then the uncertainty of $\mathbf{x}$ (i.e., $U(\mathbf{x})$) will be low. If $|\mathbf{w}^T \mathbf{x}|$ is close to 0, then the sentiment classifier is very uncertain about this instance, probably because the sentiment expressions in this instance are not covered by current sentiment classifier, and the uncertainty of the instance $\mathbf{x}$ will be high. In this case, annotating this instance and adding it to the training set are beneficial, because it can provide the information of unknown sentiment expressions and has the potential to quickly improve the quality of target domain sentiment classifier.

However, many researchers have found that unlabeled instances with high uncertainties can be outliers, whose labels are useless and even misleading (Settles, 2010; Zhu et al., 2010). Thus, here we combine uncertainty with representativeness to avoid outliers. Density is proven to be an effective measure of representativeness in active learning methods (Zhu et al., 2010; Hajmohammadi et al., 2015). Here we use the k-nearest neighbour based density proposed by Zhu et al. (2010) as the representativeness measure, which is formulated as:

$$D(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} \frac{\mathbf{x}^T \mathbf{x}_i}{\|\mathbf{x}\|_2 \cdot \|\mathbf{x}_i\|_2}, \tag{5}$$

where $\mathcal{N}(\mathbf{x})$ is the set of k most similar instances of $\mathbf{x}$. The final informativeness score of an unlabeled sample is a linear combination of uncertainty and density which is formulated as follows:

$$I(\mathbf{x}) = \eta(t) U(\mathbf{x}) + (1 - \eta(t)) D(\mathbf{x}), \tag{6}$$

where $\eta(t) \in [0, 1]$ is the combination coefficient at the t_{th} iteration. In this paper, we select a monotonically increasing function for $\eta(t)$, i.e., $\eta(t) = \frac{1}{1+\exp(2-\frac{4t}{T})}$, where T is the total number of iterations. It means that at initial iterations we put more emphasis on instances with high representativeness, because the initial sentiment classifier built by adapting the general sentiment information via the domain-specific sentiment similarities is relatively weak, and we prefer to select instances with more popular sentiment expressions to annotate. As more and more labeled samples are added to the training set and the sentiment classifier becomes stronger, we gradually focus on more difficult instances, i.e., those having higher classification uncertainty scores.

3.5 Active Domain Adaptation

Based on previous discussions, in this section we introduce the complete procedure of our active sentiment domain adaptation (*ASDA*) approach. Different from existing sentiment domain adaptation methods which rely on the sentiment classifier trained in source domains to transfer, in our approach we regard the general sentiment information in sentiment lexicons as the "background" domain and adapt it to target domain with the help of a small number of labeled samples which are selected and annotated in an active learning mode. First, we build an initial sentiment classifier according to Eq. (3) by adapting the general sentiment information to target domain using the domain-specific sentiment similarities among words mined from unlabeled samples of target domain. Second, we compute the density of each unlabeled sample in $\mathcal{U}$ according to Eq. (5). Then we repeat following steps until the annotation budget has run out. First, we compute the uncertainty of each unlabeled sample in $\mathcal{U}$ according to Eq. (4), and further we compute their informativeness by combining uncertainty with density according to Eq. (6). Next, we select the unlabeled sample with the highest informativeness from $\mathcal{U}$ and manually annotate its sentiment polarity. Then we add it to the set of labeled samples $\mathcal{L}$ and remove it from $\mathcal{U}$. After that we retrain the sentiment classifier for target domain based on the general sentiment information $\mathbf{p}$, the labeled samples $\mathcal{L}$, and the domain-

specific sentiment similarities $\mathbf{S}$ as follows:

$$\arg\min_{\mathbf{w}} -\sum_{i=1}^{D} p_i w_i + \alpha \sum_{i=1}^{D} \sum_{j \neq i} S_{i,j}(w_i - w_j)^2 + \beta \sum_{\mathbf{x}_i \in \mathcal{L}} \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) + \lambda \|\mathbf{w}\|_2^2, \quad (7)$$

where α, β, and λ are nonnegative coefficients. By the term $-\sum_{i=1}^{D} p_i w_i$ we constrain that the target domain sentiment classifier learned by our approach is consistent with the general sentiment information. Through this way, the general sentiment information extracted from sentiment lexicons can be adapted to target domain. The term $\sum_{i=1}^{D} \sum_{j \neq i} S_{i,j}(w_i - w_j)^2$ is motivated by label propagation (Bengio et al., 2006). If two words tend to have high sentiment similarity with each other according to many unlabeled samples of target domain, then we constrain that their sentiment weights in the target domain sentiment classifier are also similar. The term $\sum_{\mathbf{x}_i \in \mathcal{L}} \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))$ means that we hope to minimize the empirical classification loss on labeled samples of target domain. By this term the sentiment information in the labeled samples is incorporated into the learning of target domain sentiment classifier. The L_2-norm regularization term is introduced to control model complexity. The sentiment classifier trained in Eq. (7) is further used at the next iteration of active sentiment domain adaptation until all the budget of manual annotation has been used. Then we obtain the final sentiment classifier of target domain. The complete algorithm of our active sentiment domain adaptation (*ASDA*) approach is summarized in Algorithm 1.

Algorithm 1 Active sentiment domain adaptation.

1: **Input:** The set of unlabeled samples $\mathcal{U}$, the general sentiment information $\mathbf{p}$, the domain-specific sentiment similarities $\mathbf{S}$, and the total annotation budget N.
2: **Output:** Target domain sentiment classifier $\mathbf{w}$.
3: Train the initial sentiment classifier $\mathbf{w}_0$ (Eq. (3)).
4: Compute the density of each sample $\mathbf{x}_i$ in $\mathcal{U}$ (Eq. (5)).
5: Initialize the set of labeled samples $\mathcal{L} = \emptyset$, the iteration number $t = 0$, and the sentiment classifier $\mathbf{w} = \mathbf{w}_0$.
6: **while** $t < N$ **do**
7: $t = t + 1$.
8: Compute the uncertainty score of each sample $\mathbf{x}_i$ in $\mathcal{U}$ (Eq. (4)).
9: Compute the informativeness score of each sample $\mathbf{x}_i$ in $\mathcal{U}$ (Eq. (6)).
10: Select $\mathbf{x}^*$ from $\mathcal{U}$ which has the highest informativeness score.
11: Annotate $\mathbf{x}^*$ and obtain its sentiment label y.
12: $\mathcal{L} = \mathcal{L} + \{\mathbf{x}^*, y\}$, $\mathcal{U} = \mathcal{U} - \mathbf{x}^*$.
13: Update sentiment classifier $\mathbf{w}$ according to Eq. (7).
14: **end while**

4 Experiments

4.1 Datasets

The dataset used in our experiments is the Amazon product review dataset[1] collected by Blitzer et al. (2007), which is widely used in sentiment analysis and domain adaptation research (Pan et al., 2010; Bollegala et al., 2011). This dataset contains product reviews in four domains, i.e., *Book*, *DVD*, *Electronics*, and *Kitchen appliances*. In each domain, 1,000 positive and 1,000 negative reviews as well as a large number of unlabeled samples are included. The detailed statistics of this dataset are summarized in Table 1.

	Book	*DVD*	*Electronics*	*Kitchen*
positive	1,000	1,000	1,000	1,000
negative	1,000	1,000	1,000	1,000
unlabeled	973,194	122,438	21,009	17,856

Table 1: The statistics of the Amazon dataset.

Following many previous works (Blitzer et al., 2007; Bollegala et al., 2011), unigrams and bigrams were used to build feature vectors in our experiments. We randomly split the labeled samples in each domain into two parts with equal size. The first part was used as test data, and the second part was used as the pool of "unlabeled" samples to perform active learning. The general sentiment information was extracted from Bing Liu's sentiment lexicon[2] (Hu and Liu, 2004), which is one of the state-of-the-art general-purpose sentiment lexicons. The domain-specific sentiment similarities among words were extracted from the large-scale unlabeled samples. The total number of samples actively selected by our approach to annotate was set to 100. The values of α, β, and λ were set to 0.1, 1, and 1 respectively. We repeated each experiment 10 times independently and the average results were reported.

4.2 Algorithm Effectiveness

First we conducted several experiments to explore the effectiveness of our active sentiment domain adaptation (*ASDA*) approach. We hope to answer two questions via these experiments: 1) whether the domain-specific sentiment similarities among words mined from unlabeled samples of target

[1] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

[2] https://www.cs.uic.edu/liub/FBS/sentiment-analysis.html

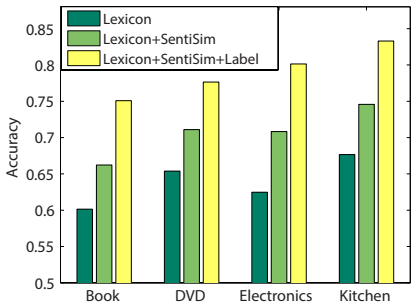


Figure 1: The performance of our approach with different combinations of sentiment information. *Lexicon*, *SentiSim*, and *Label* represent the general-purpose sentiment lexicon, the domain-specific sentiment similarities among words, and a small number of actively selected and annotated samples in target domain respectively.

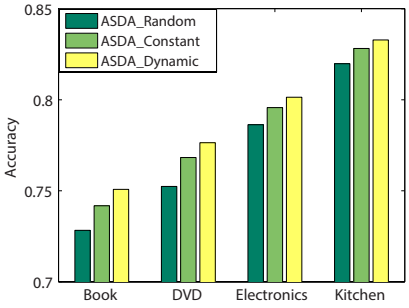


Figure 2: The performance of our approach with labeled samples selected by different strategies.

domain are useful for adapting the general sentiment information to target domain; 2) whether a small number of samples which are actively selected and annotated in target domain can help improve the domain adaptation performance. In our experiments, we implemented different versions of our *ASDA* approach using different combinations of sentiment information. The first one is *Lexicon*, which means only using the general sentiment information and no domain adaptation is conducted. It serves as a baseline. The second one is *Lexicon+SentiSim*, which means adapting general sentiment information to target domain using domain-specific sentiment similarities, but labeled samples of target domain are not incorporated. The third one is *Lexicon+SentiSim+Label*, which is the complete *ASDA* approach. The experimental results are summarized in Fig. 1.

According to Fig. 1, the performance of *Lexicon* is suboptimal. This is because the general sentiment lexicons cannot capture the domain-specific sentiment expressions in target domain (Choi and Cardie, 2009). *Lexicon+SentiSim* performs significantly better than *Lexicon*, which validates that the sentiment similarities among words extracted from unlabeled samples of target domain contain rich domain-specific sentiment information, and can help propagate the general sentiment information to many domain-specific sentiment expressions. Besides, after incorporating a small number of labeled samples which are actively selected and annotated by our approach in an active learning mode, the performance of our sentiment domain adaptation approach is significantly improved. This is because although these labeled samples are in limited size and cannot cover all the sentiment expressions in target domain, they can provide sentiment information of popular domain-specific sentiment expressions, which can be propagated to other sentiment expressions in target domain during the domain adaptation process. Thus, above experimental results validate the effectiveness of our approach.

We also conducted several experiments to verify the advantage of the actively selected samples over randomly selected samples and validate the effectiveness of our active learning algorithm. We also compared the dynamic weighting scheme for combining uncertainty and density with the constant weighting scheme. The experimental results are summarized in Fig. 2. According to Fig. 2, our approach with actively selected samples performs better than that with randomly selected samples. It indicates that these actively selected samples are more informative than randomly selected samples for sentiment domain adaptation. In addition, our approach with dynamic weighting scheme in combining uncertainty and density outperforms that with constant weighting scheme, which implies that it is beneficial to emphasize representative samples at initial iterations and gradually focus on difficult samples at later iterations. Thus, the experimental results validate the effectiveness of our active learning algorithm.

4.3 Performance Evaluation

In this section we conducted experiments to evaluate the performance of our approach by comparing it with several baseline methods. The methods to be compared include: 1) *MPQA* and *BingLiu*, using two state-of-the-art sentiment lexicons,

i.e., MPQA (Wilson et al., 2005) and Bing Liu's lexicon (Hu and Liu, 2004) for sentiment classification following the suggestions in (Hu and Liu, 2004); 2) *SVM*, *LS*, and *LR*, three popular supervised sentiment classification methods, i.e., support vector machine (Pang et al., 2002), least squares (Hu et al., 2013) and logistic regression (Wu et al., 2015); 3) *ZIAL*, the zero initialized active learning method (Cesa-Bianchi et al., 2006); 4) *LIAL*, the active learning method initialized by randomly selected labeled data (Settles, 2010); 5) *SCL* and *SFA*, two famous sentiment domain adaptation methods proposed in (Blitzer et al., 2007) and (Pan et al., 2010) respectively; 6) *ILP*, adapting sentiment lexicons to target domain via integer linear programming (Choi and Cardie, 2009); 7) *AODA*, the active online domain adaptation method (Rai et al., 2010); 8) *ALCD*, the active learning method for cross-domain sentiment classification (Li et al., 2013); 9) *ASDA*, our active sentiment domain adaptation approach. For above methods, if labeled target domain samples are needed in training, the number of labeled samples was set to 100, and if source domain labeled samples are needed in training, the number of labeled samples was set to 1,000. The parameters in baseline methods were tuned via cross-validation. The experimental results are summarized in Table 2.

	Book		*DVD*		*Electronics*		*Kitchen*	
	Acc	Fscore	Acc	Fscore	Acc	Fscore	Acc	Fscore
MPQA	0.5953	0.5673	0.6149	0.5936	0.6150	0.6070	0.6392	0.6258
BingLiu	0.6015	0.6048	0.6539	0.6604	0.6248	0.6320	0.6765	0.6930
SVM	0.6580	0.6511	0.6688	0.6652	0.7138	0.7129	0.7386	0.7412
LS	0.6543	0.6542	0.6692	0.6687	0.7194	0.7185	0.7479	0.7465
LR	0.6606	0.6582	0.6774	0.6742	0.7257	0.7226	0.7492	0.7480
RIAL	0.6693	0.6663	0.6850	0.6821	0.7310	0.7299	0.7574	0.7568
LIAL	0.6756	0.6731	0.6866	0.6838	0.7374	0.7360	0.7599	0.7595
SCL	0.7233	0.7201	0.7469	0.7438	0.7768	0.7730	0.8099	0.8095
SFA	0.7307	0.7285	0.7513	0.7485	0.7846	0.7812	0.8174	0.8153
ILP	0.6942	0.6931	0.7153	0.7124	0.7463	0.7445	0.7793	0.7768
AODA	0.6928	0.6912	0.7172	0.7165	0.7518	0.7512	0.7698	0.7690
ALCD	0.7237	0.7221	0.7369	0.7364	0.7768	0.7788	0.7979	0.7970
ASDA	0.7508	0.7501	0.7764	0.7759	0.8014	0.8011	0.8329	0.8328

Table 2: Sentiment classification performance of different methods in different domains. *Acc* and *Fscore* represent accuracy and macro-averaged Fscore respectively.

According to Table 2, the performance of directly applying sentiment lexicons to target domain is suboptimal. This is because there are many domain-specific sentiment expressions that are not covered by these general-purpose sentiment lexicons (Choi and Cardie, 2009). In addition, the performance of supervised sentiment classification methods such as *SVM*, *LS*, and *LR* is also

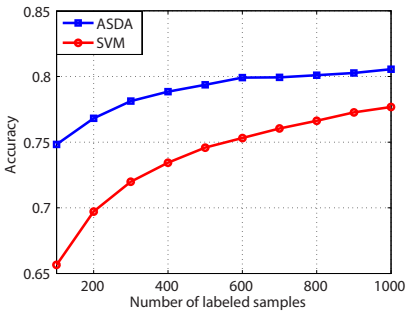
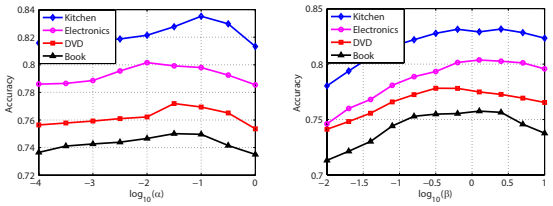


Figure 3: The performance of *ASDA* and *SVM* with different numbers of labeled samples.

limited, because the labeled samples for training are extremely scarce. The active learning methods such as *ZIAL* (Cesa-Bianchi et al., 2006) and *LIAL* (Settles, 2010) perform relatively better, because they can actively select informative samples to annotate and learn. Our approach can outperform both of them. This is because besides the labeled samples, our approach also adapts the general sentiment information in sentiment lexicons to target domain and incorporates it into the learning of target domain sentiment classifier. Our approach also performs better than state-of-the-art domain adaptation methods such as *SCL* (Blitzer et al., 2007) and *SFA* (Pan et al., 2010). It implies that a small number of actively selected labeled samples from target domain are beneficial for sentiment domain adaptation. *ILP* (Choi and Cardie, 2009) tries to adapt a sentiment lexicon to target domain, which is similar with our approach. *ILP* relies on labeled samples to extract the relations among words and relations between words and sentiment expressions. However, labeled samples in target domain are usually limited and the sentiment information in many unlabeled samples is not exploited in *ILP*. Thus, our approach can outperform it. Similar with our approach, *AODA* (Rai et al., 2010) and *ALCD* (Li et al., 2013) also apply active learning to domain adaptation. The major difference is that in our approach the general sentiment information extracted from sentiment lexicons is adapted to target domain, while in *AODA* and *ALCD* the sentiment classifier trained in source domains is transferred. The superior performance of our approach implies that the general sentiment information has better generalization ability than the sentiment classifier trained in a specific source domain, and is more suitable for sentiment domain adaptation.



(a) Parameter α. (b) Parameter β.

Figure 4: The influence of the parameter settings of α and β on the performance of our approach.

We further conducted several experiments to validate the advantage of our approach in training accurate sentiment classifier for target domain with only a few labeled samples. We varied the annotation budget, i.e., the number of labeled samples, from 100 to 1,000. The learning curve of our *ASDA* approach in *Book* domain is shown in Fig. 3. We also included a purely supervised sentiment classification method, i.e., *SVM*, in Fig. 3 as a baseline for comparison. Fig. 3 shows that our *ASDA* approach can consistently outperform *SVM* when the same number of labeled samples are used. The performance advantage of our approach is more significant when labeled samples are scarce. For example, the performance of our approach with only 200 labeled samples is similar to *SVM* with more than 800 labeled samples. Thus, the experimental results validate that by adapting the general sentiment information to target domain and selecting the most informative samples to annotate and learn, our approach can effectively reduce the manual annotation effort, and can train accurate sentiment classifier for target domain with much less labeled samples.

4.4 Parameter Analysis

In this section, we conducted several experiments to explore the influence of parameter settings on the performance of our approach. α and β are the two most important parameters in our approach, which control the relative importance of domain-specific sentiment similarities and the actively selected samples in training sentiment classifier for target domain. The experimental results of parameters α and β are summarized in Fig. 4.

According to Fig. 4, when α and β are too small, the performance of our approach is not optimal. This is because the useful sentiment information in domain-specific sentiment similarities mined from unlabeled samples and the actively selected labeled samples of target domain is not fully exploited. Thus, the performance of our approach improves when these parameters increase from a small value. However, when these parameters become too large, the performance of our approach starts to decline. This is because when β is too large the sentiment classifier learned by our approach is mainly decided by the limited labeled samples, and the general sentiment information extracted from sentiment lexicons is not fully exploited. When α is too large, the information in domain-specific sentiment similarities is over-emphasized, and many different words will have nearly the same sentiment weights. Thus, the performance of our approach in these scenarios is also not optimal. A moderate value of α and β is most suitable for our approach.

5 Conclusion

In this paper we present an active sentiment domain adaptation approach to train accurate sentiment classifier for target domain with less labeled samples. In our approach, the general sentiment information in sentiment lexicons is adapted to target domain with the help of a small number of labeled samples which are selected and annotated in an active learning mode. Both classification uncertainty and density are considered when selecting informative samples to label. In addition, we extract domain-specific sentiment similarities among words from unlabeled samples of target domain based on both syntactic rules and co-occurrence patterns, and incorporate them into the domain adaptation process to propagate the general sentiment information to many domain-specific sentiment words in target domain. We also propose a unified model to incorporate different types of sentiment information to train sentiment classifier for target domain. Experimental results on benchmark datasets show that our approach can train accurate sentiment classifier and at same time reduce the manual annotation effort.

Acknowledgements

This research is supported by the Key Research Project of the Ministry of Science and Technology of China (Grant no. 2016YFB0800402) and the Key Program of National Natural Science Foundation of China (Grant nos. U1536201, U1536207, and U1405254).

References

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. Label propagation and quadratic criterion. *Semi-supervised learning* 10.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*. volume 7, pages 440–447. http://aclweb.org/anthology-new/P/P07/P07-1056.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *ACL:HLT*. pages 132–141. http://aclweb.org/anthology/P11-1014.

Nicolo Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. 2006. Worst-case analysis of selective sampling for linear classification. *Journal of Machine Learning Research* 7(Jul):1205–1230.

Minmin Chen, Kilian Q Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *NIPS*. pages 2456–2464.

Yejin Choi and Claire Cardie. 2009. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *EMNLP*. pages 590–598. http://aclweb.org/anthology/D09-1062.

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning* 28(2-3):133–168. http://dx.doi.org/10.1023/A:1007330508534.

Yifan Fu, Xingquan Zhu, and Bin Li. 2013. A survey on instance selection for active learning. *Knowledge and Information Systems* 35(2):249–283. https://doi.org/10.1007/s10115-012-0507-8.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*. pages 513–520.

Mohammad Sadegh Hajmohammadi, Roliana Ibrahim, Ali Selamat, and Hamido Fujita. 2015. Combination of active learning and self-training for cross-lingual sentiment classification with density analysis of unlabelled samples. *Information sciences* 317:67–77. http://dx.doi.org/10.1016/j.ins.2015.04.003.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *EMNLP*. pages 595–605. http://aclweb.org/anthology/D/D16/D16-1057.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1997. Predicting the semantic orientation of adjectives. In *ACL*. pages 174–181. http://aclweb.org/anthology/P/P97/P97-1023.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *ACL:HLT*. pages 123–131. http://aclweb.org/anthology/P11-1013.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*. pages 168–177. http://doi.acm.org/10.1145/1014052.1014073.

Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. 2013. Exploiting social relations for sentiment analysis in microblogging. In *WSDM*. pages 537–546. http://doi.acm.org/10.1145/2433396.2433465.

Sheng Huang, Zhendong Niu, and Chongyang Shi. 2014. Automatic construction of domain-specific sentiment lexicon based on constrained label propagation. *Knowledge-Based Systems* 56:191–200. http://dx.doi.org/10.1016/j.knosys.2013.11.009.

Lianghao Li, Xiaoming Jin, Sinno Jialin Pan, and Jian-Tao Sun. 2012. Multi-domain active learning for text classification. In *KDD*. pages 1086–1094. http://doi.acm.org/10.1145/2339530.2339701.

Shoushan Li, Yunxia Xue, Zhongqing Wang, and Guodong Zhou. 2013. Active learning for cross-domain sentiment classification. In *IJCAI*. pages 2127–2133.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies* 5(1):1–167.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *WWW*. ACM, pages 751–760. http://doi.acm.org/10.1145/1772690.1772767.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *TKDE* 22(10):1345–1359. http://dx.doi.org/10.1109/TKDE.2009.191.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval* 2(1-2):1–135. http://dx.doi.org/10.1561/1500000011.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP*. pages 79–86. https://doi.org/10.3115/1118693.1118704.

Piyush Rai, Avishek Saha, Hal Daumé III, and Suresh Venkatasubramanian. 2010. Domain adaptation meets active learning. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*. pages 27–32. http://aclweb.org/anthology/W10-0104.

Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52(55-66):11.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*. ACM, pages 1165–1174. http://doi.acm.org/10.1145/2783258.2783307.

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research* 2:45–66. http://dx.doi.org/10.1162/153244302760185243.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *ACL*. pages 417–424. http://dx.doi.org/10.3115/1073083.1073153.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *NAACL*. pages 777–785. http://www.aclweb.org/anthology/N10-1119.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *EMNLP*. pages 347–354. http://dx.doi.org/10.3115/1220575.1220619.

Fangzhao Wu and Yongfeng Huang. 2016. Sentiment domain adaptation with multiple sources. In *ACL*. pages 301–310. http://aclweb.org/anthology/P16-1029.

Fangzhao Wu, Yangqiu Song, and Yongfeng Huang. 2015. Microblog sentiment classification with contextual knowledge regularization. In *AAAI*. pages 2332–2338.

Fangzhao Wu, Sixing Wu, Yongfeng Huang, Songfang Huang, and Yong Qin. 2016. Sentiment domain adaptation with multi-level contextual sentiment knowledge. In *CIKM*. ACM, pages 949–958. https://doi.org/10.1145/2983323.2983851.

Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G. Hauptmann. 2015. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision* 113(2):113–127. http://dx.doi.org/10.1007/s11263-014-0781-x.

Dani Yogatama and Noah A. Smith. 2014. Making the most of bag of words: Sentence regularization with alternating direction method of multipliers. In *ICML*. pages 656–664.

Jingbo Zhu, Huizhen Wang, Benjamin K Tsou, and Matthew Ma. 2010. Active learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on Audio, Speech, and Language Processing* 18(6):1323–1331. http://dx.doi.org/10.1109/TASL.2009.2033421.

Volatility Prediction using Financial Disclosures Sentiments with Word Embedding-based IR Models

Navid Rekabsaz[1], Mihai Lupu[1], Artem Baklanov[2], Allan Hanbury[1], Alexander Dür[3], Linda Anderson[1]
[1 3]TU WIEN
[2]International Institute for Applied Systems Analysis (IIASA)
[2]N.N. Krasovskii Institute of Mathematics and Mechanics
[1]{family name}@ifs.tuwien.ac.at
[2]baklanov@iiasa.ac.at
[3]alexander.duer@tuwien.ac.at

Abstract

Volatility prediction—an essential concept in financial markets—has recently been addressed using sentiment analysis methods. We investigate the sentiment of annual disclosures of companies in stock markets to forecast volatility. We specifically explore the use of recent Information Retrieval (IR) term weighting models that are effectively extended by related terms using word embeddings. In parallel to textual information, factual market data have been widely used as the mainstream approach to forecast market risk. We therefore study different fusion methods to combine text and market data resources. Our word embedding-based approach significantly outperforms state-of-the-art methods. In addition, we investigate the characteristics of the reports of the companies in different financial sectors.

1 Introduction

Financial volatility is an essential indicator of instability and risk of a company, sector or economy. Volatility forecasting has gained considerable attention during the last three decades. In addition to using historic stock prices, new methods in this domain use sentiment analysis to exploit various text resources, such as financial reports (Kogan et al., 2009; Wang et al., 2013; Tsai and Wang, 2014; Nopp and Hanbury, 2015), news (Kazemian et al., 2014; Ding et al., 2015), message boards (Nguyen and Shirai, 2015), and earning calls (Wang and Hua, 2014).

An interesting resource of textual information are the companies' annual disclosures, known as *10-K filing* reports. They contain comprehensive information about the companies' business as well as risk factors. Specifically, section *Item 1A - Risk Factors* of the reports contains information about the most significant risks for the company. These reports are however long, redundant, and written in a style that makes them complex to process. Dyer et al. (2016) notes that: *"10-K reports are getting more redundant and complex* [...] *(it) requires a reader to have 21.6 years of formal education to fully comprehend"*. Dyer et al. also analyse the topics discussed in the reports and observe a constant increase over the years in both the length of the documents as well as the number of topics. They claim that the increase in length is not the result of economic factors but is due to verboseness and redundancy in the reports. They suggest that only the risk factors topic appears to be useful and informative to investors. Their analysis motivates us to study the effectiveness of the Risk Factors section for volatility prediction.

Our research builds on previous studies on volatility prediction and information analysis of 10-K reports using sentiment analysis (Kogan et al., 2009; Tsai and Wang, 2014; Wang et al., 2013; Nopp and Hanbury, 2015; Li, 2010; Campbell et al., 2014), in the sense that since the reports are long (average length of 5000 words), different approaches are required, compared with studies of sentiment analysis on short-texts. Such previous studies on 10-K reports have mostly used the data before 2008 and there is little work on the analysis of the informativeness and effectiveness of the recent reports with regards to volatility prediction. We will indeed show that the content of the reports changes significantly not only before and after 2008, but rather in a cycle of 3-4 years.

In terms of use of the textual content for volatility prediction, this paper shows that state-of-the-art Information Retrieval (IR) term weighting models, which benefit from word embedding information, have a significantly positive impact on prediction accuracy. The most recent study on the topic (Tsai and Wang, 2014) used related terms obtained by word embeddings to expand the

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1712–1721
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1157

lexicon of sentiment terms. In contrast, similar to Rekabsaz et al. (2016b), we define the weight of each lexicon term by extending it to the similar terms in the document. The significant improvement of this approach for document retrieval by capturing the importance of the terms motivates us to apply it on sentiment analysis. We extensively evaluate various state-of-the-art sentiment analysis methods to investigate the effectiveness of our approach.

In addition to text, factual market data (i.e. historical prices) provide valuable resources for volatility prediction e.g. in the framework of GARCH models (Engle, 1982). An emerging question is how to approach the combination of the textual and factual market information. We propose various methods for this issue and show the performance and characteristics of each.

The financial system covers a wide variety of industries, from daily-consumption products to space mission technologies. It is intuitive to consider that the factors of instability and uncertainty are different between the various sectors while similar inside them. We therefore also analyse the sentiment of the reports of each sector separately and study their particular characteristics.

The present study shows the value of information in the 10-K reports for volatility prediction. Our proposed approach to sentiment analysis significantly outperforms state-of-the-art methods (Kogan et al., 2009; Tsai and Wang, 2014; Wang et al., 2013). We also show that performance can be further improved by effectively combining textual and factual market information. In addition, we shed light on the effects of tailoring the analysis to each sector: despite the reasonable expectation that domain-specific training would lead to improvements, we show that our general model generalizes well and outperforms sector-specific trained models.

The remainder of the paper is organized as follows: in the next section, we review the state-of-the-art and related studies. Section 3 formulates the problem, followed by a detailed explanation of our approach in Section 4. We explain the dataset and settings of the experiments in Section 5, followed by the full description of the experiments in Section 6. We conclude the work in Section 7.

2 Related Work

Market prediction has been attracting much attention in recent years in the natural language processing community. Kazemian et al. (2014) use sentiment analysis for predicting stock price movements in a simulated security trading system using news data, showing the advantages of the method against simple trading strategies. Ding et al. (2015) address a similar objective while using deep learning to extract and learn events in the news. Xie et al. (2013) introduce a semantic tree-based model to represent news data for predicting stock price movement. Luss et al. (2015) also exploit news in combination with return prices to predict intra-day price movements. They use the Multi Kernel Learning (MKL) algorithm for combining the two features. The combination shows improvement in final prediction in comparison to using each of the features alone. Motivated by this study, we investigate the performance of the MKL algorithm as one of the methods to combine the textual with non-textual information. Other data resources, such as stocks' message boards, are used by Nguyen and Shirai (2015) to study topic modelling for aspect-based sentiment analysis. Wang and Hua (2014) investigate the sentiment of the transcript of earning calls for volatility prediction using the Gaussian Copula regression model.

While the mentioned studies use short-length texts (sentence or paragraph level), approaching long texts (document level) for market prediction is mainly based on n-gram bag of words methods. Nopp and Hanbury (2015) study the sentiment of banks' annual reports to assess banking systems risk factors using a finance-specific lexicon, provided by Loughran and McDonald (2011), in both unsupervised and supervised manner.

More directly related to the informativeness of the 10-K reports for volatility prediction, Kogan et al. (2009) use a linear Support Vector Machine (SVM) algorithm on the reports published between 1996–2006. Wang et al. (2013) improve upon this by using the Loughran and McDonald (2011) lexicon, observing improvement in the prediction. Later, Tsai and Wang (2014) apply the same method as Wang et al. (2013) while additionally using word embedding to expand the financial lexicon. We reproduce all the methods in these studies, and show the advantage of our sentiment analysis approach.

3 Problem Formulation

In this section, we formulate the volatility forecasting problem and the prediction objectives of our experiments. Similar to previous studies (Christiansen et al., 2012; Kogan et al., 2009; Tsai and Wang, 2014), volatility is defined as the natural log of the standard deviation of (adjusted) return prices in a window of τ days. This definition is referred to as standard volatility (Li and Hong, 2011) or realized volatility (Liu and Tse, 2013), defined as follows:

$$v_{[s,s+\tau]} = ln\left(\sqrt{\frac{\sum_{t=s}^{s+\tau}(r_t - \bar{r})^2}{\tau}}\right) \quad (1)$$

where r_t is the return price and $\bar{r}$ the mean of return prices. The return price is calculated by $r_t = ln(P_t) - ln(P_{t-1})$, where P_t is the (adjusted) closing price of a given stock at the trading date t.

Given an arbitrary report i, we define a prediction label y_i^k as the volatility of the stock of the reporting company in the kth quarter-sized window starting from the issue date of the report s_i:

$$y_i^k = v_{[s_i+64(k-1),s_i+64k]} \quad (2)$$

Every quarter is considered as per convention, 64 working days, while the full year is assumed to have 256 working days.

We use 8 learners for labels y^1 to y^8. For brevity, unless otherwise mentioned, we report the volatility of the first year by calculating the mean of the first four quartiles after the publication of each report.

4 Methodology

We first describe our text sentiment analysis methods, followed by the features obtained from factual market data, and finally explain the methods to combine textual and market feature sets.

4.1 Sentiment Analysis

Similar to previous studies (Nopp and Hanbury, 2015; Wang et al., 2013), we extract the keyword set from a finance-specific lexicon (Loughran and McDonald, 2011) using the positive, negative, and uncertain groups, stemmed using the Porter stemmer. We refer to this keyword set as `Lex`. Tsai and Wang (2014) expanded this set by adding the top 20 related terms to each term to the original set. The related terms are obtained using the Word2Vec (Mikolov et al., 2013) model, built on the corpus of all the reports, with Cosine similarity. We also use this expanded set in our experiments and refer to it as `LexExt`.

The following word weighting schemes are commonly used in Information Retrieval and we consider them as well in our study:

$\mathbf{TC}: \quad log(1 + tc_{d_i}(t))$

$\mathbf{TF}: \quad \frac{log(1+tc_{d_i}(t))}{\|d_i\|}$

$\mathbf{TFIDF}: \quad \frac{log(1+tc_{d_i}(t))}{\|d_i\|} log(1 + \frac{|d_i|}{df(t)})$

$\mathbf{BM25}: \frac{(k+1)\overline{tf_{d_i}(t)}}{k+\overline{tf_{d_i}(t)}}, \quad \overline{tf_{d_i}(t)} = \frac{tc_{d_i}(t)}{(1-b)+b\frac{|d_i|}{avgdl}}$

where $tc_{d_i}(t)$ is the number of occurrences of keyword t in report i, $\|d_i\|$ denotes the Euclidean norm of the keyword weights of the report, $|d_i|$ is the length of the report (number of the words in the report), $avgdl$ is the average document length, and finally k and b are parameters. For them, we use the settings used in previous studies (Rekabsaz et al., 2016b) i.e. $k = 1.2$ and $b = 0.65$.

In addition to the standard weighting schemes, we use state-of-the-art weighting methods in Information Retrieval (Rekabsaz et al., 2016b) which benefit directly from word embedding models: They exploit similarity values between words provided by the word embedding model into the weighting schemes by extending the weight of each lexicon keyword with its similar words:

$$\widehat{tc_{d_i}}(t) = tc_{d_i}(t) + \sum_{t' \in R(t)} sim(t,t')tc_{d_i}(t') \quad (3)$$

where $R(t)$ is the list of similar words to the keyword t, and $sim(t,t')$ is the Cosine similarity value between the vector representations of the words t and t'. As previously suggested by Rekabsaz et al. (2016a, 2017), we use the Cosine similarity function with threshold 0.70 for selecting the set $R(t)$ of similar words.

We define the extended versions of the standard weighting schemes as $\widehat{\mathbf{TC}}$, $\widehat{\mathbf{TF}}$, $\widehat{\mathbf{TFIDF}}$, and $\widehat{\mathbf{BM25}}$ by replacing $tc_{d_i}(t)$ with $\widehat{tc_{d_i}}(t)$ in each of the schemes.

The feature vector generated by the weights of the `Lex` or `LexExt` lexicons is highly sparse, as the number of dimensions is larger than the number of data-points. We therefore reduce the dimensions by applying Principle Component Analysis (PCA). Our initial experiments show 400 dimen-

sion as the optimum by trying on a range of dimensions from 50 to 1000.

Given the final feature vector x with l dimensions, we apply SVM as a well-known method for training both regression and classification methods. Support Vector Regression (Drucker et al., 1997) formulates the training as the following optimization problem:

$$\min_{w \in \mathbb{R}^l} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{i=1}^{N} \max(0, \|y_i - f(x_i; w)\| - \epsilon) \tag{4}$$

Similar to previous studies (Tsai and Wang, 2014; Kogan et al., 2009), we set $C = 1.0$ and $\epsilon = 0.1$. To solve the above problem, the function f can be re-parametrized in terms of a kernel function K with weights α_i:

$$f(x_i; w) = \sum_{i=1}^{N} \alpha_i K(x_i, x) \tag{5}$$

The kernel can be considered as a (similarity) function between the feature vector of the document and vectors of all the other documents. Our initial experiments showed better performance of the Radial Basis Function (RBF) kernel in comparison to linear and cosine kernels and is therefore used in this paper.

In addition, motivated by Moraes et al.(Moraes et al., 2013), we use of an Artificial Neural Network (ANN) algorithm to test the effectiveness of neural networks for automatic feature learning. We tried several neural network architectures with different regularization methods (early-stopping, regularization term, dropout). The best performing results were achieved with two hidden layers (400 and 500 nodes respectively), $tanh$ for activation function, and learning rate of 0.001 in gradient decent with early stopping. However, the networks could not provide superior results than the SVM regressors. Therefore, for this report, we only report the SVM methods.

4.2 Market Features

In addition to textual features, we define three features using the factual market data and historical prices—referred to as *market features*—as follows:

Current Volatility is calculated on the window of one quartile before the issue date of the report: $v_{[s_i-64, s_i]}$.

GARCH (Bollerslev, 1986) is a common econometric time-series model used for predicting stock price volatility. We use a GARCH (1, 1) model, trained separately for each report on intra-day return prices. We use all price data available before the issue date of the report for fitting the model. The GARCH (1, 1) model used predicts the volatility of the next day by looking at the previous day's volatility. When forecasting further than one day into the future one needs to use the model's own predictions in order to be able to make predictions for more than one day ahead. When forecasting further into the future these conditional forecasts of the variance will converge to a value called *unconditional variance*. As our forecast period is one quarter, we will approximate the volatility of future quarters with the unconditional variance.

Sector is the sector that the corresponding company of the report belongs to, namely energy (ene), basic industries (ind), finance (fin), technology (tech), miscellaneous (misc), consumer non-durables (n-dur), consumer durables (dur), capital goods (capt), consumer services (serv), public utilities (pub), and health care (hlth)[1]. The feature is converted to numerical representation using one-hot encoding.

4.3 Feature Fusion

To combine the text and market feature sets, the first approach, used also in previous studies ((Kogan et al., 2009; Wang et al., 2013)) is simply joining all the features in one feature space. In the context of multi-model learning, the method is referred to as *early fusion*.

In contrast, *late fusion* approaches first learn a model on each feature set and then use/learn a meta model to combine their results. As our second approach, we use *stacking* (Wolpert, 1992), a special case of late fusion. In stacking, we first split the training set into two parts (70%-30% portions). Using the first portion, we train separate machine learning models for each of the text and market feature sets. Next, we predict labels of the second portion with the trained models and finally train another model to capture the combinations between the outputs of the base models. In our experiments, the final model is always trained with SVM with RBF kernel.

Stacking is computationally inexpensive. How-

[1] We follow NASDAQ categorization of sectors.

ever, due to the split of the training set, the base models or the meta model may suffer from lack of training data. A potential approach to learn both the feature sets in one model is the MKL method.

The MKL algorithm (also called *intermediate fusion* (Noble et al., 2004)) extends the kernel of the SVM model by learning (simultaneous to the parameter learning) an optimum combination of several kernels. The MKL algorithm as formulated in Lanckriet et al. (2004) adds the following criterion to Eq. 5 for kernel learning:

$$K^* = \sum_i d_i K_i \quad \text{where} \sum_i d_i = 1,\ d_i \geq 0 \quad (6)$$

where K_i is a predefined kernel. Gönen and Alpaydın (2011) mention two uses of MKL: learning the optimum kernel in SVM, and combining multiple modalities (feature sets) via each kernel.

However, the optimization can be computationally challenging. We use the mklaren method (Stražar and Curk, 2016) which has linear complexity in the number of data instances and kernels. It has been shown to outperform recent multi kernel approximation approaches. We use RBF kernels for both the text and market feature sets.

5 Experiment Setup

In this section, we first describe the data, followed by introducing the baselines. We report the parameters applied in various algorithms and describe the evaluation metrics.

Dataset We download the reports of companies of the U.S. stock markets from 2006 to 2015 from the U.S. Securities and Exchange Commission (SEC) website[2]. We remove HTML tags and extract the text parts. We extract the Risk Factors section using term matching heuristics. Finally, the texts are stemmed using the Porter stemmer. We calculate the volatility values (Eq 1) and the volatility of the GARCH model based on the stock prices, collected from the Yahoo website. We filter the volatility values greater/smaller than the mean plus/minus three times the standard deviation of all the volatility values[3].

Baselines **GARCH:** although the GARCH model is of market factual information, we use it as a baseline to compare the effectiveness of text-based methods with mainstream approaches.

Market: uses all the market features. For both the GARCH and Market baselines, we use an SVM learner with RBF kernel.

Wang et al. (2013): they use the `Lex` keyword set with TC weighting scheme and the SVM method. They combine the textual features with current volatility using the early fusion method.

Tsai et al. (2014): similar to Wang et al. (2013), while they use the `LexExt` keyword set.

Evaluation Metrics As a common metric in volatility prediction, we use the r^2 metric (square of the correlation coefficient) for evaluation:

$$r^2 = \left(\frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \right)^2 \quad (7)$$

where $\hat{y}_i$ is the predicted value, y_i denotes the labels and $\bar{y}$, their mean. The r^2 metric indicates the proportion of variance in the labels explained by the prediction. The measure is close to 1 when the predicted values can explain a large proportion of the variability in the labels and 0 when it fails to explain the labels' variabilities. An alternative metric, used in previous studies (Wang et al., 2013; Tsai and Wang, 2014; Kogan et al., 2009) is Mean Squared Error $MSE = \sum_i (\hat{y}_i - y_i)^2/n$. However, especially when comparing models, applied on different test sets (e.g. performance of first quartile with second quartile), r^2 has better interpretability since it is independent of the scale of y. We use r^2 in all the experiments while the MSE measure is reported only when the models are evaluated on the same test set.

6 Experiments and Results

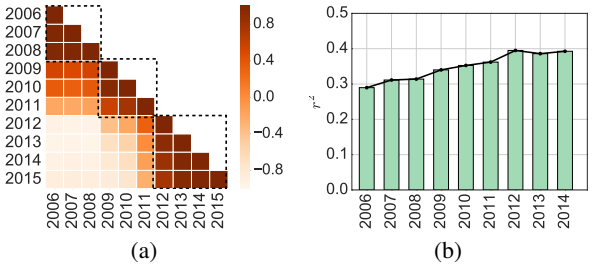
In this section, first we analyse the contents of the reports, followed by studying our sentiment analysis methods for volatility prediction. Finally, we investigate the effect of sentiment analysis of the reports in different industry sectors.

6.1 Content Analysis of 10-K Reports

Let us start our experiment with observing changes in the feature vectors of the reports over the years. To compare them, we use the state-of-the-art sentiment analysis method, introduced by Tsai and Wang (2014). We first represent the feature vector of each year by calculating the centroid

[2] https://www.sec.gov

[3] The complete dataset is available in http://ifs.tuwien.ac.at/~admire/financialvolatility



(a) (b)

Figure 1: (a) Cosine similarity between the centroid vectors of the years. (b) Volatility prediction performance when using reports from the specified year to 2015

(element-wise mean) of the feature vectors of all reports published that year and then calculate the Cosine similarity of each pair of centroid vectors, for the years 2006–2015.

Figure 1a shows the similarity heat-map for each pair of the years. We observe a high similarity between three ranges of years: 2006–2008, 2009–2011, and 2012–2015. These considerable differences between the centroid reports in years across these three groups hints at probable issues when using the data of the older years for the more recent ones.

To validate this, we apply 5-fold cross validation, first on all the data (2006–2015), and then on smaller sets by dropping the oldest year i.e. the next subsets use the reports 2007–2015, 2008–2015 and so forth. The results of the r^2 measure are shown in Figure 1b. We observe that by dropping the oldest years one by one (from left to right in the figure), the performance starts improving. We argue that this improvement is due to the reduction of noise in data, noise caused by conceptual drifts in the reports as also mentioned by Dyer et al. (2016). In fact, although in machine learning in general using more data results in better generalization of the model and therefore better prediction, the reports of the older years introduce noise.

As shown, the most coherent and largest data consists of the subset of the reports published between 2012 to 2015. This subset is also the most recent cluster and presumably more similar to the future reports. Therefore, in the following, we only use this subset, which consists of 3892 reports, belonging to 1323 companies.

Table 1: Performance of sentiment analysis methods for the first year.

Component	Method	Text (r^2)	Text (MSE)	Text+Market (r^2)	Text+Market (MSE)
Weighting Schema (+Stacking)	$\widehat{BM25}$	**0.439**	**0.132**	**0.527**	**0.111**
	$BM25$	0.433	0.136	0.523	0.114
	$\widehat{TC}$	0.427	0.136	0.517	0.115
	TC	0.425	0.137	0.521	0.114
	$\widehat{TFIDF}$	0.301	0.166	0.502	0.118
	$TFIDF$	0.264	0.189	0.497	0.119
	$\widehat{TF}$	0.218	0.190	0.495	0.120
	TF	0.233	0.200	0.495	0.120
Feature Fusion ($+\widehat{BM25}$)	**Stacking**	-	-	**0.527**	**0.111**
	MKL	-	-	0.488	0.126
	Early Fusion	-	-	0.473	0.125

Table 2: Performance of the methods using 5-fold cross validation.

Method		(r^2)	(MSE)
Text	GARCH	0.280	0.170
	Wang (2013)	0.345	0.154
	Tsai (2014)	0.395	0.142
	Our method	**0.439**	**0.132**
Text+Market	Market	0.485	0.122
	Wang (2013)	0.499	0.118
	Tsai (2014)	0.484	0.122
	Our method	**0.527**	**0.111**

6.2 Volatility Prediction

Given the dataset of the 2012–2015 reports, we try all combinations of different term weighting schemes using the `LexExt` keyword set. All weighting schemes are then combined with the market features with the introduced fusion methods. The prediction is done with 5-fold cross validation. The averages of the results of the first four quartiles (first year) are reported in Table 1. To make showing the results tractable, we use the best fusion (stacking) for the weighting schemes and the best scheme ($\widehat{BM25}$) for fusions.

Regarding the weighting schemes, $\widehat{BM25}$, $BM25$, and $\widehat{TC}$ show the best results. In general, the extended schemes (with hat) improve upon their normal forms. For the feature fusion methods, stacking outperforms the other approaches in both evaluation measures. MKL however has better performance than early fusion while it has the highest computational complexity among the methods. Based on these results, as our best performing approach in the remainder of the paper, we use $\widehat{BM25}$ (with `LexExt` set), reduced to 400 dimensions and stacking as the fusion method. Table 2 summarizes the results of our best per-

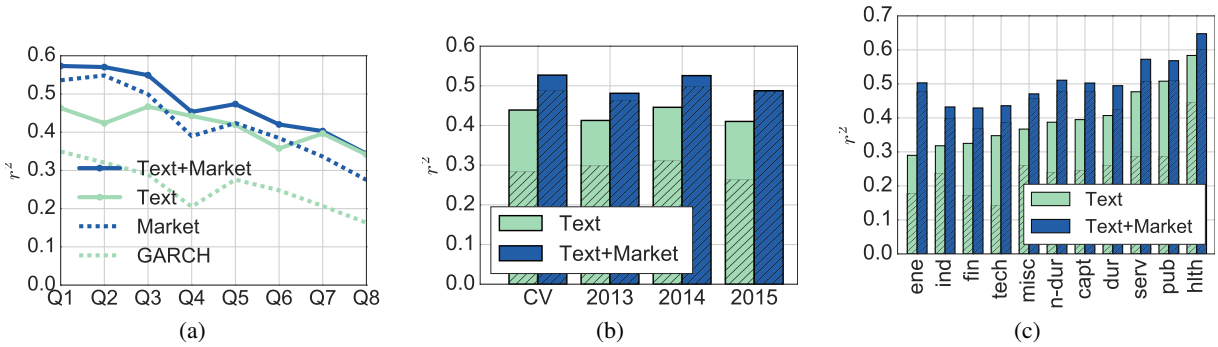


Figure 2: (a) Performance of our approach on 8 quartiles using the Text and Text+Market feature sets. The dashed lines show the market-based baselines. (b) Performance of volatility prediction of each year given the past data. The hashed areas show corresponding baselines. (c) Performance per sector. Abbreviations are defined in Section 4.2

forming method compared with previously existing methods. Our method outperforms all state-of-the-art methods both when using textual features only as well as a combination of textual and market features.

Let us now take a closer look on the changes in the performance of the prediction in time. The results of 5-fold cross validation for both tasks on the dataset of the reports, published between 2012–2015 are shown in Figure 2a. The X-axes show eight quartiles after the publication date of the report. For comparison, the GARCH and only market features are depicted with dashed lines.

As shown, the performance of the GARCH method as well as that using only market features (Market) decrease faster in the later quartiles since the historical prices used for prediction become less relevant as time goes by. Using only text features (Text), we see a roughly similar performance between the first four quartiles (first year), while the performance, in general, slightly decreases in the second year. By combining the textual and market features (Text+Market), we see a consistent improvement in comparison to each of them alone. In comparison to using only market features, the combination of the features shows more stable results in the later quartiles. These results support the informativeness of the 10-K reports to more effectively foreseen volatility in long-term windows.

While the above experiments are based on cross-validation, for the sake of completeness it is noteworthy to consider the scenarios of real-world applications where the future prediction is based on past data. We therefore design three experiments by considering the reports published in 2013, 2014, and 2015 as test set and the reports published before each year as training set (only 2012, 2012–2013, and 2012–2014 respectively). The results of predicting the reports of each year together with the cross validation scenario (CV) are shown in Figure 2b. While the performance becomes slightly worse in the target years 2013 and 2015, in general the combination of textual and market features can explain approximately half of volatility in the financial system.

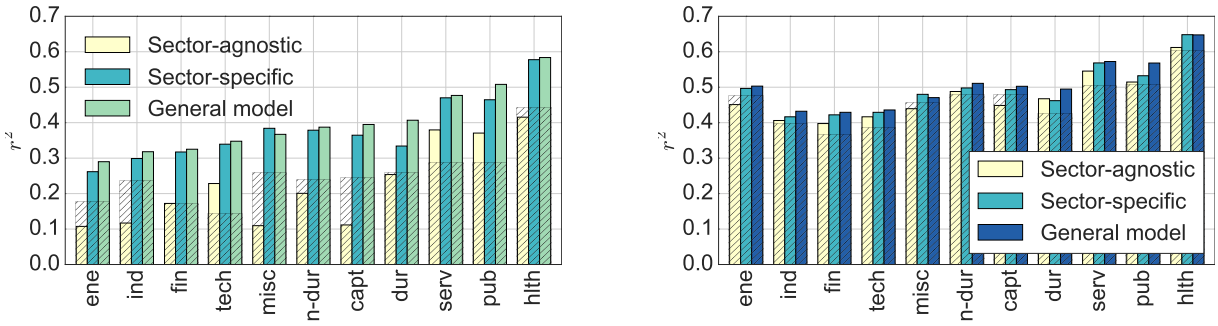
6.3 Sectors

Corporations in the same sector share not only similar products or services but also risks and instability factors. Considering the sentiment of the financial system as a homogeneous body may neglect the specific factors of each sector. We therefore set out to investigate the existence and nature of these differences.

We start by observing the prediction performance on different sectors: We use our method from the previous section, but split the test set across sectors and plot the results in Figure 2c. The hashed areas indicate the GARCH and Market baselines for the Text and Text+Market feature sets, respectively. We observe considerable differences between the performance of the sectors, especially when using only sentiment analysis methods (i.e. only text features).

Given these differences and also the probable similarities between the risk factors of the reports in the same sector, a question immediately arises: can training different models for different sectors improve the performance of prediction?

To answer it, for each sector, we train a model using only the subset of the reports in that sec-



(a) Text

(b) Text+Market

Figure 3: Results when retraining on sector-specific subsets versus the general model and versus subsets of the same size but sector-agnostic. The hashed area in (a) indicates the GARCH and in (b) the Market baseline.

Table 3: Number of reports per sectors

ene	ind	hlth	fin	tech	pub
187	160	305	847	408	217
n-dur	dur	capt	serv	misc	
151	115	255	639	153	

tor and use 5-fold validation to observe performance. We refer to these models as sector-specific in contrast to the general model, trained on all the data. Figures 3a and 3b compare their results: we can see that the sector-specific bars are lower than the general model ones. This is to some extent surprising, as one would expect that domain-specific training would improve the performance of sentiment analysis in text. However, we need to consider the size of the training set. By training on each sector we have reduced the size of our training sets to those reported in Table 3. To verify the effect of the size of training data, we train a sector-agnostic model for each sector. Each sector-agnostic model is trained by random sampling of a training set of the same size as the set available for its sector from all the reports, but evaluated–similar to sector-specific models–on the test set of the sector. Figures 3a and 3b also plot the results of the sector-agnostic models.

The large performance differences between sector-agnostic and -specific show the existence of particular risk factors in each sector and their importance. Results also confirm the hypothesis that the data for training in each sector is simply too small, and as additional data is accumulated, we can further improve on the results by training on different sectors independently.

We continue by examining some examples of essential terms in sectors. To address this, we have to train a linear regression method on all the reports of each sector, without using any dimensionality reduction. Linear regression without dimensionality reduction has the benefit of interpretability: the coefficient of each feature (i.e. term in the lexicon) can be seen as its importance with regards to volatility prediction. After training, we observe that some keywords e.g. *crisis*, or *delist* constantly have high coefficient values in the sector-specific as well as general model. However, some keywords are particularly weighted high in specific-sector models.

For instance, the keyword *fire* has a high coefficient in the energy sector, but very low in the others. The reason is due to the problem of ambiguity i.e. in the energy sector, *fire* is widely used to refer to *explosion* e.g. 'fire and explosion hazards' while in the lexicon, it is stemmed from *firing* and *fired*: the act of dismissing from a job. This later sense of word is however weighted as a low risk-sensitive keyword in the other sectors. Such an ambiguity can indeed be mitigated by sector-specific models since the variety of the words' senses are more restricted inside each sector. Another example is an interesting observation on the word *beneficial*. The word is introduced as a positive sentiment in the lexicon while it gains highly negative sentiments in some sectors (health care, and basic industries). Investigating in the reports, we observe the broad use of the expression 'beneficial owner' which is normally followed by risk-full sentences since the beneficial owners can potentially influence shareholders' decision power.

7 Conclusion

In this work, we studied the sentiment of recent 10-K annual disclosures of companies in stock markets for forecasting volatility. Our bag-of-words sentiment analysis approach benefits from state-of-the-art models in information retrieval which use word embeddings to extend the weight of the terms to the similar terms in the document. Additionally, we explored fusion methods to combine the text features with factual market features, achieved from historical prices i.e. GARCH prediction model, and current volatility. In both cases, our approach outperforms state-of-the-art volatility prediction methods with 10-K reports and demonstrates the effectiveness of sentiment analysis in long-term volatility forecasting.

In addition, we studied the characteristics of each individual sector with regard to risk-sensitive terms. Our analysis shows that reports in same sectors considerably share particular risk and instability factors. However, despite expectations, training different models on different sectors does not improve performance compared to the general model. We traced this to the size of the available data in each sector, and show that there are still benefits in considering sectors, which could be further explored in the future as more data becomes available.

8 Acknowledgement

This paper follows work produced during the Young Scientists Summer Program (YSSP) 2016 at the International Institute for Applied Systems Analysis (IIASA) in Laxenburg, Austria. This work is funded by: Self-Optimizer (FFG 852624) in the EUROSTARS programme, funded by EUREKA, the BMWFW and the European Union, ADMIRE (P 25905-N23) by FWF, and the Austrian Ministry for Science, Research and Economy. Thanks to Joni Sayeler and Linus Wretblad for their contributions in the SelfOptimizer project.

References

Tim Bollerslev. 1986. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics* 31(3):307–327.

John L Campbell, Hsinchun Chen, Dan S Dhaliwal, Hsin-min Lu, and Logan B Steele. 2014. The information content of mandatory risk factor disclosures in corporate filings. *Review of Accounting Studies* 19(1):396–455.

Charlotte Christiansen, Maik Schmeling, and Andreas Schrimpf. 2012. A comprehensive look at financial volatility prediction by economic variables. *Journal of Applied Econometrics* 27(6):956–977.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*. pages 2327–2333.

Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex Smola, Vladimir Vapnik, et al. 1997. Support vector regression machines. *Advances in neural information processing systems* 9:155–161.

Travis Dyer, Mark H Lang, and Lorien Stice-Lawrence. 2016. The ever-expanding 10-k: Why are 10-ks getting so much longer (and does it matter)? *Available at SSRN 2741682* .

Robert F Engle. 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the Econometric Society* pages 987–1007.

Mehmet Gönen and Ethem Alpaydın. 2011. Multiple kernel learning algorithms. *Journal of Machine Learning Research* 12(Jul):2211–2268.

Siavash Kazemian, Shunan Zhao, and Gerald Penn. 2014. Evaluating sentiment analysis evaluation: A case study in securities trading. *Proceedings of the Conference of the Association for Computational Linguistics (ACL)* page 119.

Shimon Kogan, Dimitry Levin, Bryan R Routledge, Jacob S Sagi, and Noah A Smith. 2009. Predicting risk from financial reports with regression. In *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics*. pages 272–280.

Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. 2004. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research* 5(Jan):27–72.

Feng Li. 2010. The information content of forward-looking statements in corporate filings–a naïve bayesian machine learning approach. *Journal of Accounting Research* 48(5):1049–1102.

Hongquan Li and Yongmiao Hong. 2011. Financial volatility forecasting with range-based autoregressive volatility model. *Finance Research Letters* 8(2):69–76.

Shouwei Liu and Yiu Kuen Tse. 2013. Estimation of monthly volatility: An empirical comparison of realized volatility, garch and acd-icv methods. *Research Collection School Of Economics* .

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance* 66(1):35–65.

Ronny Luss and Alexandre d'Aspremont. 2015. Predicting abnormal returns from news using text classification. *Quantitative Finance* 15(6):999–1012.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Rodrigo Moraes, JoãO Francisco Valiati, and Wilson P GaviãO Neto. 2013. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications* 40(2):621–633.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Topic modeling based sentiment analysis on social media for stock market prediction. In *ACL*.

William Stafford Noble et al. 2004. Support vector machine applications in computational biology. *Kernel methods in computational biology* pages 71–92.

Clemens Nopp and Allan Hanbury. 2015. Detecting risks in the banking system by sentiment analysis. *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP)* pages 591–600.

Navid Rekabsaz, Mihai Lupu, and Allan Hanbury. 2016a. Uncertainty in neural network word embedding: Exploration of threshold for similarity. *arXiv preprint arXiv:1606.06086* .

Navid Rekabsaz, Mihai Lupu, Allan Hanbury, and Guido Zuccon. 2016b. Generalizing translation models in the probabilistic relevance framework. *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)* .

Navid Rekabsaz, Mihai Lupu, Allan Hanbury, and Guido Zuccon. 2017. Exploration of a threshold for similarity based on uncertainty in word embedding. In *European Conference on IR Research (ECIR)*.

Martin Stražar and Tomaž Curk. 2016. Learning the kernel matrix via predictive low-rank approximations. *arXiv preprint arXiv:1601.04366* .

Ming-Feng Tsai and Chuan-Ju Wang. 2014. Financial keyword expansion via continuous word vector representations. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP)*. pages 1453–1458.

Chuan-Ju Wang, Ming-Feng Tsai, Tse Liu, and Chin-Ting Chang. 2013. Financial sentiment analysis for risk prediction. In *Proceedings of the Joint Conference on Natural Language Processing (IJCNLP)*. pages 802–808.

William Yang Wang and Zhenhao Hua. 2014. A semiparametric gaussian copula regression model for predicting financial risks from earnings calls. In *ACL*.

David H Wolpert. 1992. Stacked generalization. *Neural networks* 5(2):241–259.

Boyi Xie, Rebecca J Passonneau, and Leon Wu. 2013. Semantic Frames to Predict Stock Price Movement. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

CANE: Context-Aware Network Embedding for Relation Modeling

Cunchao Tu[1,2*], **Han Liu**[3*], **Zhiyuan Liu**[1,2†], **Maosong Sun**[1,2]
[1]Department of Computer Science and Technology, State Key Lab on Intelligent Technology and Systems, National Lab for Information Science and Technology, Tsinghua University, China
[2]Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University, China
[3]Northeastern University, China

Abstract

Network embedding (NE) is playing a critical role in network analysis, due to its ability to represent vertices with efficient low-dimensional embedding vectors. However, existing NE models aim to learn a fixed context-free embedding for each vertex and neglect the diverse roles when interacting with other vertices. In this paper, we assume that one vertex usually shows different aspects when interacting with different neighbor vertices, and should own different embeddings respectively. Therefore, we present Context-Aware Network Embedding (CANE), a novel NE model to address this issue. CANE learns context-aware embeddings for vertices with mutual attention mechanism and is expected to model the semantic relationships between vertices more precisely. In experiments, we compare our model with existing NE models on three real-world datasets. Experimental results show that CANE achieves significant improvement than state-of-the-art methods on link prediction and comparable performance on vertex classification. The source code and datasets can be obtained from https://github.com/thunlp/CANE.

1 Introduction

Network embedding (NE), i.e., network representation learning (NRL), aims to map vertices of a network into a low-dimensional space according to their structural roles in the network. NE provides an efficient and effective way to represent and manage large-scale networks, alleviating the computation and sparsity issues of conventional symbol-based representations. Hence, NE is attracting many research interests in recent years (Perozzi et al., 2014; Tang et al., 2015; Grover and Leskovec, 2016), and achieves promising performance on many network analysis tasks including link prediction, vertex classification, and community detection.

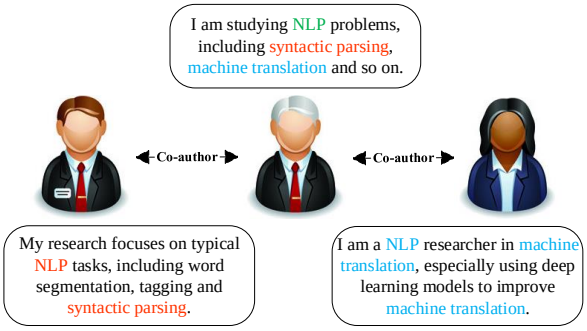


Figure 1: Example of a text-based information network. (Red, blue and green fonts represent concerns of the left user, right user and both respectively.)

In real-world social networks, it is intuitive that one vertex may demonstrate various aspects when interacting with different neighbor vertices. For example, a researcher usually collaborates with various partners on diverse research topics (as illustrated in Fig. 1), a social-media user contacts with various friends sharing distinct interests, and a web page links to multiple pages for different purposes. However, most existing NE methods only arrange one single embedding vector to each vertex, and give rise to the following two invertible issues: (1) These methods cannot flexibly cope with the aspect transition of a vertex when interacting with different neighbors. (2) In these models, a vertex tends to force the embeddings of its

*Indicates equal contribution
†Corresponding Author: Z. Liu (liuzy@tsinghua.edu.cn)

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1722–1731
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1158

neighbors close to each other, which may be not the case all the time. For example, the left user and right user in Fig. 1, share less common interests, but are learned to be close to each other since they both link to the middle person. This will accordingly make vertex embeddings indiscriminative.

To address these issues, we aim to propose a **C**ontext-**A**ware **N**etwork **E**mbedding (CANE) framework for modeling relationships between vertices precisely. More specifically, we present CANE on information networks, where each vertex also contains rich external information such as text, labels or other meta-data, and the significance of context is more critical for NE in this scenario. Without loss of generality, we implement CANE on text-based information networks in this paper, which can easily extend to other types of information networks.

In conventional NE models, each vertex is represented as a static embedding vector, denoted as **context-free embedding**. On the contrary, CANE assigns dynamic embeddings to a vertex according to different neighbors it interacts with, named as **context-aware embedding**. Take a vertex u and its neighbor vertex v for example. The context-free embedding of u remains unchanged when interacting with different neighbors. On the contrary, the context-aware embedding of u is dynamic when confronting different neighbors.

When u interacting with v, their context embeddings concerning each other are derived from their text information, S_u and S_v respectively. For each vertex, we can easily use neural models, such as convolutional neural networks (Blunsom et al., 2014; Johnson and Zhang, 2014; Kim, 2014) and recurrent neural networks (Kiros et al., 2015; Tai et al., 2015), to build **context-free** text-based embedding. In order to realize **context-aware** text-based embeddings, we introduce the selective attention scheme and build **mutual attention** between u and v into these neural models. The mutual attention is expected to guide neural models to emphasize those words that are focused by its neighbor vertices and eventually obtain context-aware embeddings.

Both context-free embeddings and context-aware embeddings of each vertex can be efficiently learned together via concatenation using existing NE methods such as DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015) and node2vec (Grover and Leskovec, 2016).

We conduct experiments on three real-world datasets of different areas. Experimental results on link prediction reveal the effectiveness of our framework as compared to other state-of-the-art methods. The results suggest that context-aware embeddings are critical for network analysis, in particular for those tasks concerning about complicated interactions between vertices such as link prediction. We also explore the performance of our framework via vertex classification and case studies, which again confirms the flexibility and superiority of our models.

2 Related Work

With the rapid growth of large-scale social networks, network embedding, i.e. network representation learning has been proposed as a critical technique for network analysis tasks.

In recent years, there have been a large number of NE models proposed to learn efficient vertex embeddings (Tang and Liu, 2009; Cao et al., 2015; Wang et al., 2016; Tu et al., 2016a). For example, DeepWalk (Perozzi et al., 2014) performs random walks over networks and introduces an efficient word representation learning model, Skip-Gram (Mikolov et al., 2013a), to learn network embeddings. LINE (Tang et al., 2015) optimizes the joint and conditional probabilities of edges in large-scale networks to learn vertex representations. Node2vec (Grover and Leskovec, 2016) modifies the random walk strategy in DeepWalk into biased random walks to explore the network structure more efficiently. Nevertheless, most of these NE models only encode the structural information into vertex embeddings, without considering heterogeneous information accompanied with vertices in real-world social networks.

To address this issue, researchers make great efforts to incorporate heterogeneous information into conventional NE models. For instance, Yang et al. (2015) present text-associated DeepWalk (TADW) to improve matrix factorization based DeepWalk with text information. Tu et al. (2016b) propose max-margin DeepWalk (MMDW) to learn discriminative network representations by utilizing labeling information of vertices. Chen et al. (2016) introduce group-enhanced network embedding (GENE) to integrate existing group information in NE. Sun et al. (2016) regard text content as a special kind

of vertices, and propose context-enhanced network embedding (CENE) through leveraging both structural and textural information to learn network embeddings.

To the best of our knowledge, all existing NE models focus on learning context-free embeddings, but ignore the diverse roles when a vertex interacts with others. In contrast, we assume that a vertex has different embeddings according to which vertex it interacts with, and propose CANE to learn context-aware vertex embeddings.

3 Problem Formulation

We first give basic notations and definitions in this work. Suppose there is an information network $G = (V, E, T)$, where V is the set of vertices, $E \subseteq V \times V$ are edges between vertices, and T denotes the text information of vertices. Each edge $e_{u,v} \in E$ represents the relationship between two vertices (u, v), with an associated weight $w_{u,v}$. Here, the text information of a specific vertex $v \in V$ is represented as a word sequence $S_v = (w_1, w_2, \ldots, w_{n_v})$, where $n_v = |S_v|$. NRL aims to learn a low-dimensional embedding $\mathbf{v} \in \mathbb{R}^d$ for each vertex $v \in V$ according to its network structure and associated information, e.g. text and labels. Note that, $d \ll |V|$ is the dimension of representation space.

Definition 1. Context-free Embeddings: Conventional NRL models learn context-free embedding for each vertex. It means the embedding of a vertex is fixed and won't change with respect to its context information (i.e., another vertex it interacts with).

Definition 2. Context-aware Embeddings: Different from existing NRL models that learn context-free embeddings, CANE learns various embeddings for a vertex according to its different contexts. Specifically, for an edge $e_{u,v}$, CANE learns context-aware embeddings $\mathbf{v}_{(u)}$ and $\mathbf{u}_{(v)}$.

4 The Method

4.1 Overall Framework

To take full use of both network structure and associated text information, we propose two types of embeddings for a vertex v, i.e., structure-based embedding $\mathbf{v}^s$ and text-based embedding $\mathbf{v}^t$. Structure-based embedding can capture the information in the network structure, while text-based embedding can capture the textual meanings lying in the associated text information. With these embeddings, we can simply concatenate them and obtain the vertex embeddings as $\mathbf{v} = \mathbf{v}^s \oplus \mathbf{v}^t$, where $\oplus$ indicates the concatenation operation. Note that, the text-based embedding $\mathbf{v}^t$ can be either context-free or context-aware, which will be introduced detailedly in section 4.4 and 4.5 respectively. When $\mathbf{v}^t$ is context-aware, the overall vertex embeddings $\mathbf{v}$ will be context-aware as well.

With above definitions, CANE aims to maximize the overall objective of edges as follows:

$$\mathcal{L} = \sum_{e \in E} L(e). \tag{1}$$

Here, the objective of each edge $L(e)$ consists of two parts as follows:

$$L(e) = L_s(e) + L_t(e), \tag{2}$$

where $L_s(e)$ denotes the structure-based objective and $L_t(e)$ represents the text-based objective.

In the following part, we give the detailed introduction to the two objectives respectively.

4.2 Structure-based Objective

Without loss of generality, we assume the network is directed, as an undirected edge can be considered as two directed edges with opposite directions and equal weights.

Thus, the structure-based objective aims to measure the log-likelihood of a directed edge using the structure-based embeddings as

$$L_s(e) = w_{u,v} \log p(\mathbf{v}^s | \mathbf{u}^s). \tag{3}$$

Following LINE (Tang et al., 2015), we define the conditional probability of v generated by u in Eq. (3) as

$$p(\mathbf{v}^s | \mathbf{u}^s) = \frac{\exp(\mathbf{u}^s \cdot \mathbf{v}^s)}{\sum_{z \in V} \exp(\mathbf{u}^s \cdot \mathbf{z}^s)}. \tag{4}$$

4.3 Text-based Objective

Vertices in real-world social networks usually accompany with associated text information. Therefore, we propose the text-based objective to take advantage of these text information, as well as learn text-based embeddings for vertices.

The text-based objective $L_t(e)$ can be defined with various measurements. To be compatible with $L_s(e)$, we define $L_t(e)$ as follows:

$$L_t(e) = \alpha \cdot L_{tt}(e) + \beta \cdot L_{ts}(e) + \gamma \cdot L_{st}(e), \tag{5}$$

where α, β and γ control the weights of various parts, and

$$L_{tt}(e) = w_{u,v} \log p(\mathbf{v}^t|\mathbf{u}^t),$$
$$L_{ts}(e) = w_{u,v} \log p(\mathbf{v}^t|\mathbf{u}^s), \quad (6)$$
$$L_{st}(e) = w_{u,v} \log p(\mathbf{v}^s|\mathbf{u}^t).$$

The conditional probabilities in Eq. (6) map the two types of vertex embeddings into the same representation space, but do not enforce them to be identical for the consideration of their own characteristics. Similarly, we employ softmax function for calculating the probabilities, as in Eq. (4).

The structure-based embeddings are regarded as parameters, the same as in conventional NE models. But for text-based embeddings, we intend to obtain them from associated text information of vertices. Besides, the text-based embeddings can be obtained either in context-free ways or context-aware ones. In the following sections, we will give detailed introduction respectively.

4.4 Context-Free Text Embedding

There has been a variety of neural models to obtain text embeddings from a word sequence, such as convolutional neural networks (CNN) (Blunsom et al., 2014; Johnson and Zhang, 2014; Kim, 2014) and recurrent neural networks (RNN) (Kiros et al., 2015; Tai et al., 2015).

In this work, we investigate different neural networks for text modeling, including CNN, Bidirectional RNN (Schuster and Paliwal, 1997) and GRU (Cho et al., 2014), and employ the best performed CNN, which can capture the local semantic dependency among words.

Taking the word sequence of a vertex as input, CNN obtains the text-based embedding through three layers, i.e. looking-up, convolution and pooling.

Looking-up. Given a word sequence $S = (w_1, w_2, \ldots, w_n)$, the looking-up layer transforms each word $w_i \in S$ into its corresponding word embedding $\mathbf{w}_i \in \mathbb{R}^{d'}$ and obtains embedding sequence as $\mathbf{S} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n)$. Here, d' indicates the dimension of word embeddings.

Convolution. After looking-up, the convolution layer extracts local features of input embedding sequence $\mathbf{S}$. To be specific, it performs convolution operation over a sliding window of length l using a convolution matrix $\mathbf{C} \in \mathbb{R}^{d \times (l \times d')}$ as follows:

$$\mathbf{x}_i = \mathbf{C} \cdot \mathbf{S}_{i:i+l-1} + \mathbf{b}, \quad (7)$$

where $\mathbf{S}_{i:i+l-1}$ denotes the concatenation of word embeddings within the i-th window and $\mathbf{b}$ is the bias vector. Note that, we add zero padding vectors (Hu et al., 2014) at the edge of the sentence.

Max-pooling. To obtain the text embedding $\mathbf{v}^t$, we operate max-pooling and non-linear transformation over $\{\mathbf{x}_0^i, \ldots, \mathbf{x}_n^i\}$ as follows:

$$r_i = \tanh(\max(\mathbf{x}_0^i, \ldots, \mathbf{x}_n^i)), \quad (8)$$

At last, we encode the text information of a vertex with CNN and obtain its text-based embedding $\mathbf{v}^t = [r_1, \ldots, r_d]^T$. As $\mathbf{v}^t$ is irrelevant to the other vertices it interacts with, we name it as context-free text embedding.

4.5 Context-Aware Text Embedding

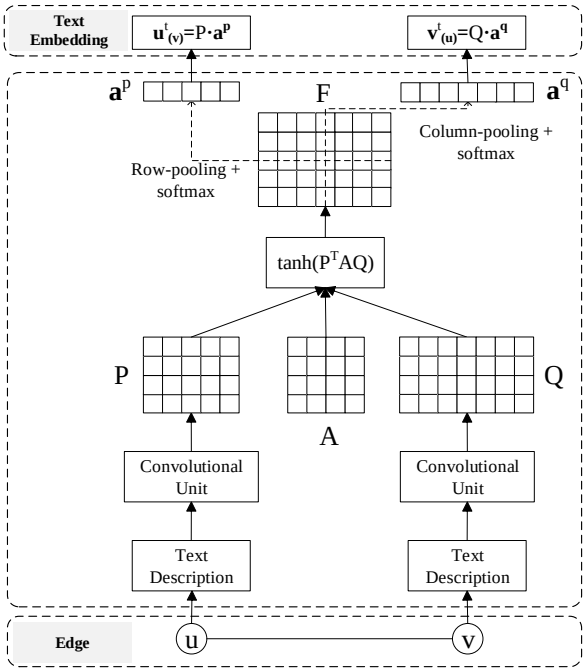


Figure 2: An illustration of context-aware text embedding.

As stated before, we assume that a specific vertex plays different roles when interacting with others vertices. In other words, each vertex should have its own points of focus about a specific vertex, which leads to its context-aware text embeddings.

To achieve this, we employ **mutual attention** to obtain context-aware text embedding. It enables the pooling layer in CNN to be aware of the vertex pair in an edge, in a way that text information from a vertex can directly affect the text embedding of the other vertex, and vice versa.

In Fig. 2, we give an illustration of the generating process of context-aware text embedding. Given an edge $e_{u,v}$ with two corresponding text sequences S_u and S_v, we can get the matrices $\mathbf{P} \in \mathbb{R}^{d\times m}$ and $\mathbf{Q} \in \mathbb{R}^{d\times n}$ through convolution layer. Here, m and n represent the lengths of S_u and S_v respectively. By introducing an attentive matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$, we compute the correlation matrix $\mathbf{F} \in \mathbb{R}^{m\times n}$ as follows:

$$\mathbf{F} = \tanh(\mathbf{P}^T\mathbf{A}\mathbf{Q}). \tag{9}$$

Note that, each element $\mathbf{F}_{i,j}$ in $\mathbf{F}$ represents the pair-wise correlation score between two hidden vectors, i.e., $\mathbf{P}_i$ and $\mathbf{Q}_j$.

After that, we conduct pooling operations along rows and columns of $\mathbf{F}$ to generate the importance vectors, named as row-pooling and column pooling respectively. According to our experiments, mean-pooling performs better than max-pooling. Thus, we employ mean-pooling operation as follows:

$$\begin{aligned} g_i^p &= \mathbf{mean}(\mathbf{F}_{i,1}, \ldots, \mathbf{F}_{i,n}), \\ g_i^q &= \mathbf{mean}(\mathbf{F}_{1,i}, \ldots, \mathbf{F}_{m,i}). \end{aligned} \tag{10}$$

The importance vectors of $\mathbf{P}$ and $\mathbf{Q}$ are obtained as $\mathbf{g}^p = [g_1^p, \ldots, g_m^p]^T$ and $\mathbf{g}^q = [g_1^q, \ldots, g_n^q]^T$.

Next, we employ softmax function to transform importance vectors $\mathbf{g}^p$ and $\mathbf{g}^q$ to attention vectors $\mathbf{a}^p$ and $\mathbf{a}^q$. For instance, the i-th element of $\mathbf{a}^p$ is formalized as follows:

$$a_i^p = \frac{\exp(g_i^p)}{\sum_{j\in[1,m]} \exp(g_j^p)}. \tag{11}$$

At last, the context-aware text embeddings of u and v are computed as

$$\begin{aligned} \mathbf{u}_{(v)}^t &= \mathbf{P}\mathbf{a}^p, \\ \mathbf{v}_{(u)}^t &= \mathbf{Q}\mathbf{a}^q. \end{aligned} \tag{12}$$

Now, given an edge (u, v), we can obtain the context-aware embeddings of vertices with their structure embeddings and context-aware text embeddings as $\mathbf{u}_{(v)} = \mathbf{u}^s \oplus \mathbf{u}_{(v)}^t$ and $\mathbf{v}_{(u)} = \mathbf{v}^s \oplus \mathbf{v}_{(u)}^t$.

4.6 Optimization of CANE

According to Eq. (3) and Eq. (6), CANE aims to maximize several conditional probabilities between $\mathbf{u} \in \{\mathbf{u}^s, \mathbf{u}_{(v)}^t\}$ and $\mathbf{v} \in \{\mathbf{v}^s, \mathbf{v}_{(u)}^t\}$. It is intuitive that optimizing the conditional probability using softmax function is computationally expensive. Thus, we employ negative sampling (Mikolov et al., 2013b) and transform the objective into the following form:

$$\log\sigma(\mathbf{u}^T\cdot\mathbf{v}) + \sum_{i=1}^{k} E_{z\sim P(v)}[\log\sigma(-\mathbf{u}^T\cdot\mathbf{z})], \tag{13}$$

where k is the number of negative samples and σ represents the sigmoid function. $P(v) \propto {d_v}^{3/4}$ denotes the distribution of vertices, where d_v is the out-degree of v.

Afterward, we employ Adam (Kingma and Ba, 2015) to optimize the transformed objective. Note that, CANE is exactly capable of zero-shot scenarios, by generating text embeddings of new vertices with well-trained CNN.

5 Experiments

To investigate the effectiveness of CANE on modeling relationships between vertices, we conduct experiments of link prediction on several real-world datasets. Besides, we also employ vertex classification to verify whether context-aware embeddings of a vertex can compose a high-quality context-free embedding in return.

5.1 Datasets

Datasets	**Cora**	**HepTh**	**Zhihu**
#Vertices	2,277	1,038	10,000
#Edges	5,214	1,990	43,894
#Labels	7	–	–

Table 1: Statistics of Datasets.

We select three real-world network datasets as follows:

Cora[1] is a typical paper citation network constructed by (McCallum et al., 2000). After filtering out papers without text information, there are 2,277 machine learning papers in this network, which are divided into 7 categories.

HepTh[2] (High Energy Physics Theory) is another citation network from arXiv[3] released by (Leskovec et al., 2005). We filter out papers without abstract information and retain 1,038 papers at last.

[1] https://people.cs.umass.edu/∼mccallum/data.html
[2] https://snap.stanford.edu/data/cit-HepTh.html
[3] https://arxiv.org/

%Training edges	15%	25%	35%	45%	55%	65%	75%	85%	95%
MMB	54.7	57.1	59.5	61.9	64.9	67.8	71.1	72.6	75.9
DeepWalk	56.0	63.0	70.2	75.5	80.1	85.2	85.3	87.8	90.3
LINE	55.0	58.6	66.4	73.0	77.6	82.8	85.6	88.4	89.3
node2vec	55.9	62.4	66.1	75.0	78.7	81.6	85.9	87.3	88.2
Naive Combination	72.7	82.0	84.9	87.0	88.7	91.9	92.4	93.9	94.0
TADW	86.6	88.2	90.2	90.8	90.0	93.0	91.0	93.4	92.7
CENE	72.1	86.5	84.6	88.1	89.4	89.2	93.9	95.0	95.9
CANE (text only)	78.0	80.5	83.9	86.3	89.3	91.4	91.8	91.4	93.3
CANE (w/o attention)	85.8	90.5	91.6	93.2	93.9	94.6	95.4	95.1	95.5
CANE	**86.8**	**91.5**	**92.2**	**93.9**	**94.6**	**94.9**	**95.6**	**96.6**	**97.7**

Table 2: AUC values on Cora. ($\alpha = 1.0, \beta = 0.3, \gamma = 0.3$)

Zhihu[4] is the largest online Q&A website in China. Users follow each other and answer questions on this site. We randomly crawl $10,000$ active users from Zhihu, and take the descriptions of their concerned topics as text information.

The detailed statistics are listed in Table 1.

5.2 Baselines

We employ the following methods as baselines:

Structure-only:

MMB (Airoldi et al., 2008) (Mixed Membership Stochastic Blockmodel) is a conventional graphical model of relational data. It allows each vertex to randomly select a different "topic" when forming an edge.

DeepWalk (Perozzi et al., 2014) performs random walks over networks and employ Skip-Gram model (Mikolov et al., 2013a) to learn vertex embeddings.

LINE (Tang et al., 2015) learns vertex embeddings in large-scale networks using first-order and second-order proximities.

Node2vec (Grover and Leskovec, 2016) proposes a biased random walk algorithm based on DeepWalk to explore neighborhood architecture more efficiently.

Structure and Text:

Naive Combination: We simply concatenate the best-performed structure-based embeddings with CNN based embeddings to represent the vertices.

TADW (Yang et al., 2015) employs matrix factorization to incorporate text features of vertices into network embeddings.

CENE (Sun et al., 2016) leverages both structure and textural information by regarding text content as a special kind of vertices, and optimizes the probabilities of heterogeneous links.

[4]https://www.zhihu.com/

5.3 Evaluation Metrics and Experiment Settings

For link prediction, we adopt a standard evaluation metric **AUC** (Hanley and McNeil, 1982), which represents the probability that vertices in a random unobserved link are more similar than those in a random nonexistent link.

For vertex classification, we employ L2-regularized logistic regression (L2R-LR) (Fan et al., 2008) to train classifiers, and evaluate the classification accuracies of various methods.

To be fair, we set the embedding dimension to 200 for all methods. In LINE, we set the number of negative samples to 5; we learn the 100 dimensional first-order and second-order embeddings respectively, and concatenate them to form the 200 dimensional embeddings. In node2vec, we employ grid search and select the best-performed hyper-parameters for training. We also apply grid search to set the hyper-parameters α, β and γ in CANE. Besides, we set the number of negative samples k to 1 in CANE to speed up the training process. To demonstrate the effectiveness of considering attention mechanism and two types of objectives in Eqs. (3) and (6), we design three versions of CANE for evaluation, i.e., CANE with text only, CANE without attention and CANE.

5.4 Link Prediction

As shown in Table 2, Table 3 and Table 4, we evaluate the AUC values while removing different ratios of edges on Cora, HepTh and Zhihu respectively. Note that, when we only keep 5% edges for training, most vertices are isolated, which results in the poor and meaningless performance of all the methods. Thus, we omit the results under this training ratio. From these tables, we have the following observations:

%Training edges	15%	25%	35%	45%	55%	65%	75%	85%	95%
MMB	54.6	57.9	57.3	61.6	66.2	68.4	73.6	76.0	80.3
DeepWalk	55.2	66.0	70.0	75.7	81.3	83.3	87.6	88.9	88.0
LINE	53.7	60.4	66.5	73.9	78.5	83.8	87.5	87.7	87.6
node2vec	57.1	63.6	69.9	76.2	84.3	87.3	88.4	89.2	89.2
Naive Combination	78.7	82.1	84.7	88.7	88.7	91.8	92.1	92.0	92.7
TADW	87.0	89.5	91.8	90.8	91.1	92.6	93.5	91.9	91.7
CENE	86.2	84.6	89.8	91.2	92.3	91.8	93.2	92.9	93.2
CANE (text only)	83.8	85.2	87.3	88.9	91.1	91.2	91.8	93.1	93.5
CANE (w/o attention)	84.5	89.3	89.2	91.6	91.1	91.8	92.3	92.5	93.6
CANE	**90.0**	**91.2**	**92.0**	**93.0**	**94.2**	**94.6**	**95.4**	**95.7**	**96.3**

Table 3: AUC values on HepTh. ($\alpha = 0.7, \beta = 0.2, \gamma = 0.2$)

%Training edges	15%	25%	35%	45%	55%	65%	75%	85%	95%
MMB	51.0	51.5	53.7	58.6	61.6	66.1	68.8	68.9	72.4
DeepWalk	56.6	58.1	60.1	60.0	61.8	61.9	63.3	63.7	67.8
LINE	52.3	55.9	59.9	60.9	64.3	66.0	67.7	69.3	71.1
node2vec	54.2	57.1	57.3	58.3	58.7	62.5	66.2	67.6	68.5
Naive Combination	55.1	56.7	58.9	62.6	64.4	68.7	68.9	69.0	71.5
TADW	52.3	54.2	55.6	57.3	60.8	62.4	65.2	63.8	69.0
CENE	56.2	57.4	60.3	63.0	66.3	66.0	70.2	69.8	73.8
CANE (text only)	55.6	56.9	57.3	61.6	63.6	67.0	68.5	70.4	73.5
CANE (w/o attention)	56.7	59.1	60.9	64.0	66.1	68.9	69.8	71.0	74.3
CANE	**56.8**	**59.3**	**62.9**	**64.5**	**68.9**	**70.4**	**71.4**	**73.6**	**75.4**

Table 4: AUC values on Zhihu. ($\alpha = 1.0, \beta = 0.3, \gamma = 0.3$)

(1) Our proposed CANE consistently achieves significant improvement comparing to all the baselines on all different datasets and different training ratios. It indicates the effectiveness of CANE when applied to link prediction task, and verifies that CANE has the capability of modeling relationships between vertices precisely.

(2) What calls for special attention is that, both CENE and TADW exhibit unstable performance under various training ratios. Specifically, CENE performs poorly under small training ratios, because it reserves much more parameters (e.g., convolution kernels and word embeddings) than TADW, which need more data for training. Different from CENE, TADW performs much better under small training ratios, because DeepWalk based methods can explore the sparse network structure well through random walks even with limited edges. However, it achieves poor performance under large ones, as its simplicity and the limitation of bag-of-words assumption. On the contrary, CANE has a stable performance in various situations. It demonstrates the flexibility and robustness of CANE.

(3) By introducing attention mechanism, the learnt context-aware embeddings obtain considerable improvements than the ones without attention. It verifies our assumption that a specific vertex should play different roles when interacting with other vertices, and thus benefits the relevant link prediction task.

To summarize, all the above observations demonstrate that CANE can learn high-quality context-aware embeddings, which are conducive to estimating the relationship between vertices precisely. Moreover, the experimental results on link prediction task state the effectiveness and robustness of CANE.

5.5 Vertex Classification

In CANE, we obtain various embeddings of a vertex according to the vertex it connects to. It's intuitive that the obtained context-aware embeddings are naturally applicable to link prediction task. However, network analysis tasks, such as vertex classification and clustering, require a global embedding, rather than several context-aware embeddings for each vertex.

To demonstrate the capability of CANE to solve these issues, we generate the global embedding of a vertex u by simply averaging all the context-

aware embeddings as follows:

$$\mathbf{u} = \frac{1}{N} \sum_{(u,v)|(v,u)\in E} \mathbf{u}_{(v)},$$

where N indicates the number of context-aware embeddings of u.

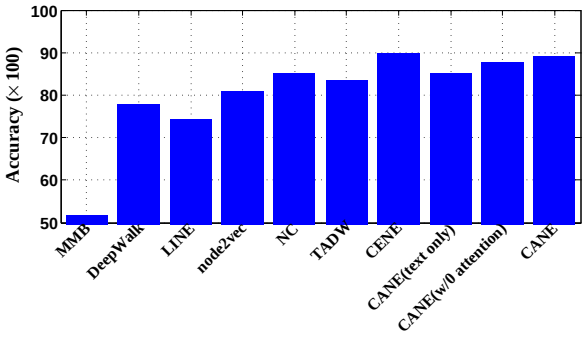


Figure 3: Vertex classification results on Cora.

With the generated global embeddings, we conduct 2-fold cross-validation and report the average accuracy of vertex classification on Cora. As shown in Fig. 3, we observe that:

(1) CANE achieves comparable performance with state-of-the-art model CENE. It states that the learnt context-aware embeddings can transform into high-quality context-free embeddings through simple average operation, which can be further employed to other network analysis tasks.

(2) With the introduction of mutual attention mechanism, CANE has an encouraging improvement than the one without attention, which is in accordance with the results of link prediction. It denotes that CANE is flexible to various network analysis tasks.

5.6 Case Study

To demonstrate the significance of **mutual attention** on selecting meaningful features from text information, we visualize the heat maps of two vertex pairs in Fig. 4. Note that, every word in this figure accompanies with various background colors. The stronger the background color is, the larger the weight of this word is. The weight of each word is calculated according to the attention weights as follows.

For each vertex pair, we can get the attention weight of each convolution window according to Eq. (11). To obtain the weights of words, we assign the attention weight to each word in this window, and add the attention weights of a word together as its final weight.

Edge #1: (A, B)

Machine Learning research making great progress many directions This article summarizes four directions discusses current open problems The four directions improving classification accuracy learning ensembles classifiers methods scaling supervised learning algorithms reinforcement learning learning complex stochastic models

The problem making optimal decisions uncertain conditions central Artificial Intelligence If state world known times world modeled Markov Decision Process MDP MDPs studied extensively many methods known determining optimal courses action policies The realistic case state information partially observable Partially Observable Markov Decision Processes POMDPs received much less attention The best exact algorithms problems inefficient space time We introduce Smooth Partially Observable Value Approximation SPOVA new approximation method quickly yield good approximations improve time This method combined reinforcement learning methods combination effective test cases

Edge #2: (A, C)

Machine Learning research making great progress many directions This article summarizes four directions discusses current open problems The four directions improving classification accuracy learning ensembles classifiers methods scaling supervised learning algorithms reinforcement learning learning complex stochastic models

In context machine learning examples paper deals problem estimating quality attributes without dependencies among Kira Rendell developed algorithm called RELIEF shown efficient estimating attributes Original RELIEF deal discrete continuous attributes limited twoclass problems In paper RELIEF analysed extended deal noisy incomplete multiclass data sets The extensions verified various artificial one well known realworld problem

Figure 4: Visualizations of mutual attention.

The proposed attention mechanism makes the relations between vertices explicit and interpretable. We select three connected vertices in Cora for example, denoted as A, B and C. From Fig. 4, we observe that, though there exists citation relations with identical paper A, paper B and C concern about different parts of A. The attention weights over A in edge #1 are assigned to "reinforcement learning". On the contrary, the weights in edge #2 are assigned to "machine learning'", "supervised learning algorithms" and "complex stochastic models". Moreover, all these key elements in A can find corresponding words in B and C. It's intuitive that these key elements give an exact explanation of the citation relations. The discovered significant correlations between vertex pairs reflect the effectiveness of mutual attention mechanism, as well as the capability of CANE for modeling relations precisely.

6 Conclusion and Future Work

In this paper, we propose the concept of Context-Aware Network Embedding (CANE) for the first

time, which aims to learn various context-aware embeddings for a vertex according to the neighbors it interacts with. Specifically, we implement CANE on text-based information networks with proposed mutual attention mechanism, and conduct experiments on several real-world information networks. Experimental results on link prediction demonstrate that CANE is effective for modeling the relationship between vertices. Besides, the learnt context-aware embeddings can compose high-quality context-free embeddings.

We will explore the following directions in future:

(1) We have investigated the effectiveness of CANE on text-based information networks. In future, we will strive to implement CANE on a wider variety of information networks with multi-modal data, such as labels, images and so on.

(2) CANE encodes latent relations between vertices into their context-aware embeddings. Furthermore, there usually exist explicit relations in social networks (e.g., families, friends and colleagues relations between social network users), which are expected to be critical to NE. Thus, we want to explore how to incorporate and predict these explicit relations between vertices in NE.

Acknowledgements

This work is supported by the 973 Program (No. 2014CB340501), the National Natural Science Foundation of China (NSFC No. 61572273, 61532010, 61661146007), and Tsinghua University Initiative Scientific Research Program (20151080406).

References

Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. 2008. Mixed membership stochastic blockmodels. *JMLR* 9(Sep):1981–2014.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of CIKM*. pages 891–900.

Jifan Chen, Qi Zhang, and Xuanjing Huang. 2016. Incorporate group information to enhance network embedding. In *Proceedings of CIKM*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *JMLR* 9:1871–1874.

Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of KDD*.

James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology* 143(1):29–36.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of NIPS*. pages 2042–2050.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058* .

Yoon Kim. 2014. Convolutional neural networks for sentence classification.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of NIPS*. pages 3294–3302.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of KDD*. pages 177–187.

Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval Journal* 3:127–163.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*. pages 3111–3119.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of KDD*. pages 701–710.

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

Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. 2016. A general framework for content-enhanced network representation learning. *arXiv preprint arXiv:1610.02906* .

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of WWW*. pages 1067–1077.

Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *Proceedings of SIGKDD*. pages 817–826.

Cunchao Tu, Hao Wang, Xiangkai Zeng, Zhiyuan Liu, and Maosong Sun. 2016a. Community-enhanced network representation learning for network analysis. *arXiv preprint arXiv:1611.06645* .

Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. 2016b. Max-margin deepwalk: Discriminative learning of network representation. In *Proceedings of IJCAI*.

Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of KDD*.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *Proceedings of IJCAI*.

Universal Dependencies Parsing for Colloquial Singaporean English

Hongmin Wang[†], **Yue Zhang**[†],
GuangYong Leonard Chan[‡], **Jie Yang**[†], **Hai Leong Chieu**[‡]
† Singapore University of Technology and Design
{hongmin_wang, yue_zhang}@sutd.edu.sg
jie_yang@mymail.sutd.edu.sg
‡ DSO National Laboratories, Singapore
{cguangyo, chaileon}@dso.org.sg

Abstract

Singlish can be interesting to the ACL community both linguistically as a major creole based on English, and computationally for information extraction and sentiment analysis of regional social media. We investigate dependency parsing of Singlish by constructing a dependency treebank under the Universal Dependencies scheme, and then training a neural network model by integrating English syntactic knowledge into a state-of-the-art parser trained on the Singlish treebank. Results show that English knowledge can lead to 25% relative error reduction, resulting in a parser of 84.47% accuracies. To the best of our knowledge, we are the first to use neural stacking to improve cross-lingual dependency parsing on low-resource languages. We make both our annotation and parser available for further research.

1 Introduction

Languages evolve temporally and geographically, both in vocabulary as well as in syntactic structures. When major languages such as English or French are adopted in another culture as the primary language, they often mix with existing languages or dialects in that culture and evolve into a stable language called a creole. Examples of creoles include the French-based Haitian Creole, and Colloquial Singaporean English (Singlish) (Mian-Lian and Platt, 1993), an English-based creole. While the majority of the natural language processing (NLP) research attention has been focused on the major languages, little work has been done on adapting the components to creoles. One notable body of work originated from the featured translation task of the EMNLP 2011 Workshop on Statistical Machine Translation (WMT11) to translate Haitian Creole SMS messages sent during the 2010 Haitian earthquake. This work highlights the importance of NLP tools on creoles in crisis situations for emergency relief (Hu et al., 2011; Hewavitharana et al., 2011).

Singlish is one of the major languages in Singapore, with borrowed vocabulary and grammars[1] from a number of languages including Malay, Tamil, and Chinese dialects such as Hokkien, Cantonese and Teochew (Leimgruber, 2009, 2011), and it has been increasingly used in written forms on web media. Fluent English speakers unfamiliar with Singlish would find the creole hard to comprehend (Harada, 2009). Correspondingly, fundamental English NLP components such as POS taggers and dependency parsers perform poorly on such Singlish texts as shown in Table 2 and 4. For example, Seah et al. (2015) adapted the Socher et al. (2013) sentiment analysis engine to the Singlish vocabulary, but failed to adapt the parser. Since dependency parsers are important for tasks such as information extraction (Miwa and Bansal, 2016) and discourse parsing (Li et al., 2015), this hinders the development of such downstream applications for Singlish in written forms and thus makes it crucial to build a dependency parser that can perform well natively on Singlish.

To address this issue, we start with investigating the linguistic characteristics of Singlish and specifically the causes of difficulties for understanding Singlish with English syntax. We found that, despite the obvious attribute of inheriting a large portion of basic vocabularies and grammars from English, Singlish not only imports terms from regional languages and dialects, its lexical

[1] We follow Leimgruber (2011) in using "grammar" to describe "syntactic constructions" and we do not differentiate the two expressions in this paper.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1732–1744
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1159

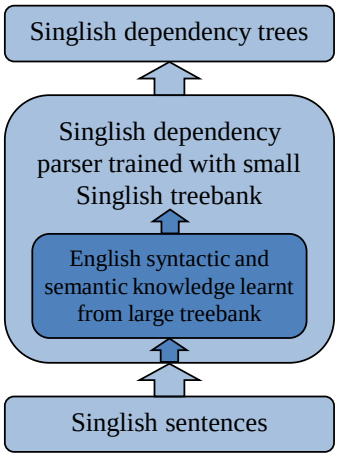


Figure 1: Overall model diagram

semantics and syntax also deviate significantly from English (Leimgruber, 2009, 2011). We categorize the challenges and formalize their interpretation using Universal Dependencies (Nivre et al., 2016), which extends to the creation of a Singlish dependency treebank with 1,200 sentences.

Based on the intricate relationship between Singlish and English, we build a Singlish parser by leveraging knowledge of English syntax as a basis. This overall approach is illustrated in Figure 1. In particular, we train a basic Singlish parser with the best off-the-shelf neural dependency parsing model using biaffine attention (Dozat and Manning, 2017), and improve it with knowledge transfer by adopting neural stacking (Chen et al., 2016; Zhang and Weiss, 2016) to integrate the English syntax. Since POS tags are important features for dependency parsing (Chen and Manning, 2014; Dyer et al., 2015), we train a POS tagger for Singlish following the same idea by integrating English POS knowledge using neural stacking.

Results show that English syntax knowledge brings 51.50% and 25.01% relative error reduction on POS tagging and dependency parsing respectively, resulting in a Singlish dependency parser with 84.47% unlabeled attachment score (UAS) and 77.76% labeled attachment score (LAS).

We make our Singlish dependency treebank, the source code for training a dependency parser and the trained model for the parser with the best performance freely available online[2].

[2] https://github.com/wanghm92/Sing_Par

2 Related Work

Neural networks have led to significant advance in the performance for dependency parsing, including transition-based parsing (Chen and Manning, 2014; Zhou et al., 2015; Weiss et al., 2015; Dyer et al., 2015; Ballesteros et al., 2015; Andor et al., 2016), and graph-based parsing (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017). In particular, the biaffine attention method of Dozat and Manning (2017) uses deep bi-directional long short-term memory (bi-LSTM) networks for high-order non-linear feature extraction, producing the highest-performing graph-based English dependency parser. We adopt this model as the basis for our Singlish parser.

Our work belongs to a line of work on transfer learning for parsing, which leverages English resources in Universal Dependencies to improve the parsing accuracies of low-resource languages (Hwa et al., 2005; Cohen and Smith, 2009; Ganchev et al., 2009). Seminal work employed statistical models. McDonald et al. (2011) investigated delexicalized transfer, where word-based features are removed from a statistical model for English, so that POS and dependency label knowledge can be utilized for training a model for low-resource language. Subsequent work considered syntactic similarities between languages for better feature transfer (Täckström et al., 2012; Naseem et al., 2012; Zhang and Barzilay, 2015).

Recently, a line of work leverages neural network models for multi-lingual parsing (Guo et al., 2015; Duong et al., 2015; Ammar et al., 2016). The basic idea is to map the word embedding spaces between different languages into the same vector space, by using sentence-aligned bilingual data. This gives consistency in tokens, POS and dependency labels thanks to the availability of Universal Dependencies (Nivre et al., 2016). Our work is similar to these methods in using a neural network model for knowledge sharing between different languages. However, ours is different in the use of a neural stacking model, which respects the distributional differences between Singlish and English words. This empirically gives higher accuracies for Singlish.

Neural stacking was previously used for cross-annotation (Chen et al., 2016) and cross-task (Zhang and Weiss, 2016) joint-modelling on monolingual treebanks. To the best of our knowledge, we are the first to employ it on cross-lingual

feature transfer from resource-rich languages to improve dependency parsing for low-resource languages. Besides these three dimensions in dealing with heterogeneous text data, another popular area of research is on the topic of domain adaption, which is commonly associated with cross-lingual problems (Nivre et al., 2007). While this large strand of work is remotely related to ours, we do not describe them in details.

Unsupervised rule-based approaches also offer an competitive alternative for cross-lingual dependency parsing (Naseem et al., 2010; Gillenwater et al., 2010; Gelling et al., 2012; Søgaard, 2012a,b; Martínez Alonso et al., 2017), and recently been benchmarked for the Universal Dependencies formalism by exploiting the linguistic constraints in the Universal Dependencies to improve the robustness against error propagation and domain adaption (Martínez Alonso et al., 2017). However, we choose a data-driven supervised approach given the relatively higher parsing accuracy owing to the availability of resourceful treebanks from the Universal Dependencies project.

3 Singlish Dependency Treebank

3.1 Universal Dependencies for Singlish

Since English is the major genesis of Singlish, we choose English as the source of lexical feature transfer to assist Singlish dependency parsing. Universal Dependencies provides a set of multilingual treebanks with cross-lingually consistent dependency-based lexicalist annotations, designed to aid development and evaluation for cross-lingual systems, such as multilingual parsers (Nivre et al., 2016). The current version of Universal Dependencies comprises not only major treebanks for 47 languages but also their siblings for domain-specific corpora and dialects. With the aligned initiatives for creating transfer-learning-friendly treebanks, we adopt the Universal Dependencies protocol for constructing the Singlish dependency treebank, both as a new resource for the low-resource languages and to facilitate knowledge transfer from English.

On top of the general Universal Dependencies guidelines, English-specific dependency relation definitions including additional subtypes are employed as the default standards for annotating the Singlish dependency treebank, unless augmented or redefined when necessary. The latest English corpus in Universal Dependencies v1.4[3] collection is constructed from the English Web Treebank (Bies et al., 2012), comprising of web media texts, which potentially smooths the knowledge transfer to our target Singlish texts in similar domains. The statistics of this dataset, from which we obtain English syntactic knowledge, is shown in Table 1 and we refer to this corpus as UD-Eng. This corpus uses 47 dependency relations and we show below how to conform to the same standard while adapting to unique Singlish grammars.

	UD English		Singlish	
	Sentences	Words	Sentences	Words
Train	12,543	204,586	900	8,221
Dev	2,002	25,148	150	1,384
Test	2,077	25,096	150	1,381

Table 1: Division of training, development, and test sets for Singlish Treebank

3.2 Challenges and Solutions for Annotating Singlish

The deviations of Singlish from English come from both the lexical and the grammatical levels (Leimgruber, 2009, 2011), which bring challenges for analysis on Singlish using English NLP tools. The former involves imported vocabularies from the first languages of the local people and the latter can be represented by a set of relatively localized features which collectively form 5 unique grammars of Singlish according to Leimgruber (2011). We find empirically that all these deviations can be accommodated by applying the existing English dependency relation definitions while ensuring consistency with the annotations in other non-English UD treebanks, which are explained with examples as follows.

Imported vocabulary: Singlish borrows a number of words and expressions from its non-English origins (Leimgruber, 2009, 2011), such as "Kiasu", which originates from Hokkien meaning "very anxious not to miss an opportunity".[4] These imported terms often constitute out-of-vocabulary (OOV) words with respect to a standard English treebank and result in difficulties for using English-trained tools on Singlish. All borrowed words are annotated based on their usages in Singlish, which mainly inherit the POS from their genesis languages. Table A4 in Appendix A

[3]Only guidelines for Universal Dependencies v2 but not the English corpus is available when this work is completed.

[4]Definition by the Oxford living Dictionaries for English.

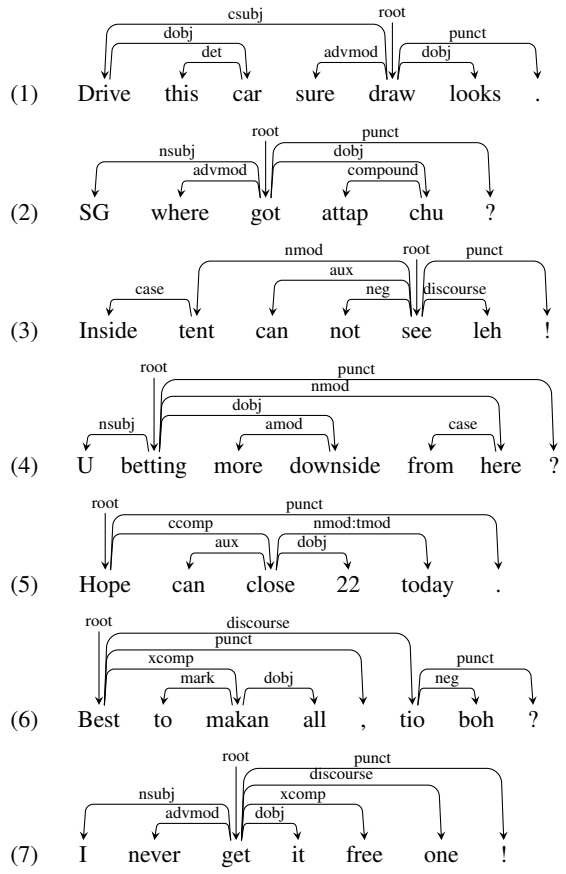


Figure 2: Unique Singlish grammars. (Arcs represent dependencies, pointing from the head to the dependent, with the dependency relation label right on top of the arc)

summarizes all borrowed terms in our treebank.

Topic-prominence: This type of sentences start with establishing its topic, which often serves as the default one that the rest of the sentence refers to, and they typically employ an object-subject-verb sentence structure (Leimgruber, 2009, 2011). In particular, three subtypes of topic-prominence are observed in the Singlish dependency treebank and their annotations are addressed as follows:

First, topics framed as clausal arguments at the beginning of the sentence are labeled as "csubj" (clausal subject), as shown by "Drive this car" of (1) in Figure 2, which is consistent with the dependency relations in its Chinese translation.

Second, noun phrases used to modify the predicate with the absence of a preposition is regarded as a "nsubj" (nominal subject). Similarly, this is a common order of words used in Chinese and one example is the "SG" of (2) in Figure 2.

Third, prepositional phrases moved in front are still treated as "nmod" (nominal modifier) of their intended heads, following the exact definition but as a Singlish-specific form of exemplification, as shown by the "Inside tent" of (3) in Figure 2.

Although the "dislocated" (dislocated elements) relation in UD is also used for preposed elements, but it captures the ones "that do not fulfill the usual core grammatical relations of a sentence" and "not for a topic-marked noun that is also the subject of the sentence" (Nivre et al., 2016). In these three scenarios, the topic words or phrases are in relatively closer grammatical relations to the predicate, as subjects or modifiers.

Copula deletion: Imported from the corresponding Chinese sentence structure, this copula verb is often optional and even deleted in Singlish, which is one of its diagnostic characteristics (Leimgruber, 2009, 2011). In UD-Eng standards, predicative "be" is the only verb used as a copula and it often depends on its complement to avoid copular head. This is explicitly designed in UD to promote parallelism for zero-copula phenomenon in languages such as Russian, Japanese, and Arabic. The deleted copula and its "cop" (copula) arcs are simply ignored, as shown by (4) in Figure 2.

NP deletion: Noun-phrase (NP) deletion often results in null subjects or objects. It may be regarded as a branch of "Topic-prominence" but is a distinctive feature of Singlish with relatively high frequency of usage (Leimgruber, 2011). NP deletion is also common in pronoun-dropping languages such as Spanish and Italian, where the anaphora can be morphologically inferred. In one example, "Vorrei ora entrare brevemente nel merito."[5], from the Italian treebank in UD, "Vorrei" means "I would like to" and depends on the sentence root, "entrare", with the "aux"(auxiliary) relation, where the subject "I" is absent but implicitly understood. Similarly, we do not recover such relations since the deleted NP imposes negligible alteration to the dependency tree, as exemplified by (5) in Figure 2.

Inversion: Inversion in Singlish involves either keeping the subject and verb in interrogative sentences in the same order as in statements, or tag questions in polar interrogatives (Leimgruber, 2011). The former also exists in non-English languages, such as Spanish and Italian, where the subject can prepose the verb in questions (La-

[5]In English: (I) would now like to enter briefly on the merit (of the discussion).

housse and Lamiroy, 2012). This simply involves a change of word orders and thus requires no special treatments. On the other hand, tag questions should be carefully analyzed in two scenarios. One type is in the form of "isn't it?" or "haven't you?", which are dependents of the sentence root with the "parataxis" relation.[6] The other type is exemplified as "right?", and its Singlish equivalent "tio boh?" (a transliteration from Hokkien) are labeled with the "discourse" (discourse element) relation with respect to the sentence root. See example (6) in Figure 2.

Discourse particles: Usage of clausal-final discourse particles, which originates from Hokkien and Cantonese, is one of the most typical feature of Singlish (Leimgruber, 2009, 2011; Lim, 2007). All discourse particles that appear in our treebank are summarized in Table A3 in Appendix A with the imported vocabulary:. These words express the tone of the sentence and thus have the "INTJ" (interjection) POS tag and depend on the root of the sentence or clause labeled with "discourse", as is shown by the "leh" of (3) in Figure 2. The word "one" is a special instance of this type with the sole purpose being a tone marker in Singlish but not English, as shown by (7) in Figure 2.

3.3 Data Selection and Annotation

Data Source: Singlish is used in written form mainly in social media and local Internet forums. After comparison, we chose the *SG Talk Forum*[7] as our data source due to its relative abundance in Singlish contents. We crawled 84,459 posts using the Scrapy framework[8] from pages dated up to 25th December 2016, retaining sentences of length between 5 and 50, which total 58,310. Sentences are reversely sorted according to the log likelihood of the sentence given by an English language model trained using the KenLM toolkit (Heafield et al., 2013)[9] normalized by the sentence length, so that those most different from standard English can be chosen. Among the top 10,000 sentences, 1,977 sentences contain unique Singlish vocabularies defined by *The Coxford Singlish Dictionary*[10], *A Dictionary of Singlish and Singapore English*[11], and the *Singlish Vocabulary* Wikipedia page[12]. The average normalized log likelihood of these 10,000 sentences is -5.81, and the same measure for all sentences in UD-Eng is -4.81. This means these sentences with Singlish contents are 10 times less probable expressed as standard English than the UD-Eng contents in the web domain. This contrast indicates the degree of lexical deviation of Singlish from English. We chose 1,200 sentences from the first 10,000. More than 70% of the selected sentences are observed to consist of the Singlish grammars and imported vocabularies described in section 3.2. Thus the evaluations on this treebank can reflect the performance of various POS taggers and parsers on Singlish in general.

Annotation: The chosen texts are divided by random selection into training, development, and testing sets according to the proportion of sentences in the training, development, and test division for UD-Eng, as summarized in Table 1. The sentences are tokenized using the NLTK Tokenizer,[13] and then annotated using the Dependency Viewer.[14] In total, all 17 UD-Eng POS tags and 41 out of the 47 UD-Eng dependency labels are present in the Singlish dependency treebank. Besides, 100 sentences are randomly selected and double annotated by one of the coauthors, and the inter-annotator agreement has a 97.76% accuracy on POS tagging and a 93.44% UAS and a 89.63% LAS for dependency parsing. A full summary of the numbers of occurrences of each POS tag and dependency label are included in Appendix A.

4 Part-of-Speech Tagging

In order to obtain automatically predicted POS tags as features for a base English dependency parser, we train a POS tagger for UD-Eng using the baseline model of Chen et al. (2016), depicted in Figure 3. The bi-LSTM networks with a CRF layer (bi-LSTM-CRF) have shown state-of-the-art performance by globally optimizing the tag sequence (Huang et al., 2015; Chen et al., 2016).

[6] In UD: Relation between the main verb of a clause and other sentential elements, such as sentential parenthetical clause, or adjacent sentences without any explicit coordination or subordination.

[7] http://sgTalk.com

[8] https://scrapy.org/

[9] Trained using the *afp_eng* and *xin_eng* sources of English Gigaword Fifth Edition (Gigaword).

[10] http://72.5.72.93/html/lexec.php

[11] http://www.singlishdictionary.com

[12] https://en.wikipedia.org/wiki/Singlish_vocabulary

[13] http://www.nltk.org/api/nltk.tokenize.html

[14] http://nlp.nju.edu.cn/tanggc/tools/DependencyViewer.exe

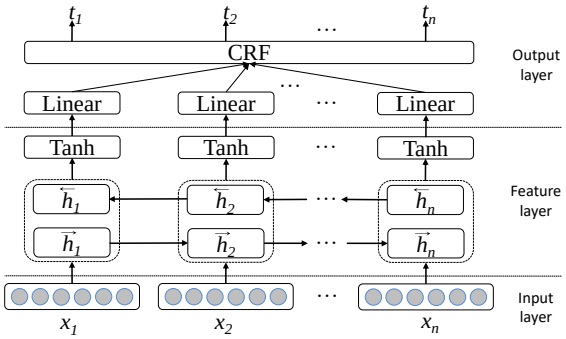


Figure 3: Base POS tagger

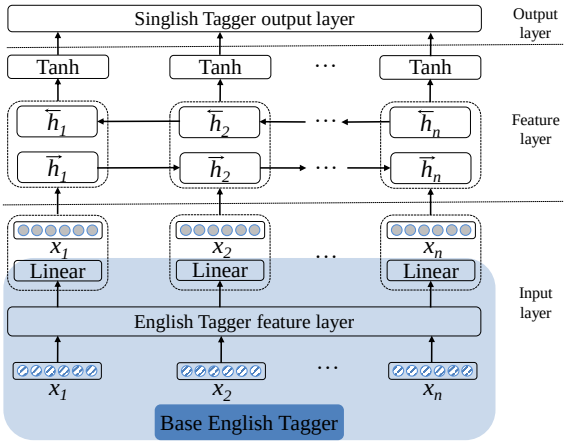


Figure 4: POS tagger with neural stacking

System	Accuracy
ENG-on-SIN	81.39%
Base-ICE-SIN	78.35%
Stack-ICE-SIN	**89.50%**

Table 2: POS tagging accuracies

Based on this English POS tagging model, we train a POS tagger for Singlish using the feature-level neural stacking model of Chen et al. (2016). Both the English and Singlish models consist of an input layer, a feature layer, and an output layer.

4.1 Base Bi-LSTM-CRF POS Tagger

Input Layer: Each token is represented as a vector by concatenating a word embedding from a lookup table with a weighted average of its character embeddings given by the attention model of Bahdanau et al. (2014). Following Chen et al. (2016), the input layer produces a dense representation for the current input token by concatenating its word vector and the ones for its surrounding context tokens in a window of finite size.

Feature Layer: This layer employs a bi-LSTM network to encode the input into a sequence of hidden vectors that embody global contextual information. Following Chen et al. (2016), we adopt bi-LSTM with peephole connections (Graves and Schmidhuber, 2005).

Output layer: This is a CRF layer to predict the POS tags for the input words by maximizing the conditional probability of the sequence of tags given input sentence.

4.2 POS Tagger with Neural Stacking

We adopt the deep integration neural stacking structure presented in Chen et al. (2016). As shown in Figure 4, the distributed vector representation for the target word at the input layer of the Singlish Tagger is augmented by concatenating the emission vector produced by the English Tagger with the original word and character-based embeddings, before applying the concatenation within a context window in section 4.1. During training, loss is back-propagated to all trainable parameters in both the Singlish Tagger and the pre-trained feature layer of the base English Tagger. At test time, the input sentence is fed to the integrated tagger model as a whole for inference.

4.3 Results

We use the publicly available source code[15] by Chen et al. (2016) to train a 1-layer bi-LSTM-CRF based POS tagger on UD-Eng, using 50-dimension pre-trained SENNA word embeddings (Collobert et al., 2011). We set the hidden layer size to 300, the initial learning rate for Adagrad (Duchi et al., 2011) to 0.01, the regularization parameter λ to 10^{-6}, and the dropout rate to 15%. The tagger gives 94.84% accuracy on the UD-Eng test set after 24 epochs, chosen according to development tests, which is comparable to the state-of-the-art accuracy of 95.17% reported by Plank et al. (2016). We use these settings to perform 10-fold jackknifing of POS tagging on the UD-Eng training set, with an average accuracy of 95.60%.

Similarly, we trained a POS tagger using the Singlish dependency treebank alone with pre-trained word embeddings on The Singapore Component of the International Corpus of English (ICE-SIN) (Nihilani, 1992; Ooi, 1997), which consists of both spoken and written texts. However, due to limited amount of training data, the

[15] https://github.com/chenhongshen/NNHetSeq

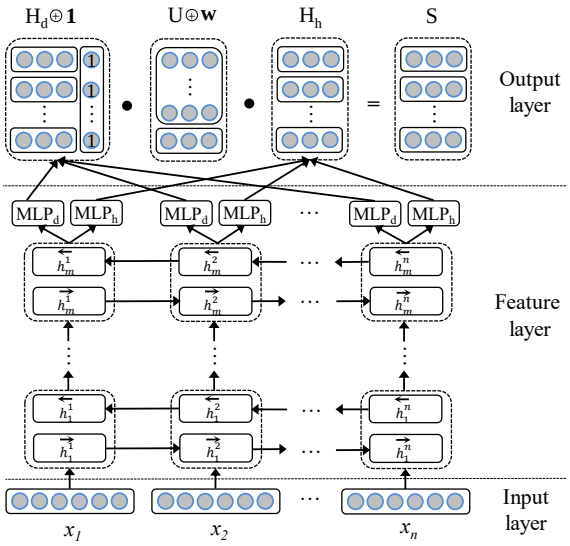


Figure 5: Base parser

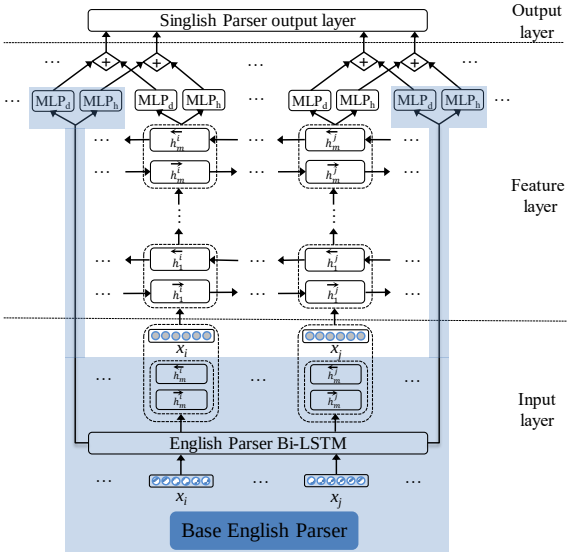


Figure 6: Parser with neural stacking

tagging accuracy is not satisfactory even with a larger dropout rate to avoid over-fitting. In contrast, the neural stacking structure on top of the English base model trained on UD-Eng achieves a POS tagging accuracy of 89.50%[16], which corresponds to a 51.50% relative error reduction over the baseline Singlish model, as shown in Table 2. We use this for 10-fold jackknifing on Singlish parsing training data, and tagging the Singlish development and test data.

5 Dependency Parsing

We adopt the Dozat and Manning (2017) parser[17] as our base model, as displayed in Figure 5, and apply neural stacking to achieve improvements over the baseline parser. Both the base and neural stacking models consist of an input layer, a feature layer, and an output layer.

5.1 Base Parser with Bi-affine Attentions

Input Layer: This layer encodes the current input word by concatenating a pre-trained word embedding with a trainable word embedding and POS tag embedding from the respective lookup tables.

Feature Layer: The two recurrent vectors produced by the multi-layer bi-LSTM network from each input vector are concatenated and mapped to multiple feature vectors in lower-dimension space by a set of parallel multilayer perceptron (MLP) layers. Following Dozat and Manning (2017), we adopt Cif-LSTM cells (Greff et al., 2016).

Output Layer: This layer applies biaffine transformation on the feature vectors to calculate the score of the directed arcs between every pair of words. The inferred trees for input sentence are formed by choosing the head with the highest score for each word and a cross-entropy loss is calculated to update the model parameters.

5.2 Parser with Neural Stacking

Inspired by the idea of feature-level neural stacking (Chen et al., 2016; Zhang and Weiss, 2016), we concatenate the pre-trained word embedding, trainable word and tag embeddings, with the two recurrent state vectors at the last bi-LSTM layer of the English Tagger as the input vector for each target word. In order to further preserve syntactic knowledge retained by the English Tagger, the feature vectors from its MLP layer is added to the ones produced by the Singlish Parser, as illustrated in Figure 6, and the scoring tensor of the Singlish Parser is initialized with the one from the trained English Tagger. Loss is back-propagated by reversely traversing all forward paths to all trainable parameter for training and the whole model is used collectively for inference.

6 Experiments

6.1 Experimental Settings

We train an English parser on UD-Eng with the default model settings in Dozat and Manning (2017).

[16] We empirically find that using ICE-SIN embeddings in neural stacking model performs better than using English SENNA embeddings. Similar findings are found for the parser, of which more details are given in section 6.

[17] https://github.com/tdozat/Parser

	Sentences	Words	Vocabulary
GloVe6B	N.A.	6000m	400,000
Giga100M	57,000	1.26m	54,554
ICE-SIN	87,084	1.26m	40,532

Table 3: Comparison of the scale of sources for training word embeddings

Trained on	System	UAS	LAS
English	ENG-on-SIN	75.89	65.62
Singlish	Baseline	75.98	66.55
	Base-Giga100M	77.67	67.23
	Base-GloVe6B	78.18	68.51
	Base-ICE-SIN	79.29	69.27
Both	ENG-plus-SIN	82.43	75.64
	Stack-ICE-SIN	**84.47**	**77.76**

Table 4: Dependency parser performances

It achieves an UAS of 88.83% and a LAS of 85.20%, which are close to the state-of-the-art 85.90% LAS on UD-Eng reported by Ammar et al. (2016), and the main difference is caused by us not using fine-grained POS tags. We apply the same settings for a baseline Singlish parser. We attempt to choose a better configuration of the number of bi-LSTM layers and the hidden dimension based on the development set performance, but the default settings turn out to perform the best. Thus we stick to all default hyper-parameters in Dozat and Manning (2017) for training the Singlish parsers.

We experimented with different word embeddings, as with the raw text sources summarized in Table 3 and further described in section 6.2. When using the neural stacking model, we fix the model configuration for the base English parser model and choose the size of the hidden vector and the number of bi-LSTM layers stacked on top based on the performance on the development set. It turns out that a 1-layer bi-LSTM with 900 hidden dimension performs the best, where the bigger hidden layer accommodates the elongated input vector to the stacked bi-LSTM and the fewer number of recurrent layers avoids over-fitting on the small Singlish dependency treebank, given the deep bi-LSTM English parser network at the bottom. The evaluation of the neural stacking model is further described in section 6.3.

System	UAS	LAS
Base-ICE-SIN	77.00	66.69
Stack-ICE-SIN	**82.43**	**73.96**

Table 5: Dependency parser performances by the 5-cross-fold validation

6.2 Investigating Distributed Lexical Characteristics

In order to learn characteristics of distributed lexical semantics for Singlish, we compare performances of the Singlish dependency parser using several sets of pre-trained word embeddings: GloVe6B, large-scale English word embeddings[18]; ICE-SIN, Singlish word embeddings trained using GloVe (Pennington et al., 2014) on the ICE-SIN (Nihilani, 1992; Ooi, 1997) corpus; Giga100M, a small-scale English word embeddings trained using GloVe (Pennington et al., 2014) with the same settings on a comparable size of English data randomly selected from the English Gigaword Fifth Edition for a fair comparison with ICE-SIN embeddings.

First, the English Giga100M embeddings marginally improve the Singlish parser from the baseline without pre-trained embeddings and also using the UD-Eng parser directly on Singlish, represented as "ENG-on-SIN" in Table 4. With much more English lexical semantics being fed to the Singlish parser using the English GloVe6B embeddings, further enhancement is achieved. Nevertheless, the Singlish ICE-SIN embeddings lead to even more improvement, with 13.78% relative error reduction, compared with 7.04% using the English Giga100M embeddings and 9.16% using the English GloVe6B embeddings, despite the huge difference in sizes in the latter case. This demonstrates the distributional differences between Singlish and English tokens, even though they share a large vocabulary. More detailed comparison is described in section 6.4.

6.3 Knowledge Transfer Using Neural Stacking

We train a parser with neural stacking and Singlish ICE-SIN embeddings, which achieves the best performance among all the models, with a UAS of 84.47%, represented as "Stack-ICE-SIN" in Table 4, which corresponds to 25.01% relative error reduction compared to the baseline. This demonstrates that knowledge from English can be successfully incorporated to boost the Singlish parser. To further evaluate the effectiveness of the neural stacking model, we also trained a base model with the combination of UD-Eng and the Singlish tree-

[18]Trained with Wikipedia 2014 the Gigaword. Downloadable from http://nlp.stanford.edu/data/glove.6B.zip

	Topic Prominence		Copula Deletion		NP Deletion		Discourse Particles		Others	
Sentences	15		19		21		51		67	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
ENG-on-SIN	78.15	62.96	66.91	56.83	72.57	64.00	70.00	59.00	78.92	68.47
Base-Giga100M	77.78	68.52	71.94	61.15	76.57	69.14	85.25	77.25	73.13	60.63
Base-ICE	81.48	72.22	74.82	63.31	**80.00**	73.71	85.25	77.75	75.56	64.37
Stack-ICE	**87.04**	**76.85**	**77.70**	**71.22**	**80.00**	**75.43**	**88.50**	**83.75**	**84.14**	**76.49**

Table 6: Error analysis with respect to grammar types

bank, represented as "ENG-plus-SIN" in Table 4, which is still outperformed by the neural stacking model. Besides, we performed a 5-cross-fold validation for the base parser with Singlish ICE-SIN embeddings and the parser using neural stacking, where half of the held-out fold is used as the development set. The average UAS and LAS across the 5 folds shown in Table 5 and the relative error reduction on average 23.61% suggest that the overall improvement from knowledge transfer using neural stacking remains consistent. This significant improvement is further explained in section 6.4.

6.4 Improvements over Grammar Types

To analyze the sources of improvements for Singlish parsing using different model configurations, we conduct error analysis over 5 syntactic categories[19], including 4 types of grammars mentioned in section 3.2[20], and 1 for all other cases, including sentences containing imported vocabularies but expressed in basic English syntax. The number of sentences and the results in each group of the test set are shown in Table 6.

The neural stacking model leads to the biggest improvement over all categories except for a tie UAS performance on "NP Deletion" cases, which explains the significant overall improvement.

Comparing the base model with ICE-SIN embeddings with the base parser trained on UD-Eng, which contain syntactic and semantic knowledge in Singlish and English, respectively, the former outperforms the latter on all 4 types of Singlish grammars but not for the remaining samples. This suggests that the base English parser mainly contributes to analyzing basic English syntax, while the base Singlish parser models unique Singlish grammars better.

Similar trends are also observed on the base model using the English Giga100M embeddings, but the overall performances are not as good as using ICE-SIN embeddings, especially over basic English syntax where it undermines the performance to a greater extent. This suggests that only limited English distributed lexical semantic information can be integrated to help modelling Singlish syntactic knowledge due to the differences in distributed lexical semantics.

7 Conclusion

We have investigated dependency parsing for Singlish, an important English-based creole language, through annotations of a Singlish dependency treebank with 10,986 words and building an enhanced parser by leveraging on knowledge transferred from a 20-times-bigger English treebank of Universal Dependencies. We demonstrate the effectiveness of using neural stacking for feature transfer by boosting the Singlish dependency parsing performance to from UAS 79.29% to UAS 84.47%, with a 25.01% relative error reduction over the parser with all available Singlish resources. We release the annotated Singlish dependency treebank, the trained model and the source code for the parser with free public access. Possible future work include expanding the investigation to other regional languages such as Malay and Indonesian.

Acknowledgments

Yue Zhang is the corresponding author. This research is supported by IGDSS1603031 from Temasek Laboratories@SUTD. We appreciate anonymous reviewers for their insightful comments, which helped to improve the paper, and Zhiyang Teng, Jiangming Liu, Yupeng Liu, and Enrico Santus for their constructive discussions.

[19]Multiple labels are allowed for one sentence.

[20]The "Inversion" type of grammar is not analyzed since there is only 1 such sentence in the test set.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association of Computational Linguistics* 4:431–444. http://aclweb.org/anthology/Q16-1031.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the ACL 2016*. Association for Computational Linguistics, pages 2442–2452. https://doi.org/10.18653/v1/P16-1231.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Miguel Ballesteros, Chris Dyer, and A. Noah Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the EMNLP 2015*. Association for Computational Linguistics, pages 349–359. https://doi.org/10.18653/v1/D15-1041.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English web treebank ldc2012t13 .

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the EMNLP 2014*. Association for Computational Linguistics, pages 740–750. https://doi.org/10.3115/v1/D14-1082.

Hongshen Chen, Yue Zhang, and Qun Liu. 2016. Neural network for heterogeneous annotations. In *Proceedings of the EMNLP 2016*. Association for Computational Linguistics, pages 731–741. http://aclweb.org/anthology/D16-1070.

Shay Cohen and A. Noah Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of the NAACL-HLT 2009*. Association for Computational Linguistics, pages 74–82. http://aclweb.org/anthology/N09-1009.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537. http://dl.acm.org/citation.cfm?id=2078186.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations 2017*. volume abs/1611.01734. http://arxiv.org/abs/1611.01734.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159. http://dl.acm.org/citation.cfm?id=2021068.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. A neural network model for low-resource universal dependency parsing. In *Proceedings of the EMNLP 2015*. Association for Computational Linguistics, pages 339–348. https://doi.org/10.18653/v1/D15-1040.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and A. Noah Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the ACL-IJCNLP 2015*. Association for Computational Linguistics, pages 334–343. https://doi.org/10.3115/v1/P15-1033.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the ACL-IJCNLP 2009*. Association for Computational Linguistics, pages 369–377. http://aclweb.org/anthology/P09-1042.

Douwe Gelling, Trevor Cohn, Phil Blunsom, and Joao Graca. 2012. The pascal challenge on grammar induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*. Association for Computational Linguistics, pages 64–80. http://www.aclweb.org/anthology/W12-1909.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. In *Proceedings of the ACL 2010 (Short Papers)*. Association for Computational Linguistics, pages 194–199. http://www.aclweb.org/anthology/P10-2036.

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

K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber. 2016. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems* PP(99):1–11. https://doi.org/10.1109/TNNLS.2016.2582924.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the ACL-IJCNLP 2015*. Association for Computational Linguistics, pages 1234–1244. https://doi.org/10.3115/v1/P15-1119.

Shinichi Harada. 2009. The roles of singapore standard english and singlish. *Information Research* 40:70–82.

Kenneth Heafield, Ivan Pouzyrevsky, H. Jonathan Clark, and Philipp Koehn. 2013. Scalable modified kneser-ney language model estimation. In *Proceedings of the ACL 2013 (Short Papers)*. Association for Computational Linguistics, pages 690–696. http://aclweb.org/anthology/P13-2121.

Sanjika Hewavitharana, Nguyen Bach, Qin Gao, Vamshi Ambati, and Stephan Vogel. 2011. *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Association for Computational Linguistics, chapter CMU Haitian Creole-English Translation System for WMT 2011, pages 386–392. http://aclweb.org/anthology/W11-2146.

Chang Hu, Philip Resnik, Yakov Kronrod, Vladimir Eidelman, Olivia Buzek, and B. Benjamin Bederson. 2011. *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Association for Computational Linguistics, chapter The Value of Monolingual Crowdsourcing in a Real-World Translation Scenario: Simulation using Haitian Creole Emergency SMS Messages, pages 399–404. http://aclweb.org/anthology/W11-2148.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint* abs/1508.01991. http://arxiv.org/abs/1508.01991.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering* 11(3):311–325. https://doi.org/10.1017/S1351324905003840.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association of Computational Linguistics* 4:313–327. http://aclweb.org/anthology/Q16-1023.

Karen Lahousse and Béatrice Lamiroy. 2012. Word order in french, spanish and italian: A grammaticalization account. *Folia Linguistica* 46(2):387–415.

Jakob R. E. Leimgruber. 2009. *Modelling variation in Singapore English*. Ph.D. thesis, Oxford University.

Jakob R. E. Leimgruber. 2011. Singapore english. *Language and Linguistics Compass* 5(1):47–62. https://doi.org/10.1111/j.1749-818X.2010.00262.x.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the EMNLP 2015*. Association for Computational Linguistics, pages 2304–2314. https://doi.org/10.18653/v1/D15-1278.

Lisa Lim. 2007. Mergers and acquisitions: on the ages and origins of singapore english particles. *World Englishes* 26(4):446–473.

Héctor Martínez Alonso, Željko Agić, Barbara Plank, and Anders Søgaard. 2017. Parsing universal dependencies without training. In *Proceedings of the EACL 2017*. Association for Computational Linguistics, pages 230–240. http://www.aclweb.org/anthology/E17-1022.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the EMNLP 2011*. Association for Computational Linguistics, pages 62–72. http://aclweb.org/anthology/D11-1006.

Ho Mian-Lian and John T. Platt. 1993. *Dynamics of a contact continuum: Singaporean English*. Oxford University Press, USA.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the ACL 2016*. Association for Computational Linguistics, pages 1105–1116. https://doi.org/10.18653/v1/P16-1105.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the ACL 2012*. Association for Computational Linguistics, pages 629–637. http://aclweb.org/anthology/P12-1066.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the EMNLP 2010*. Association for Computational Linguistics, Cambridge, MA, pages 1234–1244. http://www.aclweb.org/anthology/D10-1120.

Paroo Nihilani. 1992. The international computerized corpus of english. *Words in a cultural context. Singapore: UniPress* pages 84–88.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the LREC 2016*. European Language Resources Association.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*. Association for Computational Linguistics, pages 915–932. http://www.aclweb.org/anthology/D/D07/D07-1096.

Vincent B Y Ooi. 1997. *Analysing the Singapore ICE corpus for lexicographic evidence*. ENGLISH LANGUAGE & LITERATURE. http://scholarbank.nus.edu.sg/handle/10635/133118.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the EMNLP 2014*. Association for Computational Linguistics, pages 1532–1543. https://doi.org/10.3115/v1/D14-1162.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the ACL 2016 (Short Papers)*. Association for Computational Linguistics, pages 412–418. https://doi.org/10.18653/v1/P16-2067.

Chun-Wei Seah, Hai Leong Chieu, Kian Ming Adam Chai, Loo-Nin Teow, and Lee Wei Yeong. 2015. Troll detection by domain-adapting sentiment analysis. In *18th International Conference on Information Fusion (Fusion) 2015*. IEEE, pages 792–799.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, D. Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the EMNLP 2013*. Association for Computational Linguistics, pages 1631–1642. http://aclweb.org/anthology/D13-1170.

Anders Søgaard. 2012a. Two baselines for unsupervised dependency parsing. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*. Association for Computational Linguistics, pages 81–83. http://www.aclweb.org/anthology/W12-1910.

Anders Søgaard. 2012b. Unsupervised dependency parsing without training. *Natural Language Engineering* 18(2):187203. https://doi.org/10.1017/S1351324912000022.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the NAACL-HLT 2012*. Association for Computational Linguistics, pages 477–487. http://aclweb.org/anthology/N12-1052.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the ACL-IJCNLP 2015*. Association for Computational Linguistics, pages 323–333. https://doi.org/10.3115/v1/P15-1032.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the EMNLP 2015*. Association for Computational Linguistics, pages 1857–1867. https://doi.org/10.18653/v1/D15-1213.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th ACL*. Association for Computational Linguistics, pages 1557–1566. https://doi.org/10.18653/v1/P16-1147.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the ACL-IJCNLP 2015*. Association for Computational Linguistics, pages 1213–1222. https://doi.org/10.3115/v1/P15-1117.

A Statistics of Singlish Dependency Treebank

POS Tags					
ADJ	782	INTJ	556	PUNCT	1604
ADP	490	NOUN	1779	SCONJ	126
ADV	941	NUM	153	SYM	11
AUX	429	PART	355	VERB	1704
CONJ	167	PRON	682	X	10
DET	387	PROPN	810		

Table A1: Statistics of POS tags

Dependency labels			
acl	37	dobj	612
acl:relcl	29	expl	10
advcl	194	iobj	15
advmod	859	list	10
appos	18	mwe	105
amod	423	name	117
aux	377	neg	261
auxpass	47	nmod	398
case	463	nmod:npmod	26
cc	167	nmod:poss	153
ccomp	138	nmod:tmod	81
compound	420	nsubj	1005
compound:prt	30	nsubjpass	34
conj	238	nummod	94
cop	152	mark	275
csubj	30	parataxis	241
det	304	punct	1607
det:predet	7	remnant	17
discourse	552	vocative	41
dislocated	2	xcomp	190

Table A2: Statistics of dependency labels

ah	aiyah	ba
hah / har / huh	hiak hiak hiak	hor
huat	la / lah	lau
leh	loh / lor	ma / mah
wahlow / wah lau	wa / wah	ya ya
walaneh / wah lan eh		

Table A3: List of discourse particles

A-B		
act blur	ah beng	ah ne
angpow	arrowed	ang ku kueh
angmoh/ang moh	ahpek / ah peks	atas
boh/bo	boho jiak	boh pian
buay lin chu	buen kuey	
C		
chai tow kway	chao ah beng	chap chye png
char kway teow	chee cheong fun / che cheong fen	
cheesepie	cheong / chiong	chiam / cham
chiak liao bee / jiao liao bee		chio
ching chong	chio bu / chiobu	chui
chop chop	chow-angmoh	chwee kueh
D-F		
dey	diam diam	die kock standing
die pain pain	dun	eat grass
flip prata	fried beehoon	
G		
gahmen / garment	gam	geylang
gone case	gong kia	goreng pisang
gui		
H-J		
hai si lang	heng	hiong
hoot	Hosay / ho say	how lian
jepun kia / jepun kias		
jialat / jia lak / jia lat		
K		
ka	kaki kong kaki song	
kancheong	kateks	kautim
kay kiang	kayu	kee chia
kee siao	kelong	kena / kana
kiam	kiasu	ki seow
kkj	kong si mi	kopi
kopi lui	kopi-o	kosong
koyok	ku ku bird	
L		
lagi	lai liao	laksa
lao jio kong	lao sai	lau chwee nua
liao / ler	like dat / like that	
lim peh	lobang	
M		
mahjong kaki	makan	masak masak
mati	mee	mee pok
mee rebus	mee siam	mee sua
mei mei		
N-S		
nasi lemak	pang sai	piak
sabo	sai	same same
sia	sianz / sian	sia suay
sibeh	siew dai	siew siew dai
simi taisee	soon kuey	sotong
suay / suey	swee	
T		
tahan	tak pakai	te te kee
tong	tua	tikopeh
tio	tio pian/dio pian	
talk cock / talk cock sing song		
U-Z		
umm zai	up lorry / up one's lorry	
xiao	zhun / buay zhun	

Table A4: List of imported vocabularies

Generic Axiomatization of Families of Noncrossing Graphs in Dependency Parsing

Anssi Yli-Jyrä
University of Helsinki, Finland
anssi.yli-jyra@helsinki.fi

Carlos Gómez-Rodríguez
Universidade da Coruña, Spain
carlos.gomez@udc.es

Abstract

We present a simple encoding for unlabeled noncrossing graphs and show how its latent counterpart helps us to represent several families of directed and undirected graphs used in syntactic and semantic parsing of natural language as context-free languages. The families are separated purely on the basis of forbidden patterns in latent encoding, eliminating the need to differentiate the families of non-crossing graphs in inference algorithms: one algorithm works for all when the search space can be controlled in parser input.

1 Introduction

Dependency parsing has received wide attention in recent years, as accurate and efficient dependency parsers have appeared that are applicable to many languages. Traditionally, dependency parsers have produced syntactic analyses in tree form, including exact inference algorithms that search for maximum projective trees (Eisner and Satta, 1999) and maximum spanning trees (McDonald et al., 2005) in weighted digraphs, as well as greedy and beam-search approaches that forgo exact search for extra efficiency (Zhang and Nivre, 2011).

Recently, there has been growing interest in providing a richer analysis of natural language by going beyond trees. In semantic dependency parsing (Oepen et al., 2015; Kuhlmann and Oepen, 2016), the desired syntactic representations can have in-degree greater than 1 (re-entrancy), suggesting the search for maximum acyclic subgraphs (Schluter, 2014, 2015). As this inference task is intractable (Guruswami et al., 2011), noncrossing digraphs have been studied instead, e.g. by Kuhlmann and Johnsson (2015) who provide a $O(n^3)$ parser for maximum noncrossing acyclic subgraphs.

Yli-Jyrä (2005) studied how to axiomatize dependency trees as a special case of noncrossing digraphs. This gave rise to a new homomorphic representation of context-free languages that proves the classical Chomsky and Schützenberger theorem using a quite different internal language. In this language, the brackets indicate arcs in a dependency tree in a way that is reminiscent to encoding schemes used earlier by Greibach (1973) and Oflazer (2003). Cubic-time parsing algorithms that are incidentally or intentionally applicable to this kind of homomorphic representations have been considered, e.g., by Nederhof and Satta (2003), Hulden (2011), and Yli-Jyrä (2012).

Extending these insights to arbitrary noncrossing digraphs, or to relevant families of them, is far from obvious. In this paper, we develop (1) a *linear encoding* supporting general noncrossing digraphs, and (2) show that the encoded noncrossing digraphs form a *context-free language*. We then give it (3) two *homomorphic, nonderivative representations* and use the latent local features of the latter to characterize various families of digraphs.

Apart from the obvious relevance to the theory of context-free languages, this contribution has the practical potential to enable (4) *generic context-free parsers* that produce different families of noncrossing graphs with the same set of inference rules while the search space in each case is restricted with lexical features and the grammar.

Outline After some background on graphs and parsing as inference (Section 2), we use an **ontology of digraphs** to illustrate natural families of noncrossing digraphs in Section 3. We then develop, in Section 4, the first **latent context-free representation** for the set of noncrossing digraphs, then extended in Section 5 with additional latent states supporting our **finite-state axiomatization** of digraph properties, and allowing us to

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1745–1755
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1160

control the search space using the lexicon. The **experiments** in Section 6 cross-validate our axioms and sample the growth of the constrained search spaces. Section 7 outlines the applications for practical parsing, and Section 8 concludes.

2 Background

Graphs and Digraphs A *graph* is a pair (V,E) where V is a finite set of vertices and $E \subseteq \{\{u,v\} \subseteq V\}$ is a set of edges. A sequence of edges of the form $\{v_0,v_1\}$, $\{v_1,v_2\}$, ..., $\{v_{m-1},v_m\}$, with no repetitions in $v_1,...,v_m$, is a *path* between vertices v_0 and v_m and *empty* if $m = 0$. A graph is a *forest* if no vertex has a non-empty path to itself and *connected* if all pairs of vertices have a path. A *tree* is a connected forest.

A *digraph* is a pair (V,A) where $A \subseteq V \times V$ is a set of arcs $u \to v$, thus a *directed* graph. Its *underlying graph*, (V,E_A), has edges $E_A = \{\{u,v\} \mid (u,v) \in A\}$. A sequence of arcs $v_0 \to v_1, v_1 \to v_2, ..., v_{m-1} \to v_m$, with no repetitions in $v_1,\ldots,v_m$, is a *directed path*, and *empty* if $m = 0$.

A digraph without self-loops $v \to v$ is *loop-free* (property $\mathbf{DIGRAPH_{LF}}$). We will focus on loop-free digraphs unless otherwise specified, and denote them just by **DIGRAPH** for brevity. A digraph is d-acyclic ($\mathbf{ACYC_D}$), aka a *dag* if no vertex has a non-empty directed path to itself, *u-acyclic* ($\mathbf{ACYC_U}$) aka a *m(ixed)-forest* if its underlying graph is a forest, and *weakly connected* (w.c., $\mathbf{CONN_W}$) if its underlying graph is connected.

Dependency Parsing The *complete digraph* $G_S(V,A)$ *of a sentence* $S = x_1...x_n$ consists of vertices $V = \{1,...,n\}$ and all possible arcs $A = V \times V - \{(i,i)\}$. The vertex $i \in V$ corresponds to the word x_i and the arc $i \to j \in A$ corresponds to a possible dependency between the words x_i and x_j.

The task of *dependency parsing* is to find a constrained subgraph $G'_S(V,A')$ of the complete digraph G_S of the sentence. The standard solution is a rooted directed tree called a *dependency tree* or a dag called a *dependency graph*.

Constrained Inference In *arc-factored parsing* (McDonald et al., 2005), each possible arc $i \to j$ is equipped with a positive weight w_{ij}, usually computed as a weighted sum $w_{ij} = \mathbf{w} \cdot \Phi(S, i \to j)$ where $\mathbf{w}$ is a weight vector and $\Phi(\mathbf{x}, i \to j)$ a feature vector extracted from the sentence $\mathbf{x}$, considering the dependency relation from word x_i to word x_j. Parsing then consists in finding an arc subset $A' \subseteq A$ that gives us a constrained subgraph $(V,A') \in \text{Constrained}(V,A)$ of the complete digraph (V,A) with maximum sum of arc weights:

$$(V,A') = \underset{(V,A') \in \text{Constrained}(V,A)}{\arg\max} \sum_{i \to j \in A'} w_{i,j}.$$

The complexity of this inference task depends on the constraints imposed on the subgraph. Under no constraints, we simply set $A' = A$. Inference over dags is intractable (Guruswami et al., 2011). Efficient solutions are known for projective trees (Eisner, 1996), various classes of mildly non-projective trees (Gómez-Rodríguez, 2016), unrestricted spanning trees (McDonald et al., 2005), and both unrestricted and weakly connected non-crossing dags (Kuhlmann and Johnsson, 2015).

Parsimony Semantic parsers must be able to produce more than projective trees because the share of projective trees is pretty low (under 3%) in semantic graph banks (Kuhlmann and Johnsson, 2015). However, if we know that the parses have some restrictions, it is better to use them to restrict the search space as much as possible.

There are two strategies for reducing the search space. One is to develop a specialized inference algorithm for a particular natural language or family of dags, such as weakly connected graphs (Kuhlmann and Johnsson, 2015). The other strategy is to control the local complexity of digraphs through lexical categories (Baldridge and Kruijff, 2003) or equivalent mechanisms. This strategy produces a more sensitive model of the language, but requires a principled insight on how the complexity of digraphs can be characterized.

3 Constraints on the Search Space

We will now present a classification of digraphs on the basis of their formal properties.

The Noncrossing Property For convenience, graphs and digraphs may be ordered like in a complete digraph of a sentence. Two edges $\{i,j\}$, $\{k,l\}$ in an ordered graph or arcs $i \to j, k \to l$ in an ordered digraph are said to be *crossing* if $\min\{i,j\} < \min\{k,l\} < \max\{i,j\} < \max\{k,l\}$. A graph or digraph is *noncrossing* if it has no crossing edges or arcs. Noncrossing (di)graphs (**NC-(DI)GRAPH**) are the largest possible (di)graphs that can be drawn on a circle without crossing arcs. In the following, we assume that all digraphs and graphs are noncrossing.

An arc $x \to y$ is *(properly) covered by* an arc $z \to t$ if $(\{x,y\} \neq \{z,t\})$ and $\min\{z,t\} \leq \min\{x,y\} \leq \max\{x,y\} \leq \max\{z,t\}$.

Ontology Fig. 1 presents an ontology of such families of loop-free noncrossing digraphs that can be distinguished by digraphs with 5 vertices.

In the digraph ontology, a *multitree* aka mangrove is a dag with the property of being *strongly unambiguous* (**UNAMB**$_{\mathrm{S}}$), which asserts that, given two distinct vertices, there is at most one repeat-free path between them (Lange, 1997).[1] A *polytree* (Rebane and Pearl, 1987) is a multitree whose underlying graph is a tree. The *out* property (**OUT**) of a digraph (V,E) means that no vertex $i \in V$ has two incoming arcs $\{j,k\} \to i$ s.t. $j \neq k$.

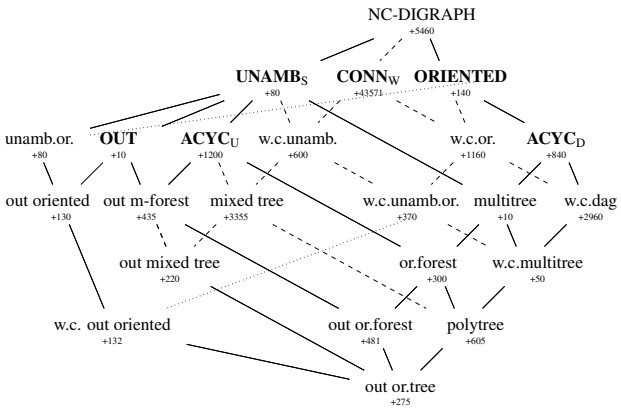


Figure 1: Basic properties split the set of 62464 noncrossing digraphs for 5 vertices into 23 classes

An ordered digraph is *weakly projective* (**PROJ**$_{\mathrm{W}}$) if for all vertices i, j and k, if $k \to j \to i$, then either $\{i,j\} < k$ or $\{i,j\} > k$. In other words, the constraint, aka the *outside-to-inside constraint* (Yli-Jyrä, 2005), states that no outgoing arc of a vertex properly covers an incoming arc. This is implied by a stronger constraint known as *Harper, Hays, Lecerf and Ihm projectivity* (Marcus, 1967).

We can embed the ontology of graphs (unrestricted, connected, forests and trees) into the ontology of digraphs by viewing an undirected graph (V,E) as an inverse digraph $(V, \{(i,j),(j,i) \mid \{i,j\} \in E\})$. This kind of digraph has an *inverse property* (**INV**). Its opposite is an *oriented (or.)* digraph (V,A) where $i \to j \in A$ implies $j \to i \notin A$ (defines the property **ORIENTED**). Out forests and trees are, by convention, oriented digraphs with an underlying forest or tree, respectively.

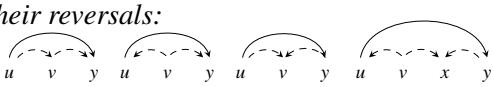
[1] A different definition forbids diamonds as minors.

Distinctive Properties A few important properties of digraphs are local and can be verified by inspecting each vertex separately with its incident arcs. These include (i) the out property (**OUT**), (ii) the nonstandard projectivity property (**PROJ**$_{\mathrm{W}}$), (iii) the inverse property (**INV**) and (iv) the orientedness (or.) property.

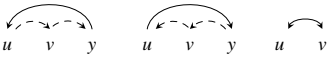
Properties **UNAMB**$_{\mathrm{S}}$, **ACYC**$_{\mathrm{D}}$, **CONN**$_{\mathrm{W}}$, and **ACYC**$_{\mathrm{U}}$ are nonlocal properties of digraphs and cannot be generally verified locally, through finite spheres of vertices (Grädel et al., 2005). The following proposition covers the configurations that we have to detect in order to decide the nonlocal properties of noncrossing digraphs.

Proposition 1. *Let $G = (V,E)$ be a noncrossing digraph.*

- *If $G \notin$ **UNAMB**$_S$, then the digraph contains one of the following four configurations or their reversals:*

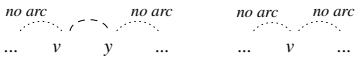


- *If $G \notin$ **ACYC**$_D$, then the graph contains one of the configurations*



- *If $G \notin$ **ACYC**$_U$, then the underlying graph contains the following configuration:*

- *If $G \notin$ **CONN**$_W$, then the underlying graph contains one of the following configurations:*



Proposition 1 gives us a means to implement the property tests in practice. It tells us intuitively that although the paths can be arbitrarily long, any underlying cycle containing more than 2 arcs consists of one covering arc and a linear chain of edges between its end points.

4 The Set of Digraphs as a Language

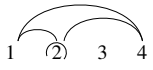
In this section, we show that the set of noncrossing digraphs is isomorphic to an unambiguous context-free language over a bracket alphabet.

4.1 Basic Encoding

Any noncrossing ordered graph $([1,\ldots,n],E)$, even with self-loops, can be encoded as a string of brackets using the algorithm `enc` in Fig. 2. For example, the output for the ordered graph

```
func enc(n,E):                    func dec(stdin):
  for i in [1,...,n]:               n = 1; E = {}; s = []
    for j in [i-1,...,2,1]:         while c in stdin:
      if {j,i} in E:                   if c == "[":
        print "]"                         s.push(n)
    for j in [n,n-1,...,i+1]:           if c == "]":
      if {i,j} in E:                       i = s.pop()
        print "["                          E.insert((i,n))
    if {i,i} in E:                      if c == "{":
      print "[]"                           n = n + 1
    if i<n:
      print "{}"                        return (n,E)
```

Figure 2: The encoding and decoding algorithms



$$n = 4, \quad E = \begin{Bmatrix} \{1,2\}, & \{2,2\} \\ \{2,4\}, & \{1,4\} \end{Bmatrix}$$

is the string `[[{}][[]{}{}]]`. Intuitively, pairs of brackets of the form {} can be interpreted as spaces between vertices, and then each set of matching brackets `[...]` encodes an arc that covers the spaces represented inside the brackets.

Any noncrossing ordered digraph $([1,\ldots,n],A)$ can be encoded with slight modifications to the algorithm. Instead of printing `[ ]` for an edge $\{i,j\} \in E_A$, $i \le j$, the algorithm should now print

$$\begin{array}{ll} \texttt{/ >} & \text{if } (i,j) \in A, (j,i) \notin A; \\ \texttt{< \textbackslash} & \text{if } (i,j) \notin A, (j,i) \in A; \\ \texttt{[ ]} & \text{if } (i,j), (j,i) \in A. \end{array}$$

In this way, we can simply encode the digraph $(\{1,2,3,4\}, \{(1,2),(2,2),(4,1),(4,2)\})$ as the string `</{}><[]{}{}\\`.

Proposition 2. *The encoding respects concatenation where the adjacent nonempty operands have a common vertex.*

Context-Freeness Arbitrary strings with balanced brackets form a context-free language that is known, generically, as a Dyck language. It is easy to see that the graphs **NC-GRAPH** are encoded with strings that belong to the Dyck language D_2 generated by the context-free grammar: $S \to \texttt{[}S\texttt{]}S \mid \{S\}S \mid \varepsilon$. The *encoded graphs*, $L_{\textbf{NC-GRAPH}}$, are, however, generated exactly by the context-free grammar $S \to \texttt{[}S'\texttt{]}\ S \mid \{\}\ S \mid \varepsilon$, $S' \to \texttt{[}S'\texttt{]}\ T \mid \{\}\ S$, $T \to \texttt{[}S'\texttt{]}\ S \mid \{\}\ S$. This language is an *unambiguous* context-free language.

Proposition 3. *The encoded graphs, $L_{\textbf{NC-GRAPH}}$, make an unambiguous context-free language.*

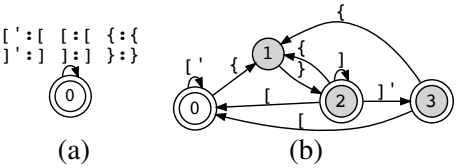
The practical significance of Proposition 3 is that there is a bijection between $L_{\textbf{NC-GRAPH}}$ and the derivation trees of a context-free grammar.

4.2 Bracketing Beyond the Encoding

Non-Derivational Representation A non-derivational representation for any context-free language L has been given by Chomsky and Schützenberger (1963). This replaces the stack with a Dyck language D and the grammar rules with co-occurrence patterns specified by a regular language *Reg*. To hide the internal alphabet from the strings of the represented language, there is a homomorphism that cleans the internal strings of *Reg* and D from internal markup to get actual strings of the target language:

$$L_{\textbf{NC-GRAPH}} = h(D \cap Reg).$$

To make this concrete, replace the previous context free grammar by $S'' \to \texttt{[}'S'\texttt{]}'\ S \mid \{\}\ S \mid \varepsilon$, $S \to \texttt{[}S'\texttt{]}\ S \mid \{\}\ S \mid \varepsilon$, $S' \to \texttt{[}'S'\texttt{]}'\ T \mid \{\}\ S$, $T \to \texttt{[}S'\texttt{]}\ S \mid \{\}\ S$. The homomorphism h (Fig. 3a) would now relate this language to the original language, mapping the string `['['{}]'[['{}]'{}]]'` to the string `[[{}][[{}]{}]]`, for example. The Dyck language $D = D_3$ checks that the internal brackets are balanced, and the regular component *Reg* (Fig. 3b) checks that the new brackets are used correctly. A similar representation for the language $L_{\textbf{NC-DIGRAPH}}$ of encoded digraphs can be obtained with straightforward extensions.



(a) (b)

Figure 3: The h and *Reg* components

The representation $L = h(D \cap Reg)$ is *unambiguous* if, for every word $w \in L$, the preimage $h^{-1}(w) \cap D \cap Reg$ is a single string. This implies that L is an *unambiguous* context-free language.

Proposition 4. *The set of encoded digraphs, $L_{\textbf{NC-DIGRAPH}}$, has an unambiguous representation.*

Proposition 5. *Let $L_i = h(D \cap R_i)$, $i \in \{0,1,2\}$ be unambiguous representations with $R_1, R_2 \subseteq R_0$. Then $L_3 = h(D \cap (R_1 \cap R_2))$ is an unambiguous context-free language and the same as $L_1 \cap L_2$.*

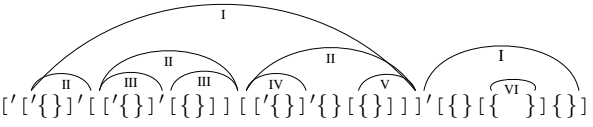
Proof. It is immediate that $L_3 \subseteq L_1 \cap L_2$ and L_3 is an unambiguous context-free language. To show that $L_1 \cap L_2 \subseteq L_3$, take an arbitrary $s \in L_1 \cap L_2$. Since $R_1, R_2 \subseteq R_0$ there is a unique $s' \in h^{-1}(s)$ such that $s' \in D \cap (R_1 \cap R_2)$. Thus $s \in L_3$. □

5 Latent Bracketing

In this section, we extend the internal strings of the non-derivational representation of $L_{\textbf{NC-DIGRAPH}}$ in

such a way that the configurations given in Proposition 1 can be detected locally from these.

Classification of Underlying Chains A *maximal linear chain* is a maximally long sequence of one or more edges that correspond to an underlying left-to-right path in the underlying graph in such a way that no edge in this chain is properly covered by an edge that does not properly cover all the edges in the chain. For example, the graph



contains six maximal linear chains, indicated with their Roman numbers on each arc. We decide nonlocal properties of noncrossing digraphs by recognizing maximal linear chains as parts of configurations presented in Proposition 1.

Every *loose* chain (like V and VI) starts with a bracket that is adjacent to a }-bracket. Such a chain can contribute only a covering edge to an underlying cycle. In contrast, a bracket with an apostrophe marks the beginning of a *non-loose* chain that can either start at the first vertex, or share starting point with a covering chain. When a nonloose chain is covered, it can be touched twice by a covering edge. The prefixes of chains are classified incrementally, from left to right, with a finite automaton (Figure 4). All states of the automaton are final and correspond to distinct classes of the chains. These classes are encoded to an extended set of brackets.

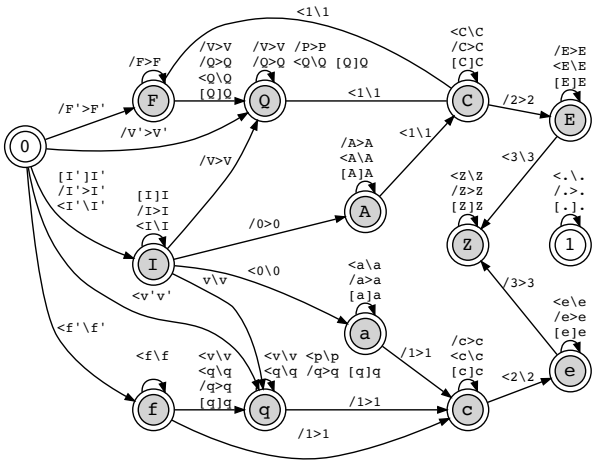


Figure 4: The finite automaton whose state 0 begins non-loose chains and state 1 loose chains

The automaton is symmetric: states with uppercase names are symmetrically related with corresponding lowercase states. Thus, it suffices to define the initial and uppercase-named states:

0	the initial state for a non-loose chain;
I	a bidirectional chain: $u \leftrightarrow (v \leftrightarrow)y$;
A	a primarily bidirectional forward chain: $u \leftrightarrow v \rightarrow y$;
F	a forward chain: $u \rightarrow v \rightarrow y$;
Q	a primarily forward chain: $u \rightarrow v \leftrightarrow (\cdots \rightarrow)y$;
C	a primarily forward 1-turn chain: $u \rightarrow v \leftarrow y$;
E	a primarily forward 2-turn chain: $u \rightarrow v \leftarrow x \rightarrow y$;
Z	a 3-turn chain;
1	the initial (and only) state for a loose chain;

Recognition of ambiguous paths in configurations $u\overleftrightarrow{\rightarrow\rightarrow}v \leftarrow y$ and $u\overleftarrow{\leftarrow v \rightarrow x \leftarrow \leftarrow}y$ involves three chain levels. To support the recognition, subtypes of edges are defined according to the chains they cover. The brackets >I', \I', >I, \I, \A, >a, \Q, >Q, >q,\q, >C, \c, \E, >e indicate edges that constitute a cycle with the chain they cover. The brackets >V', \v', >V, \v indicate edges that cover 2-turn chains. Not all states make these distinctions.

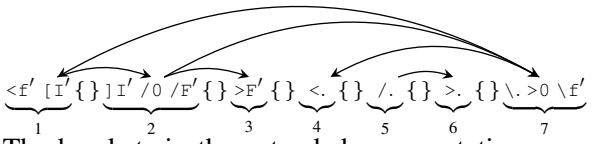
Extended Representation The extended brackets encode the latent structure of digraphs: the orientation and the subtype of the edge and the class of the chain. The total alphabet Σ of the strings now contains the boundary brackets {} and 54 pairs of brackets (Figure 4) for edges from which we obtain a new Dyck language, D_{55}, and an extended homomorphism h_{lat}.

The *Reg* component of the language representation is replaced with Reg_{lat}, that is, an intersection of (1) an inverse homomorphic image of *Reg* to strings over the extended alphabet, (2) a local language that constrains adjacent edges according to Figure 4, (3) a local language specifying how the chains start, and (4) a local language that distinguishes pure oriented edges from those that cover a cycle or a 2-turn chain. The new component requires only 24 states as a deterministic automaton.

Proposition 6. *$h_{lat}(D_{55} \cap Reg_{lat})$ is an unambiguous representation for $L_{\text{NC-DIGRAPH}}$.*

The internal language $L_{\text{NC-DIGRAPH}_{\text{lat}}} = D_{55} \cap Reg_{\text{lat}}$ is called the set of *latent encoded digraphs*.

Example Here is a digraph with its latent encoding:



The brackets in the extended representation contain information that helps us recognize, through local patterns, that this graph has a directed cycle

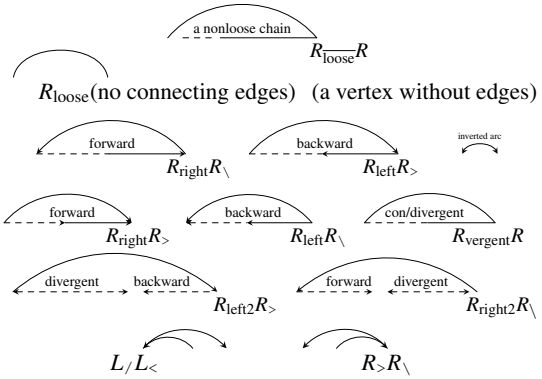
Forbidden patterns in noncrossing digraphs	Property	Constraint language
	**ACYC**$_{\text{U}}$	$A_U = \Sigma^* - \Sigma^* R_{\overline{\text{loose}}} R \Sigma^*$
	CONN$_{\text{W}}$	$C_W = \Sigma^* - \Sigma^* R_{\text{loose}}(\varepsilon \cup B\Sigma^*) - (B\Sigma^* \cup \Sigma^* B)$
	ACYC$_{\text{D}}$	$A_D = \Sigma^* - \Sigma^*(R_{\text{right}}R_{\setminus} \cup R_{\text{left}}R_{>} \cup \Sigma_{\text{inv}})\Sigma^*$
	UNAMB$_{\text{S}}$	$U_S = \Sigma^* - \Sigma^*(R_{\text{right}}R_{>} \cup R_{\text{left}}R_{\setminus} \cup R_{\text{vergent}}R)\Sigma^*$ $-\Sigma^*(R_{\text{left2}}R_{>} \cup R_{\text{right2}}R_{\setminus})\Sigma^*$
	PROJ$_{\text{W}}$	$P_W = \Sigma^* - \Sigma^*(L_{/}L_{<} \cup R_{>}R_{\setminus})\Sigma^*$
(an arc without inverse)	**INV**	$I = \Sigma^* - \Sigma^*\Sigma_{\text{or}}\Sigma^*$
(a state with more than 2 incoming arcs)	**OUT**	$Out = \Sigma^* - \Sigma^*\Sigma_{\text{in}}(\Sigma - B)^*\Sigma_{\text{in}}\Sigma^*$
(an inverted edge)	**ORIENTED**	$O = \Sigma^* - \Sigma^*\Sigma_{\text{inv}}\Sigma^*$

Table 1: Properties of encoded noncrossing digraphs as constraint languages

(directed path $1 \to 2 \to 7 \to 1$), is strongly ambiguous (two directed paths $2 \to 1$ and $2 \to 7 \to 1$) and is not weakly connected (vertices 5 and 6 are not connected to the rest of the digraph).

Expressing Properties via Forbidden Patterns We now demonstrate that all the mentioned nonlocal properties of graphs have become local in the extended internal representation of the code strings $L_{\text{NC-DIGRAPH}}$ for noncrossing digraphs.

These distinctive properties of graph families reduce to forbidden patterns in bracket strings and then compile into regular constraint languages. These are presented in Table 1. To keep the patterns simple, subsets of brackets are defined:

$L_{/}$	`[`-,/-brackets	$L_{<}$	`[`-,<-brackets
$R_{>}$	`]`-,>-brackets	$R_{\setminus}$	`]`-,\-brackets
B	{, }	R	$R_{>} \cup R_{\setminus}$
R_{loose}	}, `>.`, `\.`, `].`	$R_{\overline{\text{loose}}}$	$R - R_{\text{loose}}$
R_{right}	R reaching `F,Q,I,A`	R_{left}	R reaching `f,q,i,a`
R_{right2}	`>P`, `>2`, `>E`, `\E`, `]E`	R_{left2}	`\p`, `\2`, `\e`, `>e`, `]e`
Σ_{in}	$L_{<} \cup R_{>}$	$\overline{B}$	$\Sigma - B$
R_{vergent}	non-' R reaching `I,Q,q,A,a,C,c`		
Σ_{or}	all brackets for oriented edges		
Σ_{inv}	all brackets for inverted edges		

6 Validation Experiments

The current experiments were designed (1) to help in developing the components of Reg_{lat} and the constraint languages of axiomatic properties, (2) to validate the representation, the constraint languages and their unambiguity, (3) to learn about the ontology and (4) to sample the integer sequences associated with the cardinality of each family in the ontology.

Finding the Components Representations of Reg_{lat} were built with scripts written using a finite-state toolkit (Hulden, 2009) that supports rapid exploration with regular languages and transducers.

Validation of Languages Our scripts presented alternative approaches to compute languages of encoded digraphs with n vertices up to $n = 9$. We also implemented a Python script that enumerated elements of families of graphs up to $n = 6$. The solutions were used to cross-validate one another.

The constraint $G_n = \overline{B}^*(\{\}\overline{B}^*)^{n-1}$ ensures n-vertices in encoded digraphs. The finite set of encoded acyclic 5-vertex digraphs was computed with a finite-state approach (Yli-Jyrä et al., 2012) that takes the input projection of the composition

$$\text{Id}(Reg_{lat} \cap A_D \cap G_5) \circ T_{55} \circ T_{55} \circ T_{55} \circ T_{55} \circ T_{55} \circ \text{Id}(\varepsilon)$$

where Id defines an identity relation and transducer T_{55} eliminates matching adjacent brackets. This composition differs from the typical use where the purpose is to construct a regular relation (Kaplan and Kay, 1994) or its output projection (Roche, 1996; Oflazer, 2003).

For digraphs with a lot of vertices, we had an option to employ a dynamic programming scheme (Yli-Jyrä, 2012) that uses weighted transducers.

Building the Ontology To build the ontology in Figure 1 we first found out which combinations of digraph properties co-occur to define distinguishable families of digraphs. After the nodes of the

lattice were found, we were able to see the partial order between these.

Integer Sequences We sampled, for important families of digraphs, the prefixes of their related integer sequences. We found out that each family of graphs is pretty much described by its cardinality, see Table 2. In many cases, the number sequence was already well documented (OEIS Foundation Inc., 2017).

7 The Formal Basis of Practical Parsing

While presenting a practical parser implementation is outside of the scope of this paper, which focuses in the theory, we outline in this section the aspects to take into account when applying our representation to build practical natural language parsers.

Positioned Brackets In order to do inference in arc-factored parsing, we incorporate weights to the representation. For each vertex in G_n, the brackets are decorated with the respective position number. Then, we define an input-specific grammar representation where each pair of brackets in D gets an arc-factored weight given the directions and the vertex numbers associated with the brackets.

Grammar Intersection We associate, to each G_n, a quadratic-size context-free grammar that generates all noncrossing digraphs with n vertices. This grammar is obtained by computing (or even precomputing) the intersection $D_{55} \cap Reg_{\text{lat}} \cap G_n$ in any order, exploiting the closure of context-free languages under intersection with regular languages (Bar-Hillel et al., 1961). The introduction of the position numbers and weights in the Dyck language gives us, instead, a weighted grammar and its intersection (Lang, 1994). This grammar is a compact representation for a finite set of weighted latent encoded digraphs. Additional constraints during the intersection tailors the grammar to different families of digraphs.

Dynamic Programming The heaviest digraph is found with a dynamic programming algorithm that computes, for each nonterminal in the grammar, the weight of the heaviest subtree. A careful reader may notice some connections to Eisner algorithm (Eisner and Satta, 1999), context-free parsing through intersection (Nederhof and Satta, 2003), and a dynamic programming scheme that uses contracting transducers and factorized composition (Yli-Jyrä, 2012). Unfortunately, space does not permit discussing the connections here.

Lexicalized Search Space In practical parsing, we want the parser behavior and the dependency structure to be sensitive to the lexical entries or features of each word. We can replace the generic vertex description $\overline{B}^*$ in G_n with subsets that depend on respective lexical entries. Graphical constraints can be applied to some vertices but relaxed for others. This application of current results gives a principled, graphically motivated solution to lexicalized control over the search space.

8 Conclusion

We have investigated the search space of parsers that produce noncrossing digraphs. Parsers that can be adapted to different needs are less dependent on artificial assumptions on the search space. Adaptivity gives us freedom to model how the properties of digraphs are actually distributed in linguistic data. As the adaptive data analysis deserves to be treated in its own right, the current work focuses on the separation of the parsing algorithm from the properties of the search space.

This paper makes four significant contributions.

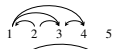
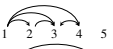
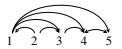
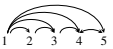
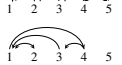
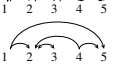
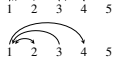
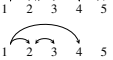
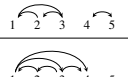
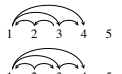
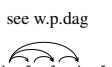
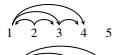
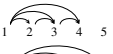
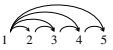
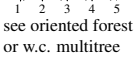
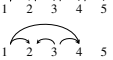
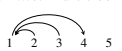
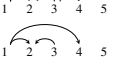
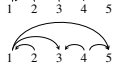
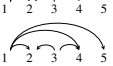
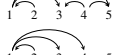
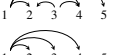
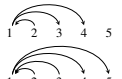
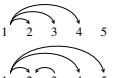
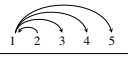
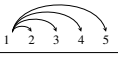
Contribution 1: Digraph Encoding The paper introduces, for noncrossing digraphs, an encoding that uses brackets to indicate edges.

Bracketed trees are widely used in generative syntax, treebanks and structured document formats. There are established conversions between phrase structure and projective dependency trees, but the currently advocated edge bracketing is expressive and captures more than just projective dependency trees. This capacity is welcome as syntactic and semantic analysis with dependency graphs is a steadily growing field.

The edge bracketing creates new avenues for the study of connections between noncrossing graphs and context-free languages, as well as their recognizable properties. By demonstrating that digraphs can be treated as strings, we suggest that practical parsing to these structures could be implemented with existing methods that restrict context-free grammars to a regular yield language.

Contribution 2: Context-Free Properties Acyclicity and other important properties of noncrossing digraphs are expressible as unambiguous context-free sets of encoded noncrossing

Table 2: Characterizations for some noncrossing families of digraphs and graphs

Name	Sequence prefix for $n = 2, 3, \ldots$	Example	Name	Sequence prefix for $n = 2, 3, \ldots$	Example
digraph	(**KJ**): 4,64,1792,**62464**,2437120,101859328 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat})}$		weakly projective digraph	4,36,480,**7744**,138880,2661376 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W)$	
w.c.digraph	3,54,1539,**53298**,2051406,84339468 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap C_W)$		w.p. w.c.digraph	3,26,339,**5278**,90686,1658772 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap C_W)$	
unamb.digr.	4,39,529,**8333**,142995,2594378 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap U_S)$		w.p. unamb.digr.	4,29,275,**3008**,35884,453489 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap U_S)$	
m-forest	4,37,469,**6871**,109369,1837396,32062711 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap A_U)$		w.p. m-forest	4,29,273,**2939**,34273,421336 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap A_U)$	
out digraph	4,27,207,**1683**,14229,123840,1102365 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap Out)$		w.p. out digraph	4,21,129,**867**,6177,45840,350379 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap Out)$	
or. digraph	3,27,405,**7533**,156735,3492639,77539113 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap O)$		w.p. or.digraph	see w.p.dag $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap O)$	see w.p.dag
dags	(**A246756**): 3,25,335,**5521**,101551 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap A_D)}$		w.p. dag	3,21,219,**2757**,38523, 574725, 8967675 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap A_D)$	
w.c. dag	(**KJ**): 2,18,242,**3890**,69074,1306466 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap A_D \cap C_W)}$		w.p. w.c. dag	2,14,142,**1706**,22554,316998,4480592 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap A_D \cap C_W)$	
multitree	3,19,167,**1721**,19447,233283,2917843 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap A_D \cap U_S)$	see oriented forest or w.c. multitree	w.p. multitree	3,17,129,**1139**,11005,112797,1203595 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap A_D \cap U_S)$	
or.forest	3,19,165,**1661**,18191,210407,2528777 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap A_D \cup A_U)$		w.p. or.forest	3,17,127,**1089**,10127,99329,1010189 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap A_D \cup A_U)$	
w.c. multitree	2,12,98,**930**,9638,105798,1201062 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap A_D \cap U_S \cap C_W)$		w.p. w.c. multitree	2,10,68,**538**,4650,42572,404354 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap A_D \cap U_S \cap C_W)$	
out or.forest	3,16,105,**756**,5738,45088,363221 $h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap A_D \cap Out)$		w.p. out or.forest	(**A003169**): 3,14,79,**494**,3294,22952 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap A_D \cap Out)}$	
polytree	(**A153231**): 2,12,96,**880**,8736,91392 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap A_D \cap C_W \cap A_U)}$		w.p. polytree	(**A027307**):2,10,66,**498**,4066,34970 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap A_D \cap C_W \cap A_U)}$	
out or.tree	(**A174687**): 2,9,48,**275**,1638,9996 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap A_D \cap C_W \cap Out)}$		projective out or.tree	(**A006013**): 2,7,30,**143**,728,3876,21318 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap P_W \cap A_D \cap C_W \cap Out)}$	
graph	(**A054726**): 2,8,48,**352**,2880,25216 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap I)}$		connected graph	(**A007297**): 1,4,23,**156**,1162,9192 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap I \cap C_W)}$	
forest	(**A054727**): 2,7,33,**181**,1083,6854 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap I \cap A_U)}$		tree	(**A001764**,**YJ**): 1,3,12,**55**,273,1428,7752 $\overline{h_{lat}(D_{55} \cap G_n \cap Reg_{lat} \cap I \cap A_U \cap C_W)}$	

A = (OEIS Foundation Inc., 2017), **KJ** = Kuhlmann (2015) or Kuhlmann and Johnsson (2015), **YJ** = Yli-Jyrä (2012)

digraphs. This facilitates the incorporation of property testing to dynamic programming algorithms that implement exact inference.

Descriptive complexity helps us understand to which degree various graphical properties are local and could be incorporated into efficient dynamic programming during exact inference. It is well known that acyclicity and connecticity are not definable in first-order logic (FO) while they can be defined easily in monadic second order logic (MSO) (Courcelle, 1997). MSO involves set-valued variables whose use in dynamic programming algorithms and tabular parsing is inefficient. MSO queries have a brute force transformation to first-order (FO) logic, but this does not generally help either as it is well known that MSO can express intractable problems.

The interesting observation of the current work is that some MSO definable properties of digraphs become local in our extended encoding. This encoding is linear compared to the size of digraphs: each string over the extended bracket alphabet encodes a fixed assignment of MSO variables. The properties of noncrossing digraphs now reduce to properties of bracketed trees with linear amount of latent information that is fixed for each digraph.

```
func noncrossing_ACYCU(n,E):
   for {u,y} in E and u < y:     # covering edge
      [v,p] = [u,u]
      while p != -1:             # chain continues
         [v,p] = [p,-1]
         for vv in [v+1,...,y]: # next vertex
            if {v,vv} in E and {v,vv} != {u,y}:
               if vv == y:
                  return False  # found cycle uvy
               p = vv           # find longest edge
   return True                  # acyclic
```

Figure 5: Testing $\mathbf{ACYC}_{\mathrm{U}}$ in logarithmic space

A deeper explanation for our observation comes from the fact that the treewidth of noncrossing and other outerplanar graphs is bounded to 2. When the treewidth is bounded, all MSO definable properties, including the intractable ones, become linear time decidable for individual structures (Courcelle, 1990). They can also be decided in a logarithmic amount of writable space (Elberfeld et al., 2010), e.g. with element indices instead of sets. By combining this insight with Proposition 1, we obtain a logspace solution for testing acyclicity of a noncrossing graph (Figure 5).

Although bounded treewidth is a weaker constraint than so-called bounded treedepth that would immediately guarantee first-order definabil-

ity (Elberfeld et al., 2016), it can sometimes turn intractable search problems to dynamic programming algorithms (Akutsu and Tamura, 2012). In our case, Proposition 1 gave rise to unambiguous context-free subsets of $L_{\text{NC-DIGRAPH}}$. These can be recognized with dynamic programming and used in efficient constrained inference when we add vertex indices to the brackets and weights to the grammar of the corresponding Dyck language.

Contribution 3: Digraph Ontology The context-free properties of encoded digraphs have elegant nonderivative language representations and they generate a semi-lattice under language intersection. Although context-free languages are not generally closed under intersection, all combinations of the properties in this lattice are context-free and define natural families of digraphs. The nonderivative representations for our axiomatic properties share the same Dyck language D_{55} and homomorphism, but differ in terms of forbidden patterns. As a consequence, also any conjunctive combination of these two properties shares these components and thus define a context-free language. The obtained semilattice is an ontology of families of noncrossing digraphs.

Our ontology contains important families of noncrossing digraphs used in syntactic and semantic dependency parsing: out trees, dags, and weakly connected digraphs. It shows the entailment between the properties and proves the existence of less known families of noncrossing digraphs such as strongly unambiguous digraphs and oriented graphs, multitrees, oriented forests and polytrees. These are generalizations of out oriented trees. However, these families can still be weakly projective. Table 2 shows integer sequences obtained by enumerating digraphs in each family. At least twelve of these sequences are previously known, which indicates that the families are natural.

We used a finite-state toolkit to build the components of the nongenerative language representation for latent encoded digraphs and the axioms.[2]

Contribution 4: Generic Parsing The fourth contribution of this paper is to show that parsing algorithms can be separated from the formal properties of their search space.

All the presented families of digraphs can be treated by parsers and other algorithms (e.g. enumeration algorithms) in a uniform manner. The parser's inference rules can stay constant and the choice of the search space is made by altering the regular component of the language representation.

The ontology of the search space can be combined with a constraint relaxation strategy, for example, when an out tree is a preferred analysis, but a dag is also possible as an analysis when no tree is strong enough. The flexibility applies also to dynamic programming algorithms that complement with the encoding and allow inference of best dependency graphs in a family simply by intersection with a weighted CFG grammar for a Dyck language that models position-indexed edges of the complete digraph.

Since the families of digraphs are distinguished by forbidden local patterns, the choice of search space can be made purely on lexical grounds, blending well with lexicalized parsing and allowing possibilities such as choosing, per each word, what kind of structures the word can go with.

Future work We are planning to extend the coverage of the approach by exploring 1-endpoint-crossing and MH_k trees (Pitler et al., 2013; Gómez-Rodríguez, 2016), and related digraphs — see (Yli-Jyrä, 2004; Gómez-Rodríguez et al., 2011). Properties such as *weakly projective*, *out*, and *strongly unambiguous* prompt further study.

An interesting avenue for future work is to explore higher order factorizations for noncrossing digraphs and the related inference. We would also like to have more insight on the transformation of MSO definable properties to the current framework and to logspace algorithms.

Acknowledgements

AYJ has received funding as Research Fellow from the Academy of Finland (dec. No 270354 - A Usable Finite-State Model for Adequate Syntactic Complexity) and Clare Hall Fellow from the University of Helsinki (dec. RP 137/2013). CGR has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714150 - FASTPARSE) and from the TELEPARES-UDC project (FFI2014-51978-C2-2-R) from MINECO. The comments of Juha Kontinen, Mark-Jan Nederhof and the anonymous reviewers helped to improve the paper.

[2]The finite-state toolkit scripts and a Python-based graph enumerator are available at https://github.com/amikael/ncdigraphs .

References

Tatsuya Akutsu and Takeyuki Tamura. 2012. A polynomial-time algorithm for computing the maximum common subgraph of outerplanar graphs of bounded degree. In Branislav Rovan, Vladimiro Sassone, and Peter Widmayer, editors, *Mathematical Foundations of Computer Science 2012: 37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27-31, 2012. Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 76–87. https://doi.org/10.1007/978-3-642-32589-2_10.

Jason Baldridge and Geert-Jan M. Kruijff. 2003. Multi-modal combinatory categorial grammar. In *Proceedings of EACL'03: the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*. Association for Computational Linguistics, Budapest, Hungary, pages 211–218. https://doi.org/10.3115/1067807.1067836.

Yehoshua Bar-Hillel, Micha Perles, and Eliahu Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonologie, Sprachwissenschaft und Kommunikationsforschung* 14:113–124.

Noam Chomsky and Marcel-Paul Schützenberger. 1963. The algebraic theory of context-free languages. *Computer Programming and Formal Systems* pages 118–161.

Bruno Courcelle. 1990. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation* 85(1):12 – 75. https://doi.org/10.1016/0890-5401(90)90043-H.

Bruno Courcelle. 1997. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformations*, World Scientific, New-Jersey, London, volume 1, chapter 5, pages 313–400.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*. Copenhagen, Denmark, pages 340–345. http://aclweb.org/anthology/C/C96/C96-1058.pdf.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and Head Automaton Grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, College Park, Maryland, USA, pages 457–464. https://doi.org/10.3115/1034678.1034748.

Michael Elberfeld, Martin Grohe, and Till Tantau. 2016. Where first-order and monadic second-order logic coincide. *ACM Trans. Comput. Logic* 17(4):25:1–25:18. https://doi.org/10.1145/2946799.

Michael Elberfeld, Andreas Jakoby, and Till Tantau. 2010. Logspace versions of the theorems of Bodlaender and Courcelle. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, Washington, DC, USA, FOCS '10, pages 143–152. https://doi.org/10.1109/FOCS.2010.21.

Carlos Gómez-Rodríguez. 2016. Restricted non-projectivity: Coverage vs. efficiency. *Computational Linguistics* 42(4):809–817. https://doi.org/10.1162/COLI_a_00267.

Carlos Gómez-Rodríguez, John A. Carroll, and David J. Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational Linguistics* 37(3):541–586. https://doi.org/10.1162/COLI_a_00060.

Erich Grädel, P. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, Moshe Y. Vardi, Y. Venema, and Scott Weinstein. 2005. *Finite Model Theory and Its Applications (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Sheila Greibach. 1973. The hardest context-free language. *SIAM Journal on Computing* 2(4):304–310. https://doi.org/10.1137/0202025.

Venkatesan Guruswami, Johan Håstad, Rajsekar Manokaran, Prasad Raghavendra, and Moses Charikar. 2011. Beating the random ordering is hard: Every ordering CSP is approximation resistant. *SIAM Journal on Computing* 40(3):878914. https://doi.org/10.1137/090756144.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the Demonstrations Session at EACL 2009*. Association for Computational Linguistics, Athens, Greece, pages 29–32. http://www.aclweb.org/anthology/E09-2008.

Mans Hulden. 2011. Parsing CFGs and PCFGs with a Chomsky-Schützenberger representation. In Zygmunt Vetulani, editor, *Human Language Technology. Challenges for Computer Science and Linguistics: 4th Language and Technology Conference, LTC 2009, Poznan, Poland, November 6-8, 2009, Revised Selected Papers*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 151–160. https://doi.org/10.1007/978-3-642-20095-3_14.

Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20(3):331–378. http://dl.acm.org/citation.cfm?id=204915.204917.

Marco Kuhlmann. 2015. Tabulation of noncrossing acyclic digraphs. arXiv:1504.04993. https://arxiv.org/abs/1504.04993.

Marco Kuhlmann and Peter Johnsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics* 3:559–570. http://aclweb.org/anthology/Q/Q15/Q15-1040.pdf.

Marco Kuhlmann and Stephan Oepen. 2016. Towards a catalogue of linguistic graph banks. *Computational Linguistics* 42(4):819–827. https://doi.org/10.1162/COLI_a_00268.

Bernard Lang. 1994. Recognition can be harder than parsing. *Computational Intelligence* 10(4):486–494. http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8640.1994.tb00011.x/full.

Klaus-Jörn Lange. 1997. An unambiguous class possessing a complete set. In Morvan Reischuk, editor, *STACKS'97 Proceedings*. Springer, volume 1200 of *Lecture Notes in Computer Science*. http://dl.acm.org/citation.cfm?id=695352.

S. Marcus. 1967. *Algebraic Linguistics; Analytical Models*, volume 29 of *Mathematics in Science and Engineering*. Academic Press, New York and London.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Vancouver, British Columbia, Canada, pages 523–530. http://www.aclweb.org/anthology/H/H05/H05-1066.pdf.

Mark-Jan Nederhof and Giorgio Satta. 2003. Probabilistic parsing as intersection. In *8th International Workshop on Parsing Technologies*. LORIA, Nancy, France, pages 137–148.

OEIS Foundation Inc. 2017. The on-line encyclopedia of integer sequences. `http://oeis.org`, read on 15 January 2017.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 915–926. http://www.aclweb.org/anthology/S15-2153.

Kemal Oflazer. 2003. Dependency parsing with an extended finite-state approach. *Computational Linguistics* 29(4):515–544. https://doi.org/10.1162/089120103322753338.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2013. Finding optimal 1-endpoint-crossing trees. *Transactions of the Association for Computational Linguistics* 1:13–24. http://aclweb.org/anthology/Q13-1002.

George Rebane and Judea Pearl. 1987. The recovery of causal poly-trees from statistical data. In *Proceedings of the 3rd Annual Conference on Uncertainty in Artificial Intelligence (UAI 1987)*. Seattle, WA, pages 222–228. http://dl.acm.org/citation.cfm?id=3023784.

Emmanuel Roche. 1996. Transducer parsing of free and frozen sentences. *Natural Language Engineering* 2(4):345–350. https://doi.org/10.1017/S1351324997001605.

Natalie Schluter. 2014. On maximum spanning DAG algorithms for semantic DAG parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*. Association for Computational Linguistics, Baltimore, MD, pages 61–65. http://www.aclweb.org/anthology/W/W14/W14-2412.pdf.

Natalie Schluter. 2015. The complexity of finding the maximum spanning DAG and other restrictions for DAG parsing of natural language. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, Denver, Colorado, pages 259–268. http://www.aclweb.org/anthology/S15-1031.

Anssi Yli-Jyrä. 2004. Axiomatization of restricted non-projective dependency trees through finite-state constraints that analyse crossing bracketings. In Geert-Jan M. Kruijff and Denys Duchier, editors, *COLING 2004 Recent Advances in Dependency Grammar*. COLING, Geneva, Switzerland, pages 25–32. https://www.aclweb.org/anthology/W/W04/W04-1504.pdf.

Anssi Yli-Jyrä. 2005. Approximating dependency grammars through intersection of star-free regular languages. *Int. J. Found. Comput. Sci.* 16(3):565–579. https://doi.org/10.1142/S0129054105003169.

Anssi Yli-Jyrä. 2012. On dependency analysis via contractions and weighted FSTs. In Diana Santos, Krister Lindén, and Wanjiku Ng'ang'a, editors, *Shall We Play the Festschrift Game?, Essays on the Occasion of Lauri Carlson's 60th Birthday*. Springer, pages 133–158. https://doi.org/10.1007/978-3-642-30773-7_10.

Anssi Yli-Jyrä, Jussi Piitulainen, and Atro Voutilainen. 2012. Refining the design of a contracting finite-state dependency parser. In Iñaki Alegria and Mans Hulden, editors, *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*. Association for Computational Linguistics, Donostia–San Sebastián, Spain, pages 108–115. http://www.aclweb.org/anthology/W12-6218.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 188–193. http://www.aclweb.org/anthology/P11-2033.

Semi-supervised sequence tagging with bidirectional language models

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, Russell Power
Allen Institute for Artificial Intelligence
{matthewp,waleeda,chandrab,russellp}@allenai.org

Abstract

Pre-trained word embeddings learned from unlabeled text have become a standard component of neural network architectures for NLP tasks. However, in most cases, the recurrent network that operates on word-level representations to produce context sensitive representations is trained on relatively little labeled data. In this paper, we demonstrate a general semi-supervised approach for adding pre-trained context embeddings from bidirectional language models to NLP systems and apply it to sequence labeling tasks. We evaluate our model on two standard datasets for named entity recognition (NER) and chunking, and in both cases achieve state of the art results, surpassing previous systems that use other forms of transfer or joint learning with additional labeled data and task specific gazetteers.

1 Introduction

Due to their simplicity and efficacy, pre-trained word embedding have become ubiquitous in NLP systems. Many prior studies have shown that they capture useful semantic and syntactic information (Mikolov et al., 2013; Pennington et al., 2014) and including them in NLP systems has been shown to be enormously helpful for a variety of downstream tasks (Collobert et al., 2011).

However, in many NLP tasks it is essential to represent not just the meaning of a word, but also the word in context. For example, in the two phrases "A Central Bank spokesman" and "The Central African Republic", the word 'Central' is used as part of both an Organization and Location. Accordingly, current state of the art sequence tagging models typically include a bidirectional recurrent neural network (RNN) that encodes token sequences into a context sensitive representation before making token specific predictions (Yang et al., 2017; Ma and Hovy, 2016; Lample et al., 2016; Hashimoto et al., 2016).

Although the token representation is initialized with pre-trained embeddings, the parameters of the bidirectional RNN are typically learned only on labeled data. Previous work has explored methods for jointly learning the bidirectional RNN with supplemental labeled data from other tasks (e.g., Søgaard and Goldberg, 2016; Yang et al., 2017).

In this paper, we explore an alternate semi-supervised approach which does not require additional labeled data. We use a neural language model (LM), pre-trained on a large, unlabeled corpus to compute an encoding of the context at each position in the sequence (hereafter an *LM embedding*) and use it in the supervised sequence tagging model. Since the LM embeddings are used to compute the probability of future words in a neural LM, they are likely to encode both the semantic and syntactic roles of words in context.

Our main contribution is to show that the context sensitive representation captured in the LM embeddings is useful in the supervised sequence tagging setting. When we include the LM embeddings in our system overall performance increases from 90.87% to 91.93% F_1 for the CoNLL 2003 NER task, a more then 1% absolute F1 increase, and a substantial improvement over the previous state of the art. We also establish a new state of the art result (96.37% F_1) for the CoNLL 2000 Chunking task.

As a secondary contribution, we show that using both forward and backward LM embeddings boosts performance over a forward only LM. We also demonstrate that domain specific pre-training is not necessary by applying a LM trained in the news domain to scientific papers.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1756–1765
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1161

2 Language model augmented sequence taggers (TagLM)

2.1 Overview

The main components in our language-model-augmented sequence tagger (TagLM) are illustrated in Fig. 1. After pre-training word embeddings and a neural LM on large, unlabeled corpora (Step 1), we extract the word and LM embeddings for every token in a given input sequence (Step 2) and use them in the supervised sequence tagging model (Step 3).

2.2 Baseline sequence tagging model

Our baseline sequence tagging model is a hierarchical neural tagging model, closely following a number of recent studies (Ma and Hovy, 2016; Lample et al., 2016; Yang et al., 2017; Chiu and Nichols, 2016) (left side of Figure 2).

Given a sentence of tokens $(t_1, t_2, \ldots, t_N)$ it first forms a representation, $\mathbf{x}_k$, for each token by concatenating a character based representation $\mathbf{c}_k$ with a token embedding $\mathbf{w}_k$:

$$\begin{aligned} \mathbf{c}_k &= C(t_k; \theta_c) \\ \mathbf{w}_k &= E(t_k; \theta_w) \\ \mathbf{x}_k &= [\mathbf{c}_k; \mathbf{w}_k] \end{aligned} \quad (1)$$

The character representation $\mathbf{c}_k$ captures morphological information and is either a convolutional neural network (CNN) (Ma and Hovy, 2016; Chiu and Nichols, 2016) or RNN (Yang et al., 2017; Lample et al., 2016). It is parameterized by $C(\cdot, \theta_c)$ with parameters θ_c. The token embeddings, $\mathbf{w}_k$, are obtained as a lookup $E(\cdot, \theta_w)$, initialized using pre-trained word embeddings, and fine tuned during training (Collobert et al., 2011).

To learn a context sensitive representation, we employ multiple layers of bidirectional RNNs. For each token position, k, the hidden state $\mathbf{h}_{k,i}$ of RNN layer i is formed by concatenating the hidden states from the forward ($\overrightarrow{\mathbf{h}}_{k,i}$) and backward ($\overleftarrow{\mathbf{h}}_{k,i}$) RNNs. As a result, the bidirectional RNN is able to use both past and future information to make a prediction at token k. More formally, for the first RNN layer that operates on $\mathbf{x}_k$ to output $\mathbf{h}_{k,1}$:

$$\begin{aligned} \overrightarrow{\mathbf{h}}_{k,1} &= \overrightarrow{R}_1(\mathbf{x}_k, \overrightarrow{\mathbf{h}}_{k-1,1}; \theta_{\overrightarrow{R}_1}) \\ \overleftarrow{\mathbf{h}}_{k,1} &= \overleftarrow{R}_1(\mathbf{x}_k, \overleftarrow{\mathbf{h}}_{k+1,1}; \theta_{\overleftarrow{R}_1}) \\ \mathbf{h}_{k,1} &= [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}] \end{aligned} \quad (2)$$

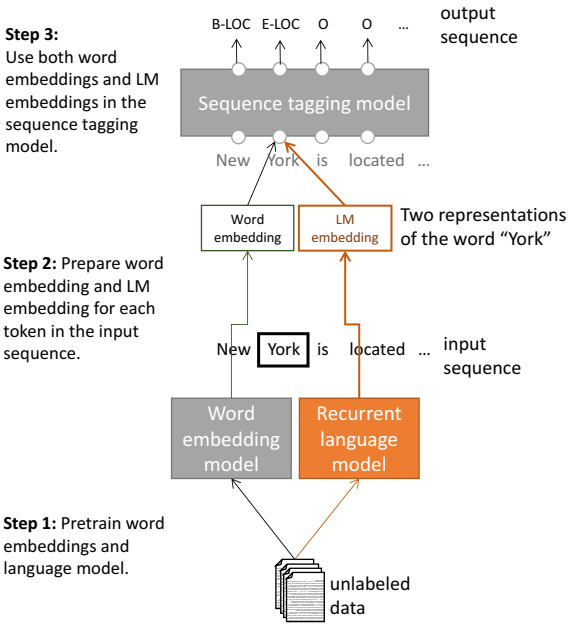


Figure 1: The main components in TagLM, our language-model-augmented sequence tagging system. The language model component (in orange) is used to augment the input token representation in a traditional sequence tagging models (in grey).

The second RNN layer is similar and uses $\mathbf{h}_{k,1}$ to output $\mathbf{h}_{k,2}$. In this paper, we use $L = 2$ layers of RNNs in all experiments and parameterize R_i as either Gated Recurrent Units (GRU) (Cho et al., 2014) or Long Short-Term Memory units (LSTM) (Hochreiter and Schmidhuber, 1997) depending on the task.

Finally, the output of the final RNN layer $\mathbf{h}_{k,L}$ is used to predict a score for each possible tag using a single dense layer. Due to the dependencies between successive tags in our sequence labeling tasks (e.g. using the BIOES labeling scheme, it is not possible for `I-PER` to follow `B-LOC`), it is beneficial to model and decode each sentence jointly instead of independently predicting the label for each token. Accordingly, we add another layer with parameters for each label bigram, computing the sentence conditional random field (CRF) loss (Lafferty et al., 2001) using the forward-backward algorithm at training time, and using the Viterbi algorithm to find the most likely tag sequence at test time, similar to Collobert et al. (2011).

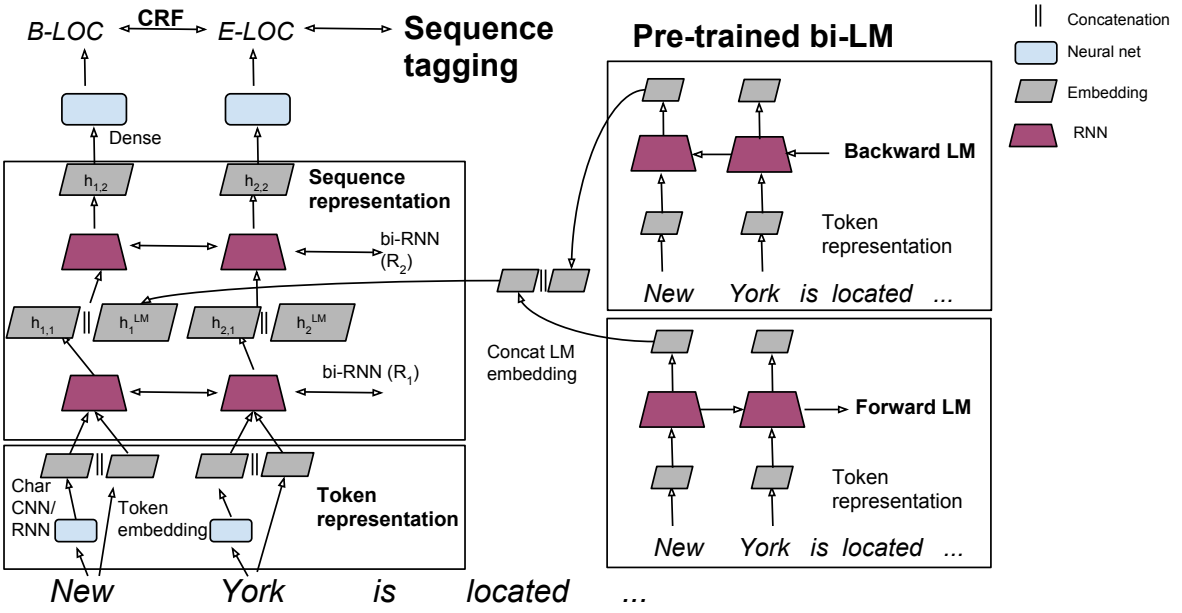


Figure 2: Overview of TagLM, our language model augmented sequence tagging architecture. The top level embeddings from a pre-trained bidirectional LM are inserted in a stacked bidirectional RNN sequence tagging model. See text for details.

2.3 Bidirectional LM

A language model computes the probability of a token sequence $(t_1, t_2, \ldots, t_N)$

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1}).$$

Recent state of the art neural language models (Józefowicz et al., 2016) use a similar architecture to our baseline sequence tagger where they pass a token representation (either from a CNN over characters or as token embeddings) through multiple layers of LSTMs to embed the history $(t_1, t_2, \ldots, t_k)$ into a fixed dimensional vector $\overrightarrow{\mathbf{h}}_k^{LM}$. This is the *forward LM embedding* of the token at position k and is the output of the top LSTM layer in the language model. Finally, the language model predicts the probability of token t_{k+1} using a softmax layer over words in the vocabulary.

The need to capture future context in the LM embeddings suggests it is beneficial to also consider a *backward* LM in additional to the traditional *forward* LM. A backward LM predicts the previous token given the future context. Given a sentence with N tokens, it computes

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N).$$

A backward LM can be implemented in an analogous way to a forward LM and produces the *backward LM embedding* $\overleftarrow{\mathbf{h}}_k^{LM}$, for the sequence $(t_k, t_{k+1}, \ldots, t_N)$, the output embeddings of the top layer LSTM.

In our final system, after pre-training the forward and backward LMs separately, we remove the top layer softmax and concatenate the forward and backward LM embeddings to form bidirectional LM embeddings, i.e., $\mathbf{h}_k^{LM} = [\overrightarrow{\mathbf{h}}_k^{LM}; \overleftarrow{\mathbf{h}}_k^{LM}]$. Note that in our formulation, the forward and backward LMs are independent, without any shared parameters.

2.4 Combining LM with sequence model

Our combined system, TagLM, uses the LM embeddings as additional inputs to the sequence tagging model. In particular, we concatenate the LM embeddings $\mathbf{h}^{LM}$ with the output from one of the bidirectional RNN layers in the sequence model. In our experiments, we found that introducing the LM embeddings at the output of the first layer performed the best. More formally, we simply replace (2) with

$$\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}]. \quad (3)$$

There are alternate possibilities for adding the LM embeddings to the sequence model. One pos-

sibility adds a non-linear mapping after the concatenation and before the second RNN (e.g. replacing (3) with $f([\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}])$ where f is a non-linear function). Another possibility introduces an attention-like mechanism that weights the all LM embeddings in a sentence before including them in the sequence model. Our initial results with the simple concatenation were encouraging so we did not explore these alternatives in this study, preferring to leave them for future work.

3 Experiments

We evaluate our approach on two well benchmarked sequence tagging tasks, the CoNLL 2003 NER task (Sang and Meulder, 2003) and the CoNLL 2000 Chunking task (Sang and Buchholz, 2000). We report the official evaluation metric (micro-averaged F_1). In both cases, we use the BIOES labeling scheme for the output tags, following previous work which showed it outperforms other options (e.g., Ratinov and Roth, 2009). Following Chiu and Nichols (2016), we use the Senna word embeddings (Collobert et al., 2011) and pre-processed the text by lowercasing all tokens and replacing all digits with 0.

CoNLL 2003 NER. The CoNLL 2003 NER task consists of newswire from the Reuters RCV1 corpus tagged with four different entity types (`PER`, `LOC`, `ORG`, `MISC`). It includes standard train, development and test sets. Following previous work (Yang et al., 2017; Chiu and Nichols, 2016) we trained on both the train and development sets after tuning hyperparameters on the development set.

The hyperparameters for our baseline model are similar to Yang et al. (2017). We use two bidirectional GRUs with 80 hidden units and 25 dimensional character embeddings for the token character encoder. The sequence layer uses two bidirectional GRUs with 300 hidden units each. For regularization, we add 25% dropout to the input of each GRU, but not to the recurrent connections.

CoNLL 2000 chunking. The CoNLL 2000 chunking task uses sections 15-18 from the Wall Street Journal corpus for training and section 20 for testing. It defines 11 syntactic chunk types (e.g., `NP`, `VP`, `ADJP`) in addition to `other`. We randomly sampled 1000 sentences from the training set as a held-out development set.

The baseline sequence tagger uses 30 dimensional character embeddings and a CNN with 30 filters of width 3 characters followed by a tanh non-linearity for the token character encoder. The sequence layer uses two bidirectional LSTMs with 200 hidden units. Following Ma and Hovy (2016) we added 50% dropout to the character embeddings, the input to each LSTM layer (but not recurrent connections) and to the output of the final LSTM layer.

Pre-trained language models. The primary bidirectional LMs we used in this study were trained on the 1B Word Benchmark (Chelba et al., 2014), a publicly available benchmark for large-scale language modeling. The training split has approximately 800 million tokens, about a 4000X increase over the number training tokens in the CoNLL datasets. Józefowicz et al. (2016) explored several model architectures and released their best single model and training recipes. Following Sak et al. (2014), they used linear projection layers at the output of each LSTM layer to reduce the computation time but still maintain a large LSTM state. Their single best model took three weeks to train on 32 GPUs and achieved 30.0 test perplexity. It uses a character CNN with 4096 filters for input, followed by two stacked LSTMs, each with 8192 hidden units and a 1024 dimensional projection layer. We use `CNN-BIG-LSTM` to refer to this language model in our results.

In addition to `CNN-BIG-LSTM` from Józefowicz et al. (2016),[1] we used the same corpus to train two additional language models with fewer parameters: forward `LSTM-2048-512` and backward `LSTM-2048-512`. Both language models use token embeddings as input to a single layer LSTM with 2048 units and a 512 dimension projection layer. We closely followed the procedure outlined in Józefowicz et al. (2016), except we used synchronous parameter updates across four GPUs instead of asynchronous updates across 32 GPUs and ended training after 10 epochs. The test set perplexities for our forward and backward `LSTM-2048-512` language models are 47.7 and 47.3, respectively.[2]

[1] `https://github.com/tensorflow/models/tree/master/lm_1b`

[2] Due to different implementations, the perplexity of the forward LM with similar configurations in Józefowicz et al. (2016) is different (45.0 vs. 47.7).

Model	$F_1 \pm$ std
Chiu and Nichols (2016)	90.91 ± 0.20
Lample et al. (2016)	90.94
Ma and Hovy (2016)	91.37
Our baseline without LM	90.87 ± 0.13
TagLM	$\mathbf{91.93 \pm 0.19}$

Table 1: Test set F_1 comparison on CoNLL 2003 NER task, using only CoNLL 2003 data and unlabeled text.

Model	$F_1 \pm$ std
Yang et al. (2017)	94.66
Hashimoto et al. (2016)	95.02
Søgaard and Goldberg (2016)	95.28
Our baseline without LM	95.00 ± 0.08
TagLM	$\mathbf{96.37 \pm 0.05}$

Table 2: Test set F_1 comparison on CoNLL 2000 Chunking task using only CoNLL 2000 data and unlabeled text.

Training. All experiments use the Adam optimizer (Kingma and Ba, 2015) with gradient norms clipped at 5.0. In all experiments, we fine tune the pre-trained Senna word embeddings but fix all weights in the pre-trained language models. In addition to explicit dropout regularization, we also use early stopping to prevent over-fitting and use the following process to determine when to stop training. We first train with a constant learning rate $\alpha = 0.001$ on the training data and monitor the development set performance at each epoch. Then, at the epoch with the highest development performance, we start a simple learning rate annealing schedule: decrease α an order of magnitude (i.e., divide by ten), train for five epochs, decrease α an order of magnitude again, train for five more epochs and stop.

Following Chiu and Nichols (2016), we train each final model configuration ten times with different random seeds and report the mean and standard deviation F_1. It is important to estimate the variance of model performance since the test data sets are relatively small.

3.1 Overall system results

Tables 1 and 2 compare results from TagLM with previously published state of the art results without additional labeled data or task specific gazetteers. Tables 3 and 4 compare results of TagLM to other systems that include additional labeled data or gazetteers. In both tasks, TagLM establishes a new state of the art using bidirectional LMs (the forward `CNN-BIG-LSTM` and the backward `LSTM-2048-512`).

In the CoNLL 2003 NER task, our model scores 91.93 mean F_1, which is a statistically significant increase over the previous best result of 91.62 $\pm$0.33 from Chiu and Nichols (2016) that used gazetteers (at 95%, two-sided Welch t-test, $p = 0.021$).

In the CoNLL 2000 Chunking task, TagLM achieves 96.37 mean F_1, exceeding all previously published results without additional labeled data by more then 1% absolute F_1. The improvement over the previous best result of 95.77 in Hashimoto et al. (2016) that jointly trains with Penn Treebank (PTB) POS tags is statistically significant at 95% ($p < 0.001$ assuming standard deviation of 0.1).

Importantly, the LM embeddings amounts to an average absolute improvement of 1.06 and 1.37 F_1 in the NER and Chunking tasks, respectively.

Adding external resources. Although we do not use external labeled data or gazetteers, we found that TagLM outperforms previous state of the art results in both tasks when external resources (labeled data or task specific gazetteers) are available. Furthermore, Tables 3 and 4 show that, in most cases, the improvements we obtain by adding LM embeddings are larger then the improvements previously obtained by adding other forms of transfer or joint learning. For example, Yang et al. (2017) noted an improvement of only 0.06 F_1 in the NER task when transfer learning from both CoNLL 2000 chunks and PTB POS tags and Chiu and Nichols (2016) reported an increase of 0.71 F_1 when adding gazetteers to their baseline. In the Chunking task, previous work has reported from 0.28 to 0.75 improvement in F_1 when including supervised labels from the PTB POS tags or CoNLL 2003 entities (Yang et al., 2017; Søgaard and Goldberg, 2016; Hashimoto et al., 2016).

3.2 Analysis

To elucidate the characteristics of our LM augmented sequence tagger, we ran a number of additional experiments on the CoNLL 2003 NER task.

How to use LM embeddings? In this experiment, we concatenate the LM embeddings at dif-

Model	External resources	F_1 Without	F_1 With	Δ
Yang et al. (2017)	transfer from CoNLL 2000/PTB-POS	91.2	91.26	+0.06
Chiu and Nichols (2016)	with gazetteers	90.91	91.62	+0.71
Collobert et al. (2011)	with gazetteers	88.67	89.59	+0.92
Luo et al. (2015)	joint with entity linking	89.9	91.2	**+1.3**
Ours	no LM vs TagLM *unlabeled data only*	90.87	**91.93**	+1.06

Table 3: Improvements in test set F_1 in CoNLL 2003 NER when including additional labeled data or task specific gazetteers (except the case of TagLM where we do not use additional labeled resources).

Model	External resources	F_1 Without	F_1 With	Δ
Yang et al. (2017)	transfer from CoNLL 2003/PTB-POS	94.66	95.41	+0.75
Hashimoto et al. (2016)	jointly trained with PTB-POS	95.02	95.77	+0.75
Søgaard and Goldberg (2016)	jointly trained with PTB-POS	95.28	95.56	+0.28
Ours	no LM vs TagLM *unlabeled data only*	95.00	**96.37**	**+1.37**

Table 4: Improvements in test set F_1 in CoNLL 2000 Chunking when including additional labeled data (except the case of TagLM where we do not use additional labeled data).

Use LM embeddings at	$F_1 \pm$ std
input to the first RNN layer	91.55 ± 0.21
output of the first RNN layer	$\mathbf{91.93 \pm 0.19}$
output of the second RNN layer	91.72 ± 0.13

Table 5: Comparison of CoNLL-2003 test set F_1 when the LM embeddings are included at different layers in the baseline tagger.

ferent locations in the baseline sequence tagger. In particular, we used the LM embeddings $\mathbf{h}_k^{LM}$ to:

- augment the *input* of the *first* RNN layer; i.e., $\mathbf{x}_k = [\mathbf{c}_k; \mathbf{w}_k; \mathbf{h}_k^{LM}]$,
- augment the *output* of the *first* RNN layer; i.e., $\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}]$,[3] and
- augment the *output* of the *second* RNN layer; i.e., $\mathbf{h}_{k,2} = [\overrightarrow{\mathbf{h}}_{k,2}; \overleftarrow{\mathbf{h}}_{k,2}; \mathbf{h}_k^{LM}]$.

Table 5 shows that the second alternative performs best. We speculate that the second RNN layer in the sequence tagging model is able to capture interactions between task specific context as expressed in the first RNN layer and general context as expressed in the LM embeddings in a way that improves overall system performance. These results are consistent with Søgaard and Goldberg (2016) who found that chunking performance was sensitive to the level at which additional POS supervision was added.

Does it matter which language model to use? In this experiment, we compare six different configurations of the forward and backward language models (including the baseline model which does not use any language models). The results are reported in Table 6.

We find that adding backward LM embeddings consistently outperforms forward-only LM embeddings, with F_1 improvements between 0.22 and 0.27%, even with the relatively small backward `LSTM-2048-512` LM.

LM size is important, and replacing the forward `LSTM-2048-512` with `CNN-BIG-LSTM` (test perplexities of 47.7 to 30.0 on 1B Word Benchmark) improves F_1 by 0.26 - 0.31%, about as much as adding backward LM. Accordingly, we hypothesize (but have not tested) that replacing the backward `LSTM-2048-512` with a backward LM analogous to the `CNN-BIG-LSTM` would further improve performance.

To highlight the importance of including language models trained on a large scale data, we also experimented with training a language model on just the CoNLL 2003 training and development data. Due to the much smaller size of this data

[3]This configuration the same as Eq. 3 in §2.4. It was reproduced here for convenience.

Forward language model	Backward language model	LM perplexity		$F_1 \pm$ std
		Fwd	Bwd	
—	—	N/A	N/A	90.87 ± 0.13
LSTM-512-256*	LSTM-512-256*	106.9	104.2	90.79 ± 0.15
LSTM-2048-512	—	47.7	N/A	91.40 ± 0.18
LSTM-2048-512	LSTM-2048-512	47.7	47.3	91.62 ± 0.23
CNN-BIG-LSTM	—	30.0	N/A	91.66 ± 0.13
CNN-BIG-LSTM	LSTM-2048-512	30.0	47.3	$\mathbf{91.93 \pm 0.19}$

Table 6: Comparison of CoNLL-2003 test set F_1 for different language model combinations. All language models were trained and evaluated on the 1B Word Benchmark, except LSTM-512-256* which was trained and evaluated on the standard splits of the NER CoNLL 2003 dataset.

set, we decreased the model size to 512 hidden units with a 256 dimension projection and normalized tokens in the same manner as input to the sequence tagging model (lower-cased, with all digits replaced with 0). The test set perplexities for the forward and backward models (measured on the CoNLL 2003 test data) were 106.9 and 104.2, respectively. Including embeddings from these language models *decreased* performance slightly compared to the baseline system without any LM. This result supports the hypothesis that adding language models help because they learn composition functions (i.e., the RNN parameters in the language model) from much larger data compared to the composition functions in the baseline tagger, which are only learned from labeled data.

Importance of task specific RNN. To understand the importance of including a task specific sequence RNN we ran an experiment that removed the task specific sequence RNN and used only the LM embeddings with a dense layer and CRF to predict output tags. In this setup, performance was very low, 88.17 F_1, well below our baseline. This result confirms that the RNNs in the baseline tagger encode essential information which is not encoded in the LM embeddings. This is unsurprising since the RNNs in the baseline tagger are trained on labeled examples, unlike the RNN in the language model which is only trained on unlabeled examples. Note that the LM weights are fixed in this experiment.

Dataset size. *A priori*, we expect the addition of LM embeddings to be most beneficial in cases where the task specific annotated datasets are small. To test this hypothesis, we replicated the setup from Yang et al. (2017) that samples 1% of the CoNLL 2003 training set and compared the performance of TagLM to our baseline without LM. In this scenario, test F_1 increased 3.35% (from 67.66 to 71.01%) compared to an increase of 1.06% F_1 for a similar comparison with the full training dataset. The analogous increases in Yang et al. (2017) are 3.97% for cross-lingual transfer from CoNLL 2002 Spanish NER and 6.28% F_1 for transfer from PTB POS tags. However, they found only a 0.06% F_1 increase when using the full training data and transferring from both CoNLL 2000 chunks and PTB POS tags. Taken together, this suggests that for very small labeled training sets, transferring from other tasks yields a large improvement, but this improvement almost disappears when the training data is large. On the other hand, our approach is less dependent on the training set size and significantly improves performance even with larger training sets.

Number of parameters. Our TagLM formulation increases the number of parameters in the second RNN layer R_2 due to the increase in the input dimension $\mathbf{h}_1$ if all other hyperparameters are held constant. To confirm that this did not have a material impact on the results, we ran two additional experiments. In the first, we trained a system without a LM but increased the second RNN layer hidden dimension so that number of parameters was the same as in TagLM. In this case, performance *decreased* slightly (by 0.15% F_1) compared to the baseline model, indicating that solely increasing parameters does not improve performance. In the second experiment, we decreased the hidden dimension of the second RNN layer in TagLM to give it the same number of parameters as the baseline no LM model. In this case, test F_1 *increased* slightly to 92.00 ± 0.11 indicating that the additional parameters in TagLM are slightly hurting

performance.[4]

Does the LM transfer across domains? One artifact of our evaluation framework is that both the labeled data in the chunking and NER tasks and the unlabeled text in the 1 Billion Word Benchmark used to train the bidirectional LMs are derived from news articles. To test the sensitivity to the LM training domain, we also applied TagLM with a LM trained on news articles to the SemEval 2017 Shared Task 10, ScienceIE.[5] ScienceIE requires end-to-end joint entity and relationship extraction from scientific publications across three diverse fields (computer science, material sciences, and physics) and defines three broad entity types (Task, Material and Process). For this task, TagLM increased F_1 on the development set by 4.12% (from 49.93 to to 54.05%) for entity extraction over our baseline without LM embeddings and it was a major component in our winning submission to ScienceIE, Scenario 1 (Ammar et al., 2017). We conclude that LM embeddings can improve the performance of a sequence tagger even when the data comes from a different domain.

4 Related work

Unlabeled data. TagLM was inspired by the widespread use of pre-trained word embeddings in supervised sequence tagging models. Besides pre-trained word embeddings, our method is most closely related to Li and McCallum (2005). Instead of using a LM, Li and McCallum (2005) uses a probabilistic generative model to infer context-sensitive latent variables for each token, which are then used as extra features in a supervised CRF tagger (Lafferty et al., 2001). Other semi-supervised learning methods for structured prediction problems include co-training (Blum and Mitchell, 1998; Pierce and Cardie, 2001), expectation maximization (Nigam et al., 2000; Mohit and Hwa, 2005), structural learning (Ando and Zhang, 2005) and maximum discriminant functions (Suzuki et al., 2007; Suzuki and Isozaki, 2008). It is easy to combine TagLM with any of the above methods by including LM embeddings as additional features in the discriminative components of the model (except for expectation maximization). A detailed discussion of semi-supervised learning methods in NLP can be found in (Søgaard, 2013).

Melamud et al. (2016) learned a context encoder from unlabeled data with an objective function similar to a bi-directional LM and applied it to several NLP tasks closely related to the unlabeled objective function: sentence completion, lexical substitution and word sense disambiguation.

LM embeddings are related to a class of methods (e.g., Le and Mikolov, 2014; Kiros et al., 2015; Hill et al., 2016) for learning sentence and document encoders from unlabeled data, which can be used for text classification and textual entailment among other tasks. Dai and Le (2015) pre-trained LSTMs using language models and sequence autoencoders then fine tuned the weights for classification tasks. In contrast to our method that uses unlabeled data to learn token-in-context embeddings, all of these methods use unlabeled data to learn an encoder for an entire text sequence (sentence or document).

Neural language models. LMs have always been a critical component in statistical machine translation systems (Koehn, 2009). Recently, neural LMs (Bengio et al., 2003; Mikolov et al., 2010) have also been integrated in neural machine translation systems (e.g., Kalchbrenner and Blunsom, 2013; Devlin et al., 2014) to score candidate translations. In contrast, TagLM uses neural LMs to encode words in the input sequence.

Unlike forward LMs, bidirectional LMs have received little prior attention. Most similar to our formulation, Peris and Casacuberta (2015) used a bidirectional neural LM in a statistical machine translation system for instance selection. They tied the input token embeddings and softmax weights in the forward and backward directions, unlike our approach which uses two distinct models without any shared parameters. Frinken et al. (2012) also used a bidirectional n-gram LM for handwriting recognition.

Interpreting RNN states. Recently, there has been some interest in interpreting the activations of RNNs. Linzen et al. (2016) showed that single LSTM units can learn to predict singular-plural distinctions. Karpathy et al. (2015) visualized character level LSTM states and showed that individual cells capture long-range dependencies such as line lengths, quotes and brackets. Our work complements these studies by showing that LM states are useful for downstream tasks as a way

[4] A similar experiment for the Chunking task did not improve F_1 so this conclusion is task dependent.

[5] https://scienceie.github.io/

of interpreting what they learn.

Other sequence tagging models. Current state of the art results in sequence tagging problems are based on bidirectional RNN models. However, many other sequence tagging models have been proposed in the literature for this class of problems (e.g., Lafferty et al., 2001; Collins, 2002). LM embeddings could also be used as additional features in other models, although it is not clear whether the model complexity would be sufficient to effectively make use of them.

5 Conclusion

In this paper, we proposed a simple and general semi-supervised method using pre-trained neural language models to augment token representations in sequence tagging models. Our method significantly outperforms current state of the art models in two popular datasets for NER and Chunking. Our analysis shows that adding a backward LM in addition to traditional forward LMs consistently improves performance. The proposed method is robust even when the LM is trained on unlabeled data from a different domain, or when the baseline model is trained on a large number of labeled examples.

Acknowledgments

We thank Chris Dyer, Julia Hockenmaier, Jayant Krishnamurthy, Matt Gardner and Oren Etzioni for comments on earlier drafts that led to substantial improvements in the final version.

References

Waleed Ammar, Matthew E. Peters, Chandra Bhagavatula, and Russell Power. 2017. The AI2 system at SemEval-2017 Task 10 (ScienceIE): semi-supervised end-to-end entity and relation extraction. In *ACL workshop (SemEval)*.

Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *ACL*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. In *JMLR*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2014. One billion word benchmark for measuring progress in statistical language modeling. *CoRR* abs/1312.3005.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. In *TACL*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *SSST@EMNLP*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. In *JMLR*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL*.

Volkmar Frinken, Alicia Fornés, Josep Lladós, and Jean-Marc Ogier. 2012. Bidirectional language model for handwriting recognition. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *CoRR* abs/1611.01587.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *HLT-NAACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR* abs/1602.02410.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. In *ICLR workshop*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jamie Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Wei Li and Andrew McCallum. 2005. Semi-supervised sequence modeling with syntactic topic models. In *AAAI*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. In *TACL*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *EMNLP*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *CoNLL*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Behrang Mohit and Rebecca Hwa. 2005. Syntax-based semi-supervised named entity tagging. In *ACL*.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine learning* .

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Álvaro Peris and Francisco Casacuberta. 2015. A bidirectional recurrent neural language model for machine translation. *Procesamiento del Lenguaje Natural* .

David Pierce and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In *EMNLP*.

Lev-Arie Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Hasim Sak, Andrew W. Senior, and Franoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *CoNLL/LLL*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Anders Søgaard. 2013. Semi-supervised learning and domain adaptation in natural language processing. *Synthesis Lectures on Human Language Technologies* .

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Jun Suzuki, Akinori Fujino, and Hideki Isozaki. 2007. Semi-supervised structured output learning based on a hybrid generative and discriminative approach. In *EMNLP-CoNLL*.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. In *ACL*.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*.

Learning Symmetric Collaborative Dialogue Agents with Dynamic Knowledge Graph Embeddings

He He and **Anusha Balakrishnan** and **Mihail Eric** and **Percy Liang**
Computer Science Department, Stanford University
{hehe,anusha28,meric,pliang}@cs.stanford.edu

Abstract

We study a *symmetric collaborative dialogue* setting in which two agents, each with private knowledge, must strategically communicate to achieve a common goal. The open-ended dialogue state in this setting poses new challenges for existing dialogue systems. We collected a dataset of 11K human-human dialogues, which exhibits interesting lexical, semantic, and strategic elements. To model both structured knowledge and unstructured language, we propose a neural model with dynamic knowledge graph embeddings that evolve as the dialogue progresses. Automatic and human evaluations show that our model is both more effective at achieving the goal and more human-like than baseline neural and rule-based models.

1 Introduction

Current task-oriented dialogue systems (Young et al., 2013; Wen et al., 2017; Dhingra et al., 2017) require a pre-defined dialogue state (e.g., slots such as food type and price range for a restaurant searching task) and a fixed set of dialogue acts (e.g., request, inform). However, human conversation often requires richer dialogue states and more nuanced, pragmatic dialogue acts. Recent open-domain chat systems (Shang et al., 2015; Serban et al., 2015b; Sordoni et al., 2015; Li et al., 2016a; Lowe et al., 2017; Mei et al., 2017) learn a mapping directly from previous utterances to the next utterance. While these models capture open-ended aspects of dialogue, the lack of structured dialogue state prevents them from being directly applied to settings that require interfacing with structured knowledge.

In order to bridge the gap between the two types of systems, we focus on a *symmetric collaborative dialogue* setting, which is task-oriented but encourages open-ended dialogue acts. In our setting, two agents, each with a private list of items with attributes, must communicate to identify the unique shared item. Consider the dialogue in Figure 1, in which two people are trying to find their mutual friend. By asking "do you have anyone who went to columbia?", B is suggesting that she has some Columbia friends, and that they probably work at Google. Such conversational implicature is lost when interpreting the utterance as simply an information request. In addition, it is hard to define a structured state that captures the diverse semantics in many utterances (e.g., defining "most of", "might be"; see details in Table 1).

To model both structured and open-ended context, we propose the *Dynamic Knowledge Graph Network* (DynoNet), in which the dialogue state is modeled as a knowledge graph with an embedding

Friends of agent A:

Name	School	Major	Company
Jessica	Columbia	Computer Science	Google
Josh	Columbia	Linguistics	Google
...	...	...	...

A: Hi! Most of my friends work for Google
B: do you have anyone who went to columbia?
A: *Hello?*
A: I have Jessica a friend of mine
A: and Josh, both went to columbia
B: *or anyone working at apple?*
B: SELECT (Jessica, Columbia, Computer Science, Google)
A: SELECT (Jessica, Columbia, Computer Science, Google)

Figure 1: An example dialogue from the MutualFriends task in which two agents, A and B, each given a private list of a friends, try to identify their mutual friend. Our objective is to build an agent that can perform the task with a human. Crosstalk (Section 2.3) is *italicized*.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1766–1776
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1162

for each node (Section 3). Our model is similar to EntNet (Henaff et al., 2017) in that node/entity embeddings are updated recurrently given new utterances. The difference is that we structure entities as a knowledge graph; as the dialogue proceeds, new nodes are added and new context is propagated on the graph. An attention-based mechanism (Bahdanau et al., 2015) over the node embeddings drives generation of new utterances. Our model's use of knowledge graphs captures the grounding capability of classic task-oriented systems and the graph embedding provides the representational flexibility of neural models.

The naturalness of communication in the symmetric collaborative setting enables large-scale data collection: We were able to crowdsource around 11K human-human dialogues on Amazon Mechanical Turk (AMT) in less than 15 hours.[1] We show that the new dataset calls for more flexible representations beyond fully-structured states (Section 2.2).

In addition to conducting the third-party human evaluation adopted by most work (Liu et al., 2016; Li et al., 2016b,c), we also conduct partner evaluation (Wen et al., 2017) where AMT workers rate their conversational partners (other workers or our models) based on fluency, correctness, cooperation, and human-likeness. We compare DynoNet with baseline neural models and a strong rule-based system. The results show that DynoNet can perform the task with humans efficiently and naturally; it also captures some strategic aspects of human-human dialogues.

The contributions of this work are: (i) a new symmetric collaborative dialogue setting and a large dialogue corpus that pushes the boundaries of existing dialogue systems; (ii) DynoNet, which integrates semantically rich utterances with structured knowledge to represent open-ended dialogue states; (iii) multiple automatic metrics based on bot-bot chat and a comparison of third-party and partner evaluation.

2 Symmetric Collaborative Dialogue

We begin by introducing a collaborative task between two agents and describe the human-human dialogue collection process. We show that our data exhibits diverse, interesting language phenomena.

[1]The dataset is available publicly at https://stanfordnlp.github.io/cocoa/.

2.1 Task Definition

In the symmetric collaborative dialogue setting, there are two agents, A and B, each with a private knowledge base—KB_A and KB_B, respectively. Each knowledge base includes a list of *items*, where each item has a value for each *attribute*. For example, in the MutualFriends setting, Figure 1, items are friends and attributes are name, school, etc. There is a shared item that A and B both have; their goal is to converse with each other to determine the shared item and select it. Formally, an agent is a mapping from its private KB and the dialogue thus far (sequence of utterances) to the next utterance to generate or a selection. A dialogue is considered *successful* when both agents correctly select the shared item. This setting has parallels in human-computer collaboration where each agent has complementary expertise.

2.2 Data collection

We created a schema with 7 attributes and approximately 3K entities (attribute values). To elicit linguistic and strategic variants, we generate a random scenario for each task by varying the number of items (5 to 12), the number attributes (3 or 4), and the distribution of values for each attribute (skewed to uniform). See Appendix A and B for details of schema and scenario generation.

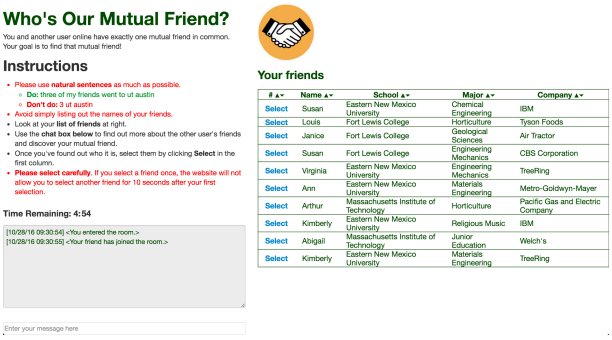


Figure 2: Screenshot of the chat interface.

We crowdsourced dialogues on AMT by randomly pairing up workers to perform the task within 5 minutes.[2] Our chat interface is shown in Figure 2. To discourage random guessing, we prevent workers from selecting more than once every 10 seconds. Our task was very popular and we col-

[2]If the workers exceed the time limit, the dialogue is marked as unsuccessful (but still logged).

Type	%	Easy example	Hard example
Inform	30.4	I know a judy. / I have someone who studied the bible in the afternoon.	**About equal** indoor and outdoor friends / **me too**. his major is forestry / **might be** kelly
Ask	17.7	Do any of them like Poi? / What does your henry do?	What can you tell me about our friend? / **Or maybe** north park college?
Answer	7.4	None of mine did / Yup / They do. / Same here.	yes 3 of them / No he likes poi / yes if boston college

Table 1: Main utterance types and examples. We show both standard utterances whose meaning can be represented by simple logical forms (e.g., ask(`indoor`)), and open-ended ones which require more complex logical forms (difficult parts in bold). Text spans corresponding to entities are underlined.

Phenomenon	Example
Coreference	(I know one Debra) does she like the indoors? / (I have two friends named TIffany) at World airways?
Coordination	keep on going with the fashion / Ok. let's try something else. / go by hobby / great. select him. thanks!
Chit-chat	Yes, that is **good ole** Terry. / All indoorsers! **my friends hate nature**
Categorization	same, most of mine are **female too / Does any of them names start with B**
Correction	I know one friend into Embroidery - her name is Emily. **Sorry – Embroidery friend is named Michelle**

Table 2: Communication phenomena in the dataset. Evident parts is in bold and text spans corresponding to an entity are underlined. For coreference, the antecedent is in parentheses.

lected 11K dialogues over a period of 13.5 hours.[3] Of these, over 9K dialogues are successful. Unsuccessful dialogues are usually the result of either worker leaving the chat prematurely.

2.3 Dataset statistics

We show the basic statistics of our dataset in Table 3. An utterance is defined as a message sent by one of the agents. The average utterance length is short due to the informality of the chat, however, an agent usually sends multiple utterances in one turn. Some example dialogues are shown in Table 6 and Appendix I.

# dialogues	11157
# completed dialogues	9041
Vocabulary size	5325
Average # of utterances	11.41
Average time taken per task (sec.)	91.18
Average utterance length (tokens)	5.08
Number of linguistic templates[4]	41561

Table 3: Statistics of the MutualFriends dataset.

We categorize utterances into coarse types—inform, ask, answer, greeting, apology—by pattern matching (Appendix E). There are 7.4% multi-type utterances, and 30.9% utterances contain more than one entity. In Table 1, we show example utterances with rich semantics that cannot be sufficiently represented by traditional slot-values. Some of the standard ones are also non-trivial due to coreference and logical compositionality.

Our dataset also exhibits some interesting communication phenomena. Coreference occurs frequently when people check multiple attributes of one item. Sometimes mentions are dropped, as an utterance simply continues from the partner's utterance. People occasionally use external knowledge to group items with out-of-schema attributes (e.g., gender based on names, location based on schools). We summarize these phenomena in Table 2. In addition, we find 30% utterances involve cross-talk where the conversation does not progress linearly (e.g., italic utterances in Figure 1), a common characteristic of online chat (Ivanovic, 2005).

One strategic aspect of this task is choosing the order of attributes to mention. We find that people tend to start from attributes with fewer unique values, e.g., "all my friends like morning" given the KB_B in Table 6, as intuitively it would help exclude items quickly given fewer values to check.[5] We provide a more detailed analysis of strategy in Section 4.2 and Appendix F.

3 Dynamic Knowledge Graph Network

The diverse semantics in our data motivates us to combine unstructured representation of the dialogue history with structured knowledge. Our

[3] Tasks are put up in batches; the total time excludes intervals between batches.

[4] Entity names are replaced by their entity types.

[5] Our goal is to model human behavior thus we do not discuss the optimal strategy here.

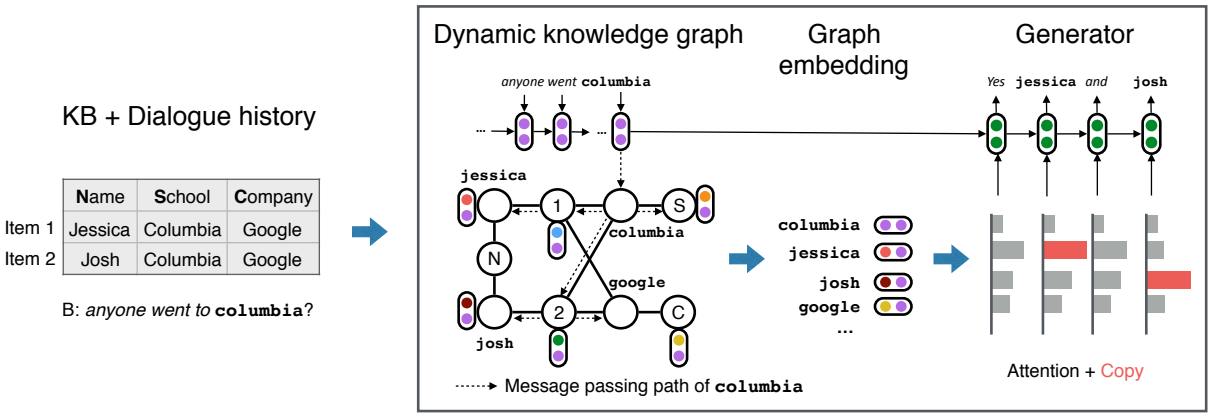


Figure 3: Overview of our approach. First, the KB and dialogue history (entities in **bold**) is mapped to a graph. Here, an item node is labeled by the item ID and an attribute node is labeled by the attribute's first letter. Next, each node is embedded using relevant utterance embeddings through message passing. Finally, an LSTM generates the next utterance based on attention over the node embeddings.

model consists of three components shown in Figure 3: (i) a dynamic knowledge graph, which represents the agent's private KB and shared dialogue history as a graph (Section 3.1), (ii) a graph embedding over the nodes (Section 3.2), and (iii) an utterance generator (Section 3.3).

The knowledge graph represents entities and relations in the agent's private KB, e.g., `item-1`'s `company` is `google`. As the conversation unfolds, utterances are embedded and incorporated into node embeddings of mentioned entities. For instance, in Figure 3, "anyone went to columbia" updates the embedding of `columbia`. Next, each node recursively passes its embedding to neighboring nodes so that related entities (e.g., those in the same row or column) also receive information from the most recent utterance. In our example, `jessica` and `josh` both receive new context when `columbia` is mentioned. Finally, the utterance generator, an LSTM, produces the next utterance by attending to the node embeddings.

3.1 Knowledge Graph

Given a dialogue of T utterances, we construct graphs $(G_t)_{t=1}^{T}$ over the KB and dialogue history for agent A.[6] There are three types of nodes: item nodes, attribute nodes, and entity nodes. Edges between nodes represent their relations. For example, `(item-1, hasSchool, columbia)` means that the first item has attribute `school` whose value is `columbia`. An example graph is shown in Figure 3. The graph G_t is updated based on utterance t by taking G_{t-1} and adding a new node for any entity mentioned in utterance t but not in KB_A.[7]

3.2 Graph Embedding

Given a knowledge graph, we are interested in computing a vector representation for each node v that captures both its unstructured context from the dialogue history and its structured context in the KB. A node embedding $V_t(v)$ for each node $v \in G_t$ is built from three parts: structural properties of an entity defined by the KB, embeddings of utterances in the dialogue history, and message passing between neighboring nodes.

Node Features. Simple structural properties of the KB often govern what is talked about; e.g., a high-frequency entity is usually interesting to mention (consider "All my friends like dancing."). We represent this type of information as a feature vector $F_t(v)$, which includes the degree and type (item, attribute, or entity type) of node v, and whether it has been mentioned in the current turn. Each feature is encoded as a one-hot vector and they are concatenated to form $F_t(v)$.

Mention Vectors. A mention vector $M_t(v)$ contains unstructured context from utterances relevant to node v up to turn t. To compute it, we first define the utterance representation $\tilde{u}_t$ and the set of relevant entities E_t. Let u_t be the embedding of utterance t (Section 3.3). To differentiate between

[6] It is important to differentiate perspectives of the two agents as they have different KBs. Thereafter we assume the perspective of agent A, i.e., accessing KB_A for A only, and refer to B as the partner.

[7] We use a rule-based lexicon to link text spans to entities. See details in Appendix D.

the agent's and the partner's utterances, we represent it as $\tilde{u}_t = \left[u_t \cdot \mathbb{1}_{\{u_t \in U_{\text{self}}\}}, u_t \cdot \mathbb{1}_{\{u_t \in U_{\text{partner}}\}}\right]$, where U_{self} and U_{partner} denote sets of utterances generated by the agent and the partner, and $[\cdot, \cdot]$ denotes concatenation. Let E_t be the set of entity nodes mentioned in utterance t if utterance t mentions some entities, or utterance $t-1$ otherwise.[8] The mention vector $M_t(v)$ of node v incorporates the current utterance if v is mentioned and inherits $M_{t-1}(v)$ if not:

$$M_t(v) = \lambda_t M_{t-1}(v) + (1-\lambda_t)\tilde{u}_t; \quad (1)$$

$$\lambda_t = \begin{cases} \sigma\left(W^{\text{inc}}\left[M_{t-1}(v), \tilde{u}_t\right]\right) & \text{if } v \in E_t, \\ 1 & \text{otherwise.} \end{cases}$$

Here, σ is the sigmoid function and W^{inc} is a parameter matrix.

Recursive Node Embeddings. We propagate information between nodes according to the structure of the knowledge graph. In Figure 3, given "anyone went to columbia?", the agent should focus on her friends who went to Columbia University. Therefore, we want this utterance to be sent to item nodes connected to `columbia`, and one step further to other attributes of these items because they might be mentioned next as relevant information, e.g., `jessica` and `josh`.

We compute the node embeddings recursively, analogous to belief propagation:

$$V_t^k(v) = \max_{v' \in N_t(v)} \tanh \left(W^{\text{mp}}\left[V_t^{k-1}(v'), R(e_{v \to v'})\right]\right), \quad (2)$$

where $V_t^k(v)$ is the depth-k node embedding at turn t and $N_t(v)$ denotes the set of nodes adjacent to v. The message from a neighboring node v' depends on its embedding at depth-$(k-1)$, the edge label $e_{v \to v'}$ (embedded by a relation embedding function R), and a parameter matrix W^{mp}. Messages from all neighbors are aggregated by $\max$, the element-wise max operation.[9] Example message passing paths are shown in Figure 3.

The final node embedding is the concatenation of embeddings at each depth:

$$V_t(v) = \left[V_t^0(v), \ldots, V_t^K(v)\right], \quad (3)$$

where K is a hyperparameter (we experiment with $K \in \{0, 1, 2\}$) and $V_t^0(v) = [F_t(v), M_t(v)]$.

[8] Relying on utterance $t-1$ is useful when utterance t answers a question, e.g., "do you have any google friends?" "No."

[9] Using sum or mean slightly hurts performance.

3.3 Utterance Embedding and Generation

We embed and generate utterances using Long Short Term Memory (LSTM) networks that take the graph embeddings into account.

Embedding. On turn t, upon receiving an utterance consisting of n_t tokens, $x_t = (x_{t,1}, \ldots, x_{t,n_t})$, the LSTM maps it to a vector as follows:

$$h_{t,j} = \text{LSTM}_{\text{enc}}(h_{t,j-1}, A_t(x_{t,j})), \quad (4)$$

where $h_{t,0} = h_{t-1,n_{t-1}}$, and A_t is an *entity abstraction* function, explained below. The final hidden state h_{t,n_t} is used as the utterance embedding u_t, which updates the mention vectors as described in Section 3.2.

In our dialogue task, the identity of an entity is unimportant. For example, replacing `google` with `alphabet` in Figure 1 should make little difference to the conversation. The role of an entity is determined instead by its relation to other entities and relevant utterances. Therefore, we define the abstraction $A_t(y)$ for a word y as follows: if y is linked to an entity v, then we represent an entity by its type (`school`, `company` etc.) embedding concatenated with its current node embedding: $A_t(y) = [E_{\text{type}(y)}, V_t(v)]$. Note that $V_t(v)$ is determined only by its structural features and its context. If y is a non-entity, then $A_t(y)$ is the word embedding of y concatenated with a zero vector of the same dimensionality as $V_t(v)$. This way, the representation of an entity only depends on its structural properties given by the KB and the dialogue context, which enables the model to generalize to unseen entities at test time.

Generation. Now, assuming we have embedded utterance x_{t-1} into $h_{t-1,n_{t-1}}$ as described above, we use another LSTM to generate utterance x_t. Formally, we carry over the last utterance embedding $h_{t,0} = h_{t-1,n_{t-1}}$ and define:

$$h_{t,j} = \text{LSTM}_{\text{dec}}(h_{t,j-1}, [A_t(x_{t,j}), c_{t,j}]), \quad (5)$$

where $c_{t,j}$ is a weighted sum of node embeddings in the current turn: $c_{t,j} = \sum_{v \in G_t} \alpha_{t,j,v} V_t(v)$, where $\alpha_{t,j,v}$ are the attention weights over the nodes. Intuitively, high weight should be given to relevant entity nodes as shown in Figure 3,. We compute the weights through standard attention mechanism (Bahdanau et al., 2015):

$$\alpha_{t,j} = \text{softmax}(s_{t,j}),$$

$$s_{t,j,v} = w^{\text{attn}} \cdot \tanh\left(W^{\text{attn}}\left[h_{t,j-1}, V_t(v)\right]\right),$$

where vector w^{attn} and W^{attn} are parameters.

Finally, we define a distribution over both words in the vocabulary and nodes in G_t using the copying mechanism of Jia and Liang (2016):

$$p(x_{t,j+1} = y \mid G_t, x_{t,\leq j}) \propto \exp\left(W^{\text{vocab}} h_{t,j} + b\right),$$
$$p(x_{t,j+1} = r(v) \mid G_t, x_{t,\leq j}) \propto \exp\left(s_{t,j,v}\right),$$

where y is a word in the vocabulary, W^{vocab} and b are parameters, and $r(v)$ is the realization of the entity represented by node v, e.g., `google` is realized to "Google" during copying.[10]

4 Experiments

We compare our model with a rule-based system and a baseline neural model. Both automatic and human evaluations are conducted to test the models in terms of fluency, correctness, cooperation, and human-likeness. The results show that DynoNet is able to converse with humans in a coherent and strategic way.

4.1 Setup

We randomly split the data into train, dev, and test sets (8:1:1). We use a one-layer LSTM with 100 hidden units and 100-dimensional word vectors for both the encoder and the decoder (Section 3.3). Each successful dialogue is turned into two examples, each from the perspective of one of the two agents. We maximize the log-likelihood of all utterances in the dialogues. The parameters are optimized by AdaGrad (Duchi et al., 2010) with an initial learning rate of 0.5. We trained for at least 10 epochs; after that, training stops if there is no improvement on the dev set for 5 epochs. By default, we perform $K = 2$ iterations of message passing to compute node embeddings (Section 3.2). For decoding, we sequentially sample from the output distribution with a softmax temperature of 0.5.[11] Hyperparameters are tuned on the dev set.

We compare DynoNet with its static cousion (StanoNet) and a rule-based system (Rule). StanoNet uses G_0 throughout the dialogue, thus the dialogue history is completely contained in the LSTM states instead of being injected into the knowledge graph. Rule maintains weights for each entity and each item in the KB to decide what to talk about and which item to select. It has a pattern-matching semantic parser, a rule-based policy, and a templated generator. See Appendix G for details.

4.2 Evaluation

We test our systems in two interactive settings: bot-bot chat and bot-human chat. We perform both automatic evaluation and human evaluation.

Automatic Evaluation. First, we compute the cross-entropy (ℓ) of a model on test data. As shown in Table 4, DynoNet has the lowest test loss. Next, we have a model chat with itself on the scenarios from the test set.[12] We evaluate the chats with respect to language variation, effectiveness, and strategy.

For language variation, we report the average utterance length L_u and the unigram entropy H in Table 4. Compared to Rule, the neural models tend to generate shorter utterances (Li et al., 2016b; Serban et al., 2017b). However, they are more diverse; for example, questions are asked in multiple ways such as "Do you have ...", "Any friends like ...", "What about ...".

At the discourse level, we expect the distribution of a bot's utterance types to match the distribution of human's. We show percentages of each utterance type in Table 4. For Rule, the decision about which action to take is written in the rules, while StanoNet and DynoNet learned to behave in a more human-like way, frequently informing and asking questions.

To measure effectiveness, we compute the overall success rate (C) and the success rate per turn (C_T) and per selection (C_S). As shown in Table 4, humans are the best at this game, followed by Rule which is comparable to DynoNet.

Next, we investigate the strategies leading to these results. An agent needs to decide which entity/attribute to check first to quickly reduce the search space. We hypothesize that humans tend to first focus on a majority entity and an attribute with fewer unique values (Section 2.3). For example, in the scenario in Table 6, `time` and `location` are likely to be mentioned first. We show the average frequency of first-mentioned entities (#Ent$_1$) and the average number of unique values for first-mentioned attributes ($|\text{Attr}_1|$) in Ta-

[10] We realize an entity by sampling from the empirical distribution of its surface forms found in the training data.

[11] Since selection is a common 'utterance' in our dataset and neural generation models are susceptible to over-generating common sentences, we halve its probability during sampling.

[12] We limit the number of turns in bot-bot chat to be the maximum number of turns humans took in the test set (46 turns).

System	$\ell \downarrow$	L_u	H	$C \uparrow$	$C_T \uparrow$	$C_S \uparrow$	Sel	Inf	Ask	Ans	Greet	#Ent$_1$	|Attr$_1$|	#Ent	#Attr
Human	-	5.10	4.57	.82	.07	.38	.21	.31	.17	.08	.08	.55	.35	6.1	2.6
Rule	-	7.61	3.37	.90	.05	**.29**	.18	**.34**	.23	.00	.12	.24	.61	9.9	3.0
StanoNet	2.20	4.01	**4.05**	.78	.04	.18	.19	.26	.12	.23	**.09**	.61	**.19**	7.1	2.9
DynoNet	**2.13**	3.37	3.90	**.96**	**.06**	.25	**.22**	.26	**.13**	.20	.12	**.55**	.18	**5.2**	**2.5**

Table 4: Automatic evaluation on human-human and bot-bot chats on test scenarios. We use ↑ / ↓ to indicate that higher / lower values are better; otherwise the objective is to match humans' statistics. Best results (except Human) are in bold. Neural models generate shorter (lower L_u) but more diverse (higher H) utterances. Overall, their distributions of utterance types match those of the humans'. (We only show the most frequent speech acts therefore the numbers do not sum to 1.) Rule is effective in completing the task (higher C_S), but it is not information-efficient given the large number of attributes (#Attr) and entities (#Ent) mentioned.

ble 4.[13] Both DynoNet and StanoNet successfully match human's starting strategy by favoring entities of higher frequency and attributes of smaller domain size.

To examine the overall strategy, we show the average number of attributes (#Attr) and entities (#Ent) mentioned during the conversation in Table 4. Humans and DynoNet strategically focus on a few attributes and entities, whereas Rule needs almost twice entities to achieve similar success rates. This suggests that the effectiveness of Rule mainly comes from large amounts of unselective information, which is consistent with comments from their human partners.

Partner Evaluation. We generated 200 new scenarios and put up the bots on AMT using the same chat interface that was used for data collection. The bots follow simple turn-taking rules explained in Appendix H. Each AMT worker is randomly paired with Rule, StanoNet, DynoNet, or another human (but the worker doesn't know which), and we make sure that all four types of agents are tested in each scenario at least once. At the end of each dialogue, humans are asked to rate their partner in terms of fluency, correctness, cooperation, and human-likeness from 1 (very bad) to 5 (very good), along with optional comments.

We show the average ratings (with significance tests) in Table 5 and the histograms in Appendix J. In terms of fluency, the models have similar performance since the utterances are usually short. Judgment on correctness is a mere guess since the evaluator cannot see the partner's KB; we will analyze correctness more meaningfully in the third-party evaluation below.

[13] Both numbers are normalized to $[0, 1]$ with respect to all entities/attributes in the corresponding KB.

Noticeably, DynoNet is more cooperative than the other models. As shown in the example dialogues in Table 6, DynoNet cooperates smoothly with the human partner, e.g., replying with relevant information about morning/indoor friends when the partner mentioned that all her friends prefer morning and most like indoor. StanoNet starts well but doesn't follow up on the morning friend, presumably because the `morning` node is not updated dynamically when mentioned by the partner. Rule follows the partner poorly. In the comments, the biggest complaint about Rule was that it was not 'listening' or 'understanding'. Overall, DynoNet achieves better partner satisfaction, especially in cooperation.

Third-party Evaluation. We also created a *third-party evaluation* task, where an independent AMT worker is shown a conversation and the KB of one of the agents; she is asked to rate the same aspects of the agent as in the partner evaluation and provide justifications. Each agent in a dialogue is rated by at least 5 people.

The average ratings and histograms are shown in Table 5 and Appendix J. For correctness, we see that Rule has the best performance since it always tells the truth, whereas humans can make mistakes due to carelessness and the neural models can generate false information. For example, in Table 6, DynoNet 'lied' when saying that it has a morning friend who likes outdoor.

Surprisingly, there is a discrepancy between the two evaluation modes in terms of cooperation and human-likeness. Manual analysis of the comments indicates that third-party evaluators focus less on the dialogue strategy and more on linguistic features, probably because they were not fully engaged in the dialogue. For example, justification

System	C	C_T	C_S	Partner eval				Third-party eval			
				Flnt	Crct	Coop	Human	Flnt	Crct	Coop	Human
Human	.89	.07	.36	4.2^{rds}	4.3^{rds}	4.2^{rds}	4.1^{rds}	4.0	4.3^{ds}	4.0^{ds}	4.1^{rds}
Rule	**.88**	**.06**	**.29**	3.6	4.0	3.5	3.5	4.0	$\mathbf{4.4}^{hds}$	$\mathbf{3.9}^{s}$	$\mathbf{4.0}^{s}$
StanoNet	.76	.04	.23	3.5	3.8	3.4	3.3	4.0	4.0	3.8	3.8
DynoNet	.87	.05	.27	$\mathbf{3.8}^{s}$	4.0	$\mathbf{3.8}^{rs}$	$\mathbf{3.6}^{s}$	4.0	4.1	3.9	3.9

Table 5: Results on human-bot/human chats. Best results (except Human) in each column are in bold. We report the average ratings of each system. For third-party evaluation, we first take mean of each question then average the ratings. DynoNet has the best partner satisfaction in terms of fluency (Flnt), correctness (Crct), cooperation (Coop), human likeness (Human). The superscript of a result indicates that its advantage over other systems (r: Rule, s: StanoNet, d: DynoNet) is statistically significant with $p < 0.05$ given by paired t-tests.

for cooperation often mentions frequent questions and timely answers, less attention is paid to what is asked about though.

For human-likeness, partner evaluation is largely correlated with coherence (e.g., not repeating or ignoring past information) and task success, whereas third-party evaluators often rely on informality (e.g., usage of colloquia like "hiya", capitalization, and abbreviation) or intuition. Interestingly, third-party evaluators noted most phenomena listed in Table 2 as indicators of human-beings, e.g., correcting oneself, making chit-chat other than simply finishing the task. See example comments in Appendix K.

4.3 Ablation Studies

Our model has two novel designs: entity abstraction and message passing for node embeddings. Table 7 shows what happens if we ablate these. When the number of message passing iterations, K, is reduced from 2 to 0, the loss consistently increases. Removing entity abstraction—meaning adding entity embeddings to node embeddings and the LSTM input embeddings—also degrades performance. This shows that DynoNet benefits from contextually-defined, structural node embeddings rather than ones based on a classic lookup table.

Model	ℓ
DynoNet (K = 2)	**2.16**
DynoNet (K = 1)	2.20
DynoNet (K = 0)	2.26
DynoNet (K = 2) w/o entity abstraction	2.21

Table 7: Ablations of our model on the dev set show the importance of entity abstraction and message passing ($K = 2$).

5 Discussion and Related Work

There has been a recent surge of interest in end-to-end task-oriented dialogue systems, though progress has been limited by the size of available datasets (Serban et al., 2015a). Most work focuses on information-querying tasks, using Wizard-of-Oz data collection (Williams et al., 2016; Asri et al., 2016) or simulators (Bordes and Weston, 2017; Li et al., 2016d), In contrast, collaborative dialogues are easy to collect as natural human conversations, and are also challenging enough given the large number of scenarios and diverse conversation phenomena. There are some interesting strategic dialogue datasets—settlers of Catan (Afantenos et al., 2012) (2K turns) and the cards corpus (Potts, 2012) (1.3K dialogues), as well as work on dialogue strategies (Keizer et al., 2017; Vogel et al., 2013), though no full dialogue system has been built for these datasets.

Most task-oriented dialogue systems follow the POMDP-based approach (Williams and Young, 2007; Young et al., 2013). Despite their success (Wen et al., 2017; Dhingra et al., 2017; Su et al., 2016), the requirement for handcrafted slots limits their scalability to new domains and burdens data collection with extra state labeling. To go past this limit, Bordes and Weston (2017) proposed a Memory-Networks-based approach without domain-specific features. However, the memory is unstructured and interfacing with KBs relies on API calls, whereas our model embeds both the dialogue history and the KB structurally. Williams et al. (2017) use an LSTM to automatically infer the dialogue state, but as they focus on dialogue control rather than the full problem, the response is modeled as a templated action, which restricts the generation of richer utterances. Our network ar-

Friends of A

ID	Name	Company	Time	Location
1	Kathy	TRT Holdings	afternoon	indoor
2	Jason	Dollar General	afternoon	indoor
3	Johnny	TRT Holdings	afternoon	outdoor
4	Frank	SFN Group	afternoon	indoor
5	Catherine	Dollar General	afternoon	indoor
6	Catherine	Weis Markets	afternoon	indoor
7	Kathleen	TRT Holdings	morning	indoor
8	Lori	TRT Holdings	afternoon	indoor
9	Frank	L&L Hawaiian Barbecue	afternoon	outdoor

Friends of B

ID	Name	Company	Time	Location
1	Justin	New Era Tickets	morning	indoor
2	Kathleen	TRT Holdings	morning	indoor
3	Gloria	L&L Hawaiian Barbecue	morning	indoor
4	Kathleen	Advance Auto Parts	morning	outdoor
5	Justin	Arctic Cat	morning	indoor
6	Anna	Dollar General	morning	indoor
7	Steven	SFN Group	morning	indoor
8	Wayne	R.J. Corman Railroad Group	morning	indoor
9	Alexander	R.J. Corman Railroad Group	morning	indoor

A: Human B: Human

A: Hi
B: hey || i have one outdoor
A: I have 4 TRT Holdings || I have 2 outdoor one Johnny, other Frank
B: i only have one TRT Holdings - Kathleen
A: SELECT 7 B: SELECT 2

A: DynoNet B: Human

A: **hi**
B: Hello || all my friends prefer morning
A: **1 of my morning likes the outdoors**
B: and all like indoor except for one
A: **do they work for trt holdings?**
B: Kathleen?
A: **SELECT 7** B: SELECT 2

A: StanoNet B: Human

A: **Hello**
B: hi
A: **Hello || I have one morning person.**
B: all of my friends like mornings
A: **My friend prefers afternoon works at trt holdings.**
B: what is their name?
A: **Likes indoors.**
B: what is your fiend who likes morning name?
A: **They work for trt holdings.**
B: SELECT 2 A: **SELECT 7**

A: Human B: Rule

B: **hiya**
A: hEY
B: **I have 1 indoors and kathleen.**
A: Most of mine are indoors.
B: **SELECT 1**
A: I have one morning and rest afternoon.
B: **Do you have any friend working at l hawaiian?**
A: I don't know Justin
B: **I have 1 alexander.**
...

Table 6: Examples of human-bot chats. The mutual friend is highlighted in blue in each KB. Bots' utterances are in bold and selected items are represented by item IDs. Only the first half of the human-Rule chat is shown due to limited space. Multiple utterances of one agent rae separated by ||.

chitecture is most similar to EntNet (Henaff et al., 2017), where memories are also updated by input sentences recurrently. The main difference is that our model allows information to be propagated between structured entities, which is shown to be crucial in our setting (Section 4.3).

Our work is also related to language generation conditioned on knowledge bases (Mei et al., 2016; Kiddon et al., 2016). One challenge here is to avoid generating false or contradicting statements, which is currently a weakness of neural models. Our model is mostly accurate when generating facts and answering existence questions about a single entity, but will need a more advanced attention mechanism for generating utterances involving multiple entities, e.g., attending to items or attributes first, then selecting entities; generating high-level concepts before composing them to natural tokens (Serban et al., 2017a).

In conclusion, we believe the symmetric collaborative dialogue setting and our dataset provide unique opportunities at the interface of traditional task-oriented dialogue and open-domain chat. We also offered DynoNet as a promising means for open-ended dialogue state representation. Our dataset facilitates the study of pragmatics and human strategies in dialogue—a good stepping stone towards learning more complex dialogues such as negotiation.

Acknowledgments. This work is supported by DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF-15-1-0462. Mike Kayser worked on an early version of the project while he was at Stanford. We also thank members of the Stanford NLP group for insightful discussions.

Reproducibility. All code, data, and experiments for this paper are available on the CodaLab platform: https://worksheets.codalab.org/worksheets/0xc757f29f5c794e5eb7bfa8ca9c945573.

References

S. Afantenos, N. Asher, F. Benamara, A. Cadilhac, C. Dégremont, P. Denis, M. Guhe, S. Keizer, A. Lascarides, O. Lemon, P. Muller, S. Paul, V. Rieser, and L. Vieu. 2012. Developing a corpus of strategic conversation in the settlers of catan. In *SeineDial 2012 - The 16th Workshop on the Semantics and Pragmatics of Dialogue*.

L. E. Asri, H. Schulz, S. Sharma, J. Zumer, J. Harris, E. Fine, R. Mehrotra, and K. Suleman. 2016. Frames: A corpus for adding memory to goal-oriented dialogue systems. *Maluuba Technical Report* .

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

A. Bordes and J. Weston. 2017. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations (ICLR)*.

B. Dhingra, L. Li, X. Li, J. Gao, Y. Chen, F. Ahmed, and L. Deng. 2017. End-to-end reinforcement learning of dialogue agents for information access. In *Association for Computational Linguistics (ACL)*.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

M. Henaff, J. Weston, A. Szlam, A. Bordes, and Y. LeCun. 2017. Tracking the world state with recurrent entity networks. In *International Conference on Learning Representations (ICLR)*.

E. Ivanovic. 2005. Dialogue act tagging for instant messaging chat sessions. In *Association for Computational Linguistics (ACL)*.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

S. Keizer, M. Guhe, H. Cuayahuitl, I. Efstathiou, K. Engelbrecht, M. Dobre, A. Lascarides, and O. Lemon. 2017. Evaluating persuasion strategies and deep reinforcement learning methods for negotiation dialogue agents. In *European Association for Computational Linguistics (EACL)*.

C. Kiddon, L. S. Zettlemoyer, and Y. Choi. 2016. Globally coherent text generation with neural checklist models. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. 2016a. A persona-based neural conversation model. In *Association for Computational Linguistics (ACL)*.

J. Li, M. Galley, C. Brockett, J. Gao, and W. B. Dolan. 2016b. A diversity-promoting objective function for neural conversation models. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

X. Li, Z. C. Lipton, B. Dhingra, L. Li, J. Gao, and Y. Chen. 2016d. A user simulator for task-completion dialogues. *arXiv* .

C. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

R. T. Lowe, N. Pow, I. Serban, L. Charlin, C. Liu, and J. Pineau. 2017. Training End-to-End dialogue systems with the ubuntu dialogue corpus. *Dialogue and Discourse* 8.

H. Mei, M. Bansal, and M. R. Walter. 2016. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

H. Mei, M. Bansal, and M. R. Walter. 2017. Coherent dialogue with attention-based language models. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

C. Potts. 2012. Goal-driven answers in the Cards dialogue corpus. In *Proceedings of the 30th West Coast Conference on Formal Linguistics*.

I. Serban, T. Klinger, G. Tesauro, K. Talamadupula, B. Zhou, Y. Bengio, and A. C. Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

I. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. C. Courville, and Y. Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

I. V. Serban, R. Lowe, L. Charlin, and J. Pineau. 2015a. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742* .

I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. 2015b. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808* .

L. Shang, Z. Lu, and H. Li. 2015. Neural responding machine for short-text conversation. In *Association for Computational Linguistics (ACL)*.

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J. Nie, J. Gao, and B. Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *North American Association for Computational Linguistics (NAACL)*.

P. Su, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, S. Ultes, D. Vandyke, T. Wen, and S. J. Young. 2016. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689* .

A. Vogel, M. Bodoia, C. Potts, and D. Jurafsky. 2013. Emergence of gricean maxims from multi-agent decision theory. In *North American Association for Computational Linguistics (NAACL)*. pages 1072–1081.

T. Wen, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, P. Su, S. Ultes, D. Vandyke, and S. Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *European Association for Computational Linguistics (EACL)*.

J. D. Williams, K. Asadi, and G. Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Association for Computational Linguistics (ACL)*.

J. D. Williams, A. Raux, and M. Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue and Discourse* 7.

J. D. Williams and S. Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language* 21(2):393–422.

S. Young, M. Gasic, B. Thomson, and J. D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.

Neural Belief Tracker: Data-Driven Dialogue State Tracking

Nikola Mrkšić[1], **Diarmuid Ó Séaghdha**[2]
Tsung-Hsien Wen[1], **Blaise Thomson**[2], **Steve Young**[1]
[1] University of Cambridge
[2] Apple Inc.
{nm480, thw28, sjy}@cam.ac.uk
{doseaghdha, blaisethom}@apple.com

Abstract

One of the core components of modern spoken dialogue systems is the *belief tracker*, which estimates the user's goal at every step of the dialogue. However, most current approaches have difficulty scaling to larger, more complex dialogue domains. This is due to their dependency on either: **a)** Spoken Language Understanding models that require large amounts of annotated training data; or **b)** hand-crafted lexicons for capturing some of the linguistic variation in users' language. We propose a novel Neural Belief Tracking (NBT) framework which overcomes these problems by building on recent advances in representation learning. NBT models reason over pre-trained word vectors, learning to compose them into distributed representations of user utterances and dialogue context. Our evaluation on two datasets shows that this approach surpasses past limitations, matching the performance of state-of-the-art models which rely on hand-crafted semantic lexicons and outperforming them when such lexicons are not provided.

1 Introduction

Spoken dialogue systems (SDS) allow users to interact with computer applications through conversation. Task-based systems help users achieve goals such as finding restaurants or booking flights. The *dialogue state tracking* (DST) component of an SDS serves to interpret user input and update the *belief state*, which is the system's internal representation of the state of the conversation (Young et al., 2010). This is a probability distribution over dialogue states used by the downstream *dialogue manager* to decide which action the system should perform next (Su et al., 2016a,b); the system action is then verbalised by the natural language generator (Wen et al., 2015a,b; Dušek and Jurčíček, 2015).

User: I'm looking for a cheaper restaurant
`inform(price=cheap)`
System: Sure. What kind - and where?
User: Thai food, somewhere downtown
`inform(price=cheap, food=Thai, area=centre)`
System: The House serves cheap Thai food
User: Where is it?
`inform(price=cheap, food=Thai, area=centre); request(address)`
System: The House is at 106 Regent Street

Figure 1: Annotated dialogue states in a sample dialogue. Underlined words show rephrasings which are typically handled using semantic dictionaries.

The Dialogue State Tracking Challenge (DSTC) series of shared tasks has provided a common evaluation framework accompanied by labelled datasets (Williams et al., 2016). In this framework, the dialogue system is supported by a *domain ontology* which describes the range of user intents the system can process. The ontology defines a collection of *slots* and the *values* that each slot can take. The system must track the search constraints expressed by users (*goals* or *informable* slots) and questions the users ask about search results (*requests*), taking into account each user utterance (input via a speech recogniser) and the dialogue context (e.g., what the system just said). The example in Figure 1 shows the true state after each user utterance in a three-turn conversation. As can be seen in this example, DST models depend on identifying mentions of ontology items in user utterances. This becomes a non-trivial task when confronted with lexical variation, the dynamics of context and noisy automated speech recognition (ASR) output.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1777–1788
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1163

FOOD=CHEAP: [affordable, budget, low-cost, low-priced, inexpensive, cheaper, economic, ...]

RATING=HIGH: [best, high-rated, highly rated, top-rated, cool, chic, popular, trendy, ...]

AREA=CENTRE: [center, downtown, central, city centre, midtown, town centre, ...]

Figure 2: An example semantic dictionary with rephrasings for three ontology values in a *restaurant search* domain.

Traditional statistical approaches use separate Spoken Language Understanding (SLU) modules to address lexical variability within a single dialogue turn. However, training such models requires substantial amounts of domain-specific annotation. Alternatively, turn-level SLU and cross-turn DST can be coalesced into a single model to achieve superior belief tracking performance, as shown by Henderson et al. (2014d). Such coupled models typically rely on manually constructed semantic dictionaries to identify alternative mentions of ontology items that vary lexically or morphologically. Figure 2 gives an example of such a dictionary for three slot-value pairs. This approach, which we term *delexicalisation*, is clearly not scalable to larger, more complex dialogue domains. Importantly, the focus on English in DST research understates the considerable challenges that morphology poses to systems based on exact matching in morphologically richer languages such as Italian or German (see Vulić et al. (2017)).

In this paper, we present two new models, collectively called the Neural Belief Tracker (NBT) family. The proposed models couple SLU and DST, efficiently learning to handle variation without requiring any hand-crafted resources. To do that, NBT models move away from exact matching and instead reason entirely over pre-trained word vectors. The vectors making up the user utterance and preceding system output are first composed into intermediate representations. These representations are then used to decide which of the ontology-defined intents have been expressed by the user up to that point in the conversation.

To the best of our knowledge, NBT models are the first to successfully use pre-trained word vector spaces to improve the language understanding capability of belief tracking models. In evaluation on two datasets, we show that: **a)** NBT models match the performance of delexicalisation-based models which make use of hand-crafted semantic lexicons; and **b)** the NBT models significantly outperform those models when such resources are not available. Consequently, we believe this work proposes a framework better-suited to scaling belief tracking models for deployment in real-world dialogue systems operating over sophisticated application domains where the creation of such domain-specific lexicons would be infeasible.

2 Background

Models for probabilistic dialogue state tracking, or *belief tracking*, were introduced as components of spoken dialogue systems in order to better handle noisy speech recognition and other sources of uncertainty in understanding a user's goals (Bohus and Rudnicky, 2006; Williams and Young, 2007; Young et al., 2010). Modern dialogue management policies can learn to use a tracker's distribution over intents to decide whether to execute an action or request clarification from the user. As mentioned above, the DSTC shared tasks have spurred research on this problem and established a standard evaluation paradigm (Williams et al., 2013; Henderson et al., 2014b,a). In this setting, the task is defined by an *ontology* that enumerates the goals a user can specify and the attributes of entities that the user can request information about. Many different belief tracking models have been proposed in the literature, from generative (Thomson and Young, 2010) and discriminative (Henderson et al., 2014d) statistical models to rule-based systems (Wang and Lemon, 2013). To motivate the work presented here, we categorise prior research according to their reliance (or otherwise) on a separate SLU module for interpreting user utterances:[1]

Separate SLU Traditional SDS pipelines use Spoken Language Understanding (SLU) decoders to detect slot-value pairs expressed in the Automatic Speech Recognition (ASR) output. The downstream DST model then combines this information with the past dialogue context to update the belief state (Thomson and Young, 2010; Wang and Lemon, 2013; Lee and Kim, 2016; Perez, 2016; Perez and Liu, 2017; Sun et al., 2016; Jang et al., 2016; Shi et al., 2016; Dernoncourt et al., 2016; Liu and Perez, 2017; Vodolán et al., 2017).

[1]The best-performing models in DSTC2 all used both raw ASR output and the output of (potentially more than one) SLU decoders (Williams, 2014; Williams et al., 2016). This does not mean that those models are immune to the drawbacks identified here for the two model categories; in fact, they share the drawbacks of both.

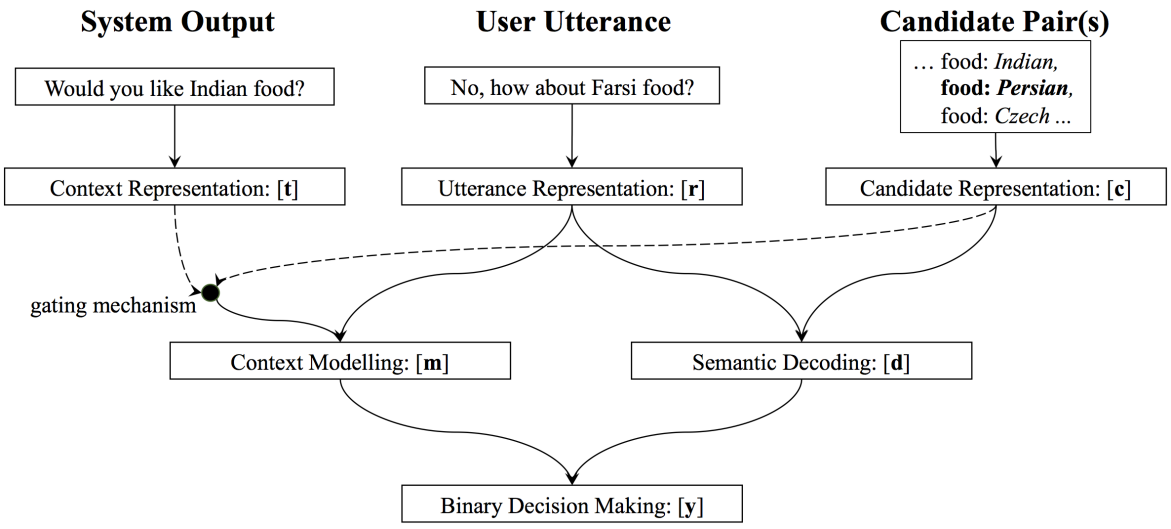


Figure 3: Architecture of the NBT Model. The implementation of the three representation learning subcomponents can be modified, as long as these produce adequate vector representations which the downstream model components can use to decide whether the current candidate slot-value pair was expressed in the user utterance (taking into account the preceding system act).

In the DSTC challenges, some systems used the output of template-based matching systems such as Phoenix (Wang, 1994). However, more robust and accurate statistical SLU systems are available. Many discriminative approaches to spoken dialogue SLU train independent binary models that decide whether each slot-value pair was expressed in the user utterance. Given enough data, these models can learn which lexical features are good indicators for a given value and can capture elements of paraphrasing (Mairesse et al., 2009). This line of work later shifted focus to robust handling of rich ASR output (Henderson et al., 2012; Tur et al., 2013). SLU has also been treated as a sequence labelling problem, where each word in an utterance is labelled according to its role in the user's intent; standard labelling models such as CRFs or Recurrent Neural Networks can then be used (Raymond and Ricardi, 2007; Yao et al., 2014; Celikyilmaz and Hakkani-Tur, 2015; Mesnil et al., 2015; Peng et al., 2015; Zhang and Wang, 2016; Liu and Lane, 2016b; Vu et al., 2016; Liu and Lane, 2016a, i.a.). Other approaches adopt a more complex modelling structure inspired by semantic parsing (Saleh et al., 2014; Vlachos and Clark, 2014). One drawback shared by these methods is their resource requirements, either because they need to learn independent parameters for each slot and value or because they need fine-grained manual annotation at the word level. This hinders scaling to larger, more realistic application domains.

Joint SLU/DST Research on belief tracking has found it advantageous to reason about SLU and DST jointly, taking ASR predictions as input and generating belief states as output (Henderson et al., 2014d; Sun et al., 2014; Zilka and Jurcicek, 2015; Mrkšić et al., 2015). In DSTC2, systems which used no external SLU module outperformed all systems that only used external SLU features. Joint models typically rely on a strategy known as *delexicalisation* whereby slots and values mentioned in the text are replaced with generic labels. Once the dataset is transformed in this manner, one can extract a collection of template-like n-gram features such as **[want *tagged-value* food]**. To perform belief tracking, the shared model iterates over all slot-value pairs, extracting delexicalised feature vectors and making a separate binary decision regarding each pair. Delexicalisation introduces a hidden dependency that is rarely discussed: how do we identify slot/value mentions in text? For toy domains, one can manually construct *semantic dictionaries* which list the potential rephrasings for all slot values. As shown by Mrkšić et al. (2016), the use of such dictionaries is essential for the performance of current delexicalisation-based models. Again though, this will not scale to the rich variety of user language or to general domains.

The primary motivation for the work presented in this paper is to overcome the limitations that affect previous belief tracking models. The NBT model efficiently learns from the avail-

able data by: **1)** leveraging semantic information from pre-trained word vectors to resolve lexical/morphological ambiguity; **2)** maximising the number of parameters shared across ontology values; and **3)** having the flexibility to learn domain-specific paraphrasings and other kinds of variation that make it infeasible to rely on exact matching and delexicalisation as a robust strategy.

3 Neural Belief Tracker

The Neural Belief Tracker (NBT) is a model designed to detect the slot-value pairs that make up the user's goal at a given turn during the flow of dialogue. Its input consists of the system dialogue acts preceding the user input, the user utterance itself, and a single candidate slot-value pair that it needs to make a decision about. For instance, the model might have to decide whether the goal FOOD=ITALIAN has been expressed in *'I'm looking for good pizza'*. To perform belief tracking, the NBT model *iterates* over all candidate slot-value pairs (defined by the ontology), and decides which ones have just been expressed by the user.

Figure 3 presents the flow of information in the model. The first layer in the NBT hierarchy performs representation learning given the three model inputs, producing vector representations for the user utterance ($\mathbf{r}$), the current candidate slot-value pair ($\mathbf{c}$) and the system dialogue acts ($\mathbf{t_q}, \mathbf{t_s}, \mathbf{t_v}$). Subsequently, the learned vector representations interact through the *context modelling* and *semantic decoding* submodules to obtain the intermediate *interaction summary* vectors $\mathbf{d_r}, \mathbf{d_c}$ and $\mathbf{d}$. These are used as input to the final *decision-making* module which decides whether the user expressed the intent represented by the candidate slot-value pair.

3.1 Representation Learning

For any given user utterance, system act(s) and candidate slot-value pair, the representation learning submodules produce vector representations which act as input for the downstream components of the model. All representation learning subcomponents make use of pre-trained collections of word vectors. As shown by Mrkšić et al. (2016), specialising word vectors to express *semantic similarity* rather than *relatedness* is essential for improving belief tracking performance. For this reason, we use the semantically-specialised Paragram-SL999 word vectors (Wieting et al., 2015) throughout this work. The NBT training procedure keeps these vectors fixed: that way, at test time, unseen words semantically related to familiar slot values (i.e. *inexpensive* to *cheap*) will be recognised purely by their position in the original vector space (see also Rocktäschel et al. (2016)). This means that the NBT model parameters can be shared across all values of the given slot, or even across all slots.

Let u represent a user utterance consisting of k_u words $u_1, u_2, \ldots, u_{k_u}$. Each word has an associated word vector $\mathbf{u}_1, \ldots, \mathbf{u}_{k_u}$. We propose two model variants which differ in the method used to produce vector representations of u: NBT-DNN and NBT-CNN. Both act over the constituent n-grams of the utterance. Let $\mathbf{v}_i^n$ be the concatenation of the n word vectors starting at index i, so that:

$$\mathbf{v}_i^n = \mathbf{u}_i \oplus \ldots \oplus \mathbf{u}_{i+n-1} \tag{1}$$

where $\oplus$ denotes vector concatenation. The simpler of our two models, which we term NBT-DNN, is shown in Figure 4. This model computes cumulative n-gram representation vectors $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$, which are the n-gram 'summaries' of the unigrams, bigrams and trigrams in the user utterance:

$$\mathbf{r}_n = \sum_{i=1}^{k_u - n + 1} \mathbf{v}_i^n \tag{2}$$

Each of these vectors is then non-linearly mapped to intermediate representations of the same size:

$$\mathbf{r}'_n = \sigma(W_n^s \mathbf{r}_n + b_n^s) \tag{3}$$

where the weight matrices and bias terms map the cumulative n-grams to vectors of the same dimensionality and σ denotes the sigmoid activation function. We maintain a separate set of parameters for each slot (indicated by superscript s). The three vectors are then summed to obtain a single representation for the user utterance:

$$\mathbf{r} = \mathbf{r}'_1 + \mathbf{r}'_2 + \mathbf{r}'_3 \tag{4}$$

The cumulative n-gram representations used by this model are just unweighted sums of all word vectors in the utterance. Ideally, the model should learn to recognise which parts of the utterance are more relevant for the subsequent classification task. For instance, it could learn to ignore verbs or stop words and pay more attention to adjectives and nouns which are more likely to express slot values.

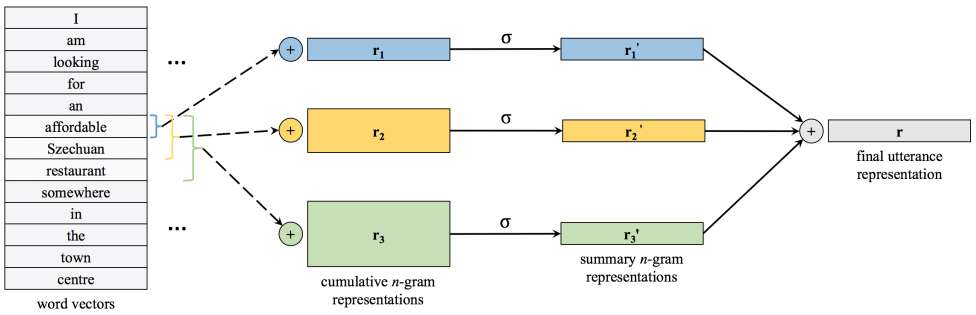


Figure 4: NBT-DNN MODEL. Word vectors of n-grams ($n = 1, 2, 3$) are summed to obtain *cumulative* n-grams, then passed through another hidden layer and summed to obtain the utterance representation $\mathbf{r}$.

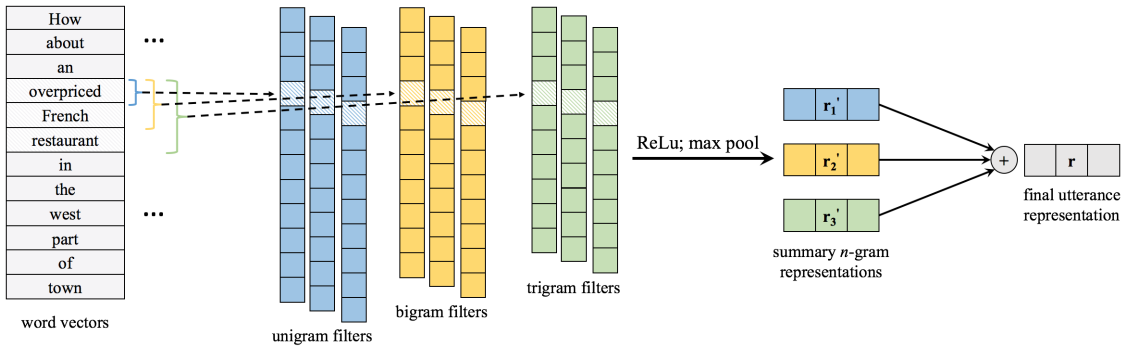


Figure 5: NBT-CNN Model. L convolutional filters of window sizes $1, 2, 3$ are applied to word vectors of the given utterance ($L = 3$ in the diagram, but $L = 300$ in the system). The convolutions are followed by the ReLU activation function and max-pooling to produce summary n-gram representations. These are summed to obtain the utterance representation $\mathbf{r}$.

NBT-CNN Our second model draws inspiration from successful applications of Convolutional Neural Networks (CNNs) for language understanding (Collobert et al., 2011; Kalchbrenner et al., 2014; Kim, 2014). These models typically apply a number of convolutional filters to n-grams in the input sentence, followed by non-linear activation functions and max-pooling. Following this approach, the NBT-CNN model applies $L = 300$ different filters for n-gram lengths of $1, 2$ and 3 (Figure 5). Let $F_n^s \in R^{L \times nD}$ denote the collection of filters for each value of n, where $D = 300$ is the word vector dimensionality. If $\mathbf{v}_i^n$ denotes the concatenation of n word vectors starting at index i, let $\mathbf{m}_n = [\mathbf{v}_1^n; \mathbf{v}_2^n; \ldots; \mathbf{v}_{k_u-n+1}^n]$ be the list of n-grams that convolutional filters of length n run over. The three intermediate representations are then given by:

$$R_n = F_n^s \, \mathbf{m}_n \tag{5}$$

Each column of the intermediate matrices R_n is produced by a single convolutional filter of length n. We obtain summary n-gram representations by pushing these representations through a rectified linear unit (ReLU) activation function (Nair and Hinton, 2010) and max-pooling over time (i.e. columns of the matrix) to get a single feature for each of the L filters applied to the utterance:

$$\mathbf{r}'_n = \texttt{maxpool}\left(\texttt{ReLU}\left(R_n + b_n^s\right)\right) \tag{6}$$

where b_n^s is a bias term broadcast across all filters. Finally, the three summary n-gram representations are summed to obtain the final utterance representation vector $\mathbf{r}$ (as in Equation 4). The NBT-CNN model is (by design) better suited to longer utterances, as its convolutional filters interact directly with subsequences of the utterance, and not just their noisy summaries given by the NBT-DNN's cumulative n-grams.

3.2 Semantic Decoding

The NBT diagram in Figure 3 shows that the utterance representation $\mathbf{r}$ and the candidate slot-value pair representation $\mathbf{c}$ directly interact through the *semantic decoding* module. This component decides whether the user explicitly expressed an intent matching the current candidate pair

(i.e. without taking the dialogue context into account). Examples of such matches would be *'I want Thai food'* with `food=Thai` or more demanding ones such as *'a pricey restaurant'* with `price=expensive`. This is where the use of high-quality pre-trained word vectors comes into play: a delexicalisation-based model could deal with the former example but would be helpless in the latter case, unless a human expert had provided a semantic dictionary listing all potential rephrasings for each value in the domain ontology.

Let the vector space representations of a candidate pair's slot name and value be given by $\mathbf{c_s}$ and $\mathbf{c_v}$ (with vectors of multi-word slot names/values summed together). The NBT model learns to map this tuple into a single vector $\mathbf{c}$ of the same dimensionality as the utterance representation $\mathbf{r}$. These two representations are then forced to interact in order to learn a similarity metric which discriminates between interactions of utterances with slot-value pairs that they either do or do not express:

$$\mathbf{c} = \sigma\big(W_c^s(\mathbf{c_s} + \mathbf{c_v}) + b_c^s\big) \quad (7)$$

$$\mathbf{d} = \mathbf{r} \otimes \mathbf{c} \quad (8)$$

where $\otimes$ denotes *element-wise* vector multiplication. The dot product, which may seem like the more intuitive similarity metric, would reduce the rich set of features in $\mathbf{d}$ to a single scalar. The element-wise multiplication allows the downstream network to make better use of its parameters by learning non-linear interactions between sets of features in $\mathbf{r}$ and $\mathbf{c}$.[2]

3.3 Context Modelling

This 'decoder' does not yet suffice to extract intents from utterances in human-machine dialogue. To understand some queries, the belief tracker must be aware of *context*, i.e. the flow of dialogue leading up to the latest user utterance. While all previous system and user utterances are important, the most relevant one is the last system utterance, in which the dialogue system could have performed (among others) one of the following two *system acts*:

1. **System Request**: The system asks the user about the value of a specific slot T_q. If the system utterance is: *'what price range would you like?'* and the user answers with *any*, the model must infer the reference to *price range*, and not to other slots such as *area* or *food*.

2. **System Confirm:** The system asks the user to confirm whether a specific slot-value pair (T_s, T_v) is part of their desired constraints. For example, if the user responds to *'how about Turkish food?'* with *'yes'*, the model must be aware of the system act in order to correctly update the belief state.

If we make the Markovian decision to only consider the last set of system acts, we can incorporate context modelling into the NBT. Let $\mathbf{t_q}$ and $(\mathbf{t_s}, \mathbf{t_v})$ be the word vectors of the arguments for the system request and confirm acts (zero vectors if none). The model computes the following measures of similarity between the system acts, candidate pair $(\mathbf{c_s}, \mathbf{c_v})$ and utterance representation $\mathbf{r}$:

$$\mathbf{m_r} = (\mathbf{c_s} \cdot \mathbf{t_q})\mathbf{r} \quad (9)$$

$$\mathbf{m_c} = (\mathbf{c_s} \cdot \mathbf{t_s})(\mathbf{c_v} \cdot \mathbf{t_v})\mathbf{r} \quad (10)$$

where $\cdot$ denotes dot product. The computed similarity terms act as gating mechanisms which only pass the utterance representation through if the system asked about the current candidate slot or slot-value pair. This type of interaction is particularly useful for the confirm system act: if the system asks the user to confirm, the user is likely not to mention any slot values, but to just respond affirmatively or negatively. This means that the model must consider the *three-way interaction* between the utterance, candidate slot-value pair and the slot value pair offered by the system. If (and only if) the latter two are the same should the model consider the affirmative or negative polarity of the user utterance when making the subsequent binary decision.

Binary Decision Maker The intermediate representations are passed through another hidden layer and then combined. If $\phi_{dim}(\mathbf{x}) = \sigma(W\mathbf{x} + b)$ is a layer which maps input vector $\mathbf{x}$ to a vector of size dim, the input to the final binary softmax (which represents the decision) is given by:

$$\mathbf{y} = \phi_2\big(\phi_{100}(\mathbf{d}) + \phi_{100}(\mathbf{m_r}) + \phi_{100}(\mathbf{m_c})\big)$$

4 Belief State Update Mechanism

In spoken dialogue systems, belief tracking models operate over the output of automatic speech recognition (ASR). Despite improvements to speech

[2] We also tried to concatenate $\mathbf{r}$ and $\mathbf{c}$ and pass that vector to the downstream decision-making neural network. However, this set-up led to very weak performance since our relatively small datasets did not suffice for the network to learn to model the interaction between the two feature vectors.

recognition, the need to make the most out of imperfect ASR will persist as dialogue systems are used in increasingly noisy environments.

In this work, we define a simple rule-based belief state update mechanism which can be applied to ASR N-best lists. For dialogue turn t, let sys^{t-1} denote the preceding system output, and let h^t denote the list of N ASR hypotheses h_i^t with posterior probabilities p_i^t. For any hypothesis h_i^t, slot s and slot value $v \in V_s$, NBT models estimate $\mathbb{P}(s, v \mid h_i^t, sys^{t-1})$, which is the (turn-level) probability that (s, v) was expressed in the given hypothesis. The predictions for N such hypotheses are then combined as:

$$\mathbb{P}(s, v \mid h^t, sys^{t-1}) = \sum_{i=1}^{N} p_i^t \, \mathbb{P}\left(s, v \mid h_i^t, sys^t\right)$$

This turn-level belief state estimate is then combined with the (cumulative) belief state up to time $(t-1)$ to get the updated belief state estimate:

$$\begin{aligned}\mathbb{P}(s, v \mid h^{1:t}, sys^{1:t-1}) \; = \; & \lambda \, \mathbb{P}\left(s, v \mid h^t, sys^{t-1}\right) \\ & + (1-\lambda) \, \mathbb{P}\left(s, v \mid h^{1:t-1}, sys^{1:t-2}\right)\end{aligned}$$

where λ is the coefficient which determines the relative weight of the turn-level and previous turns' belief state estimates.[3] For slot s, the set of its *detected values* at turn t is then given by:

$$V_s^t = \{v \in V_s \mid \mathbb{P}\left(s, v \mid h^{1:t}, sys^{1:t-1}\right) \geq 0.5\}$$

For informable (i.e. goal-tracking) slots, the value in V_s^t with the highest probability is chosen as the current goal (if $V_s^t \neq \{\emptyset\}$). For requests, all slots in V_{req}^t are deemed to have been requested. As requestable slots serve to model single-turn user queries, they require no belief tracking across turns.

5 Experiments

5.1 Datasets

Two datasets were used for training and evaluation. Both consist of user conversations with task-oriented dialogue systems designed to help users find suitable restaurants around Cambridge, UK. The two corpora share the same domain ontology, which contains three *informable* (i.e. goal-tracking) slots: FOOD, AREA and PRICE. The users can specify values for these slots in order to find restaurants which best meet their criteria. Once the system suggests a restaurant, the users can ask about the values of up to eight *requestable* slots (PHONE NUMBER, ADDRESS, etc.). The two datasets are:

1. **DSTC2**: We use the transcriptions, ASR hypotheses and turn-level semantic labels provided for the Dialogue State Tracking Challenge 2 (Henderson et al., 2014a). The official transcriptions contain various spelling errors which we corrected manually; the cleaned version of the dataset is available at `mi.eng.cam.ac.uk/~nm480/dstc2-clean.zip`. The training data contains 2207 dialogues and the test set consists of 1117 dialogues. We train NBT models on transcriptions but report belief tracking performance on test set ASR hypotheses provided in the original challenge.

2. **WOZ 2.0**: Wen et al. (2017) performed a Wizard of Oz style experiment in which Amazon Mechanical Turk users assumed the role of the system or the user of a task-oriented dialogue system based on the DSTC2 ontology. Users typed instead of using speech, which means performance in the WOZ experiments is more indicative of the model's capacity for semantic understanding than its robustness to ASR errors. Whereas in the DSTC2 dialogues users would quickly adapt to the system's (lack of) language understanding capability, the WOZ experimental design gave them freedom to use more sophisticated language. We expanded the original WOZ dataset from Wen et al. (2017) using the same data collection procedure, yielding a total of 1200 dialogues. We divided these into 600 training, 200 validation and 400 test set dialogues. The WOZ 2.0 dataset is available at `mi.eng.cam.ac.uk/~nm480/woz_2.0.zip`.

Training Examples The two corpora are used to create training data for two separate experiments. For each dataset, we iterate over all train set utterances, generating one example for *each* of the slot-value pairs in the ontology. An example consists of a transcription, its context (i.e. list of preceding system acts) and a candidate slot-value pair. The binary label for each example indicates whether or not its utterance and context express the example's candidate pair. For instance, '*I would like Irish*

[3] This coefficient was tuned on the DSTC2 development set. The best performance was achieved with $\lambda = 0.55$.

food' would generate a positive example for candidate pair FOOD=IRISH, and a negative example for every other slot-value pair in the ontology.

Evaluation We focus on two key evaluation metrics introduced in (Henderson et al., 2014a):

1. **Goals** ('joint goal accuracy'): the proportion of dialogue turns where all the user's search goal constraints were correctly identified;

2. **Requests**: similarly, the proportion of dialogue turns where user's requests for information were identified correctly.

5.2 Models

We evaluate two NBT model variants: NBT-DNN and NBT-CNN. To train the models, we use the Adam optimizer (Kingma and Ba, 2015) with cross-entropy loss, backpropagating through all the NBT subcomponents while keeping the pre-trained word vectors fixed (in order to allow the model to deal with unseen words at test time). The model is trained separately for each slot. Due to the high class bias (most of the constructed examples are negative), we incorporate a fixed number of positive examples in each mini-batch.[4]

Baseline Models For each of the two datasets, we compare the NBT models to:

1. A baseline system that implements a well-known competitive delexicalisation-based model for that dataset. For DSTC2, the model is that of Henderson et al. (2014c; 2014d). This model is an n-gram based neural network model with recurrent connections between turns (but not inside utterances) which replaces occurrences of slot names and values with generic delexicalised features. For WOZ 2.0, we compare the NBT models to a more sophisticated belief tracking model presented in (Wen et al., 2017). This model uses an RNN for belief state updates and a CNN for turn-level feature extraction. Unlike NBT-CNN, their CNN operates not over vectors, but over delexicalised features akin to those used by Henderson et al. (2014c).

2. The same baseline model supplemented with a task-specific semantic dictionary (produced by the baseline system creators). The two dictionaries are available at `mi.eng.cam.ac.uk/~nm480/sem-dict.zip`. The DSTC2 dictionary contains only three rephrasings. Nonetheless, the use of these rephrasings translates to substantial gains in DST performance (see Sect. 6.1). We believe this result supports our claim that the vocabulary used by Mechanical Turkers in DSTC2 was constrained by the system's inability to cope with lexical variation and ASR noise. The WOZ dictionary includes 38 rephrasings, showing that the unconstrained language used by Mechanical Turkers in the Wizard-of-Oz setup requires more elaborate lexicons.

Both baseline models map exact matches of ontology-defined intents (and their lexicon-specified rephrasings) to one-hot delexicalised n-gram features. This means that pre-trained vectors cannot be incorporated directly into these models.

6 Results

6.1 Belief Tracking Performance

Table 1 shows the performance of NBT models trained and evaluated on DSTC2 and WOZ 2.0 datasets. The NBT models outperformed the baseline models in terms of both joint goal and request accuracies. For goals, the gains are *always* statistically significant (paired t-test, $p < 0.05$). Moreover, there was no statistically significant variation between the NBT and the lexicon-supplemented models, showing that the NBT can handle semantic relations which otherwise had to be explicitly encoded in semantic dictionaries.

While the NBT performs well across the board, we can compare its performance on the two datasets to understand its strengths. The improvement over the baseline is greater on WOZ 2.0, which corroborates our intuition that the NBT's ability to learn linguistic variation is vital for this dataset containing longer sentences, richer vocabulary and no ASR errors. By comparison, the language of the subjects in the DSTC2 dataset is less rich, and compensating for ASR errors is the main hurdle: given access to the DSTC2 test set transcriptions, the NBT models' goal accuracy rises to 0.96. This

[4] Model hyperparameters were tuned on the respective validation sets. For both datasets, the initial Adam learning rate was set to 0.001, and $\frac{1}{8}$th of positive examples were included in each mini-batch. The batch size did not affect performance: it was set to 256 in all experiments. Gradient clipping (to $[-2.0, 2.0]$) was used to handle exploding gradients. Dropout (Srivastava et al., 2014) was used for regularisation (with 50% dropout rate on all intermediate representations). Both NBT models were implemented in TensorFlow (Abadi et al., 2015).

DST Model	DSTC2		WOZ 2.0	
	Goals	Requests	Goals	Requests
Delexicalisation-Based Model	69.1	95.7	70.8	87.1
Delexicalisation-Based Model + Semantic Dictionary	72.9*	95.7	83.7*	87.6
NEURAL BELIEF TRACKER: NBT-DNN	72.6*	96.4	**84.4***	91.2*
NEURAL BELIEF TRACKER: NBT-CNN	**73.4***	**96.5**	84.2*	**91.6***

Table 1: DSTC2 and WOZ 2.0 test set accuracies for: **a)** joint goals; and **b)** turn-level requests. The asterisk indicates statistically significant improvement over the baseline trackers (paired t-test; $p < 0.05$).

indicates that future work should focus on better ASR compensation if the model is to be deployed in environments with challenging acoustics.

6.2 The Importance of Word Vector Spaces

The NBT models use the semantic relations embedded in the pre-trained word vectors to handle semantic variation and produce high-quality intermediate representations. Table 2 shows the performance of NBT-CNN[5] models making use of three different word vector collections: **1)** 'random' word vectors initialised using the XAVIER initialisation (Glorot and Bengio, 2010); **2)** distributional GloVe vectors (Pennington et al., 2014), trained using co-occurrence information in large textual corpora; and **3)** *semantically specialised* Paragram-SL999 vectors (Wieting et al., 2015), which are obtained by injecting *semantic similarity constraints* from the Paraphrase Database (Ganitkevitch et al., 2013) into the distributional GloVe vectors in order to improve their semantic content.

The results in Table 2 show that the use of semantically specialised word vectors leads to considerable performance gains: Paragram-SL999 vectors (significantly) outperformed GloVe and XAVIER vectors for goal tracking on both datasets. The gains are particularly robust for noisy DSTC2 data, where both collections of pre-trained vectors consistently outperformed random initialisation. The gains are weaker for the noise-free WOZ 2.0 dataset, which seems to be large (and clean) enough for the NBT model to learn task-specific rephrasings and compensate for the lack of semantic content in the word vectors. For this dataset, GloVe vectors do not improve over the randomly initialised ones. We believe this happens because distributional models keep related, yet antonymous words close together (e.g. *north* and *south*, *expensive* and *inexpensive*), offsetting the useful semantic content embedded in this vector spaces.

[5]The NBT-DNN model showed the same trends. For brevity, Table 2 presents only the NBT-CNN figures.

Word Vectors	DSTC2		WOZ 2.0	
	Goals	Requests	Goals	Requests
XAVIER (No Info.)	64.2	81.2	81.2	90.7
GloVe	69.0*	96.4*	80.1	91.4
Paragram-SL999	**73.4***	**96.5***	**84.2***	**91.6**

Table 2: DSTC2 and WOZ 2.0 test set performance (*joint goals* and *requests*) of the NBT-CNN model making use of three different word vector collections. The asterisk indicates statistically significant improvement over the baseline XAVIER (random) word vectors (paired t-test; $p < 0.05$).

7 Conclusion

In this paper, we have proposed a novel neural belief tracking (NBT) framework designed to overcome current obstacles to deploying dialogue systems in real-world dialogue domains. The NBT models offer the known advantages of coupling Spoken Language Understanding and Dialogue State Tracking, without relying on hand-crafted semantic lexicons to achieve state-of-the-art performance. Our evaluation demonstrated these benefits: the NBT models match the performance of models which make use of such lexicons and vastly outperform them when these are not available. Finally, we have shown that the performance of NBT models improves with the semantic quality of the underlying word vectors. To the best of our knowledge, we are the first to move past intrinsic evaluation and show that *semantic specialisation* boosts performance in downstream tasks.

In future work, we intend to explore applications of the NBT for multi-domain dialogue systems, as well as in languages other than English that require handling of complex morphological variation.

Acknowledgements

The authors would like to thank Ivan Vulić, Ulrich Paquet, the Cambridge Dialogue Systems Group and the anonymous ACL reviewers for their constructive feedback and helpful discussions.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems.

Dan Bohus and Alex Rudnicky. 2006. A "k hypotheses + other" belief updating model. In *Proceedings of the AAAI Workshop on Statistical and Empirical Methods in Spoken Dialogue Systems*.

Asli Celikyilmaz and Dilek Hakkani-Tur. 2015. Convolutional Neural Network Based Semantic Tagging with Entity Embeddings. In *Proceedings of NIPS Workshop on Machine Learning for Spoken Language Understanding and Interaction*.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Franck Dernoncourt, Ji Young Lee, Trung H. Bui, and Hung H. Bui. 2016. Robust dialog state tracking for large ontologies. In *Proceedings of IWSDS*.

Ondřej Dušek and Filip Jurčíček. 2015. Training a Natural Language Generator From Unaligned Data. In *Proceedings of ACL*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL HLT*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*.

Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. 2012. Discriminative Spoken Language Understanding Using Word Confusion Networks. In *Spoken Language Technology Workshop, 2012. IEEE*.

Matthew Henderson, Blaise Thomson, and Jason D. Wiliams. 2014a. The Second Dialog State Tracking Challenge. In *Proceedings of SIGDIAL*.

Matthew Henderson, Blaise Thomson, and Jason D. Wiliams. 2014b. The Third Dialog State Tracking Challenge. In *Proceedings of IEEE SLT*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014c. Robust Dialog State Tracking using Delexicalised Recurrent Neural Networks and Unsupervised Adaptation. In *Proceedings of IEEE SLT*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014d. Word-Based Dialog State Tracking with Recurrent Neural Networks. In *Proceedings of SIGDIAL*.

Youngsoo Jang, Jiyeon Ham, Byung-Jun Lee, Youngjae Chang, and Kee-Eung Kim. 2016. Neural dialog state tracker for large ontologies by attention mechanism. In *Proceedings of IEEE SLT*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of ACL*.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of ICLR*.

Byung-Jun Lee and Kee-Eung Kim. 2016. Dialog History Construction with Long-Short Term Memory for Robust Generative Dialog State Tracking. *Dialogue & Discourse* 7(3):47–64.

Bing Liu and Ian Lane. 2016a. Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling. In *Proceedings of Interspeech*.

Bing Liu and Ian Lane. 2016b. Joint Online Spoken Language Understanding and Language Modeling with Recurrent Neural Networks. In *Proceedings of SIGDIAL*.

Fei Liu and Julien Perez. 2017. Gated End-to-End Memory Networks. In *Proceedings of EACL*.

F. Mairesse, M. Gasic, F. Jurcicek, S. Keizer, B. Thomson, K. Yu, and S. Young. 2009. Spoken Language Understanding from Unaligned Data using Discriminative Classification Models. In *Proceedings of ICASSP*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Dong Yu, and Geoffrey Zweig. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23(3):530–539.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting Word Vectors to Linguistic Constraints. In *Proceedings of HLT-NAACL*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain Dialog State Tracking using Recurrent Neural Networks. In *Proceedings of ACL*.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*.

Baolin Peng, Kaisheng Yao, Li Jing, and Kam-Fai Wong. 2015. Recurrent Neural Networks with External Memory for Language Understanding. In *Proceedings of the National CCF Conference on Natural Language Processing and Chinese Computing*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of EMNLP*.

Julien Perez. 2016. Spectral decomposition method of dialog state tracking via collective matrix factorization. *Dialogue & Discourse* 7(3):34–46.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using Memory Network. In *Proceedings of EACL*.

Christian Raymond and Giuseppe Ricardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Proceedings of Interspeech*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *ICLR*.

Iman Saleh, Shafiq Joty, Lluís Màrquez, Alessandro Moschitti, Preslav Nakov, Scott Cyphers, and Jim Glass. 2014. A study of using syntactic and semantic structures for concept segmentation and labeling. In *Proceedings of COLING*.

Hongjie Shi, Takashi Ushio, Mitsuru Endo, Katsuyoshi Yamagami, and Noriaki Horii. 2016. Convolutional Neural Networks for Multi-topic Dialog State Tracking. In *Proceedings of IWSDS*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* .

Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016a. Continuously learning neural dialogue management. In *arXiv preprint: 1606.02689*.

Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016b. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of ACL*.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014. The SJTU System for Dialog State Tracking Challenge 2. In *Proceedings of SIGDIAL*.

Kai Sun, Qizhe Xie, and Kai Yu. 2016. Recurrent Polynomial Network for Dialogue State Tracking. *Dialogue & Discourse* 7(3):65–88.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language* .

Gokhan Tur, Anoop Deoras, and Dilek Hakkani-Tur. 2013. Semantic Parsing Using Word Confusion Networks With Conditional Random Fields. In *Proceedings of Interspeech*.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *TACL* 2:547–559.

Miroslav Vodolán, Rudolf Kadlec, and Jan Kleindienst. 2017. Hybrid Dialog State Tracker with ASR Features. In *Proceedings of EACL*.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016. Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *Proceedings of ICASSP*.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of ACL*.

Wayne Wang. 1994. Extracting Information From Spontaneous Speech. In *Proceedings of Interspeech*.

Zhuoran Wang and Oliver Lemon. 2013. A Simple and Generic Belief Tracking Mechanism for the Dialog State Tracking Challenge: On the believability of observed information. In *Proceedings of SIGDIAL*.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. In *Proceedings of SIGDIAL*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of EMNLP*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *TACL* 3:345–358.

Jason D. Williams. 2014. Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of SIGDIAL*.

Jason D. Williams, Antoine Raux, and Matthew Henderson. 2016. The Dialog State Tracking Challenge series: A review. *Dialogue & Discourse* 7(3):4–33.

Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan W. Black. 2013. The Dialogue State Tracking Challenge. In *Proceedings of SIGDIAL*.

Jason D. Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language* 21:393–422.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *Proceedings of ASRU*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language* 24:150–174.

Xiaodong Zhang and Houfeng Wang. 2016. A Joint Model of Intent Determination and Slot Filling for Spoken Language Understanding. In *Proceedings of IJCAI*.

Lukas Zilka and Filip Jurcicek. 2015. Incremental LSTM-based dialog state tracker. In *Proceedings of ASRU*.

Exploiting Argument Information to Improve Event Detection via Supervised Attention Mechanisms

Shulin Liu[1,2], **Yubo Chen**[1,2], **Kang Liu**[1] and **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
{shulin.liu, yubo.chen, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

This paper tackles the task of event detection (ED), which involves identifying and categorizing events. We argue that arguments provide significant clues to this task, but they are either completely ignored or exploited in an indirect manner in existing detection approaches. In this work, we propose to exploit argument information explicitly for ED via supervised attention mechanisms. In specific, we systematically investigate the proposed model under the supervision of different attention strategies. Experimental results show that our approach advances state-of-the-arts and achieves the best F_1 score on ACE 2005 dataset.

1 Introduction

In the ACE (Automatic Context Extraction) event extraction program, an event is represented as a structure comprising an **event trigger** and a set of **arguments**. This work tackles event detection (ED) task, which is a crucial part of event extraction (EE) and focuses on identifying event triggers and categorizing them. For instance, in the sentence "**He *died* in the hospital**", an ED system is expected to detect a ***Die*** event along with the trigger word "*died*". Besides, the task of EE also includes event argument extraction (AE), which involves event argument identification and role classification. In the above sentence, the arguments of the event include "He"($Role = Person$) and "hospital"($Role = Place$). However, this paper does not focus on AE and only tackles the former task.

According to the above definitions, event arguments seem to be not essentially necessary to ED. However, we argue that they are capable of providing significant clues for identifying and categorizing events. They are especially useful for ambiguous trigger words. For example, consider a sentence in ACE 2005 dataset:

> **Mohamad *fired* Anwar**, his **former protege**, in **1998**.

In this sentence, "*fired*" is the trigger word and the other bold words are event arguments. The correct type of the event triggered by "*fired*" in this case is ***End-Position***. However, it might be easily misidentified as ***Attack*** because "*fired*" is a multivocal word. In this case, if we consider the phrase "former protege", which serves as an argument ($Role = Position$) of the target event, we would have more confidence in predicting it as an ***End-Position*** event.

Unfortunately, most existing methods performed event detection individually, where the annotated arguments in training set are totally ignored (Ji and Grishman, 2008; Gupta and Ji, 2009; Hong et al., 2011; Chen et al., 2015; Nguyen and Grishman, 2015; Liu et al., 2016a,b; Nguyen and Grishman, 2016). Although some joint learning based methods have been proposed, which tackled event detection and argument extraction simultaneously (Riedel et al., 2009; Li et al., 2013; Venugopal et al., 2014; Nguyen et al., 2016), these approaches usually only make remarkable improvements to AE, but insignificant to ED. Table 1 illustrates our observations. Li et al. (2013) and Nguyen et al. (2016) are state-of-the-art joint models in symbolic and embedding methods for event extraction, respectively. Compared with state-of-the-art pipeline systems, both join-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1789–1798
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1164

Methods		ED	AE
Symbolic Methods	Hong's pipeline (2011)	68.3	48.3
	Li's joint (2013)	67.5	52.7
Embedding Methods	Chen's pipeline (2015)	69.1	53.5
	Nguyen's joint (2016)	69.3	55.4

Table 1: Performances of pipeline and joint approaches on ACE 2005 dataset. The pipeline method in each group was the state-of-the-art system when the corresponding joint method was proposed.

t methods achieved remarkable improvements on AE (over 1.9 points), whereas achieved insignificant improvements on ED (less than 0.2 points). The symbolic joint method even performed worse (67.5 vs. 68.3) than pipeline system on ED.

We believe that this phenomenon may be caused by the following two reasons. On the one hand, since joint methods simultaneously solve ED and AE, methods following this paradigm usually combine the loss functions of these two tasks and are jointly trained under the supervision of annotated triggers and arguments. However, training corpus contains much more annotated arguments than triggers (about 9800 arguments and 5300 triggers in ACE 2005 dataset) because each trigger may be along with multiple event arguments. Thus, the unbalanced data may cause joint models to favor AE task. On the other hand, in implementation, joint models usually pre-predict several potential triggers and arguments first and then make global inference to select correct items. When pre-predicting potential triggers, almost all existing approaches do not leverage any argument information. In this way, ED does hardly benefit from the annotated arguments. By contrast, the component for pre-prediction of arguments always exploits the extracted trigger information. Thus, we argue that annotated arguments are actually used for AE, not for ED in existing joint methods, which is also the reason we call it an indirect way to use arguments for ED.

Contrast to joint methods, this paper proposes to exploit argument information explicitly for ED. We have analyzed that arguments are capable of providing significant clues to ED, which gives us an enlightenment that arguments should be focused on when performing this task. Therefore, we propose a neural network based approach to detect events in texts. And in the proposed approach, we adopt a supervised attention mechanism to achieve this goal, where argument words are expected to acquire more attention than other words. The attention value of each word in a given sentence is calculated by an operation between the current word and the target trigger candidate. Specifically, in training procedure, we first construct gold attentions for each trigger candidate based on annotated arguments. Then, treating gold attentions as the supervision to train the attention mechanism, we learn attention and event detector jointly both in supervised manner. In testing procedure, we use the ED model with learned attention mechanisms to detect events.

In the experiment section, we systematically conduct comparisons on a widely used benchmark dataset ACE2005[1]. In order to further demonstrate the effectiveness of our approach, we also use events from FrameNet (FN) (F. Baker et al., 1998) as extra training data, as the same as Liu et al. (2016a) to alleviate the data-sparseness problem for ED to augment the performance of the proposed approach. The experimental results demonstrate that the proposed approach is effective for ED task, and it outperforms state-of-the-art approaches with remarkable gains.

To sum up, our main contributions are: (1) we analyze the problem of joint models on the task of ED, and propose to use the annotated argument information explicitly for this task. (2) to achieve this goal, we introduce a supervised attention based ED model. Furthermore, we systematically investigate different attention strategies for the proposed model. (3) we improve the performance of ED and achieve the best performance on the widely used benchmark dataset ACE 2005.

2 Task Description

The ED task is a subtask of ACE event evaluations where an event is defined as a specific occurrence involving one or more participants. Event extraction task requires certain specified types of events, which are mentioned

[1]https://catalog.ldc.upenn.edu/LDC2006T06

in the source language data, be detected. We firstly introduce some ACE terminologies to facilitate the understanding of this task:

Entity: an object or a set of objects in one of the semantic categories of interests.

Entity mention: a reference to an entity (typically, a noun phrase).

Event trigger: the main word that most clearly expresses an event occurrence.

Event arguments: the mentions that are involved in an event (participants).

Event mention: a phrase or sentence within which an event is described, including the trigger and arguments.

The goal of ED is to identify event triggers and categorize their event types. For instance, in the sentence "**He *died* in the hospital**", an ED system is expected to detect a ***Die*** event along with the trigger word "*died*". The detection of event arguments "He"($Role = Person$) and "hospital"($Role = Place$) is not involved in the ED task. The 2005 ACE evaluation included 8 super types of events, with 33 subtypes. Following previous work, we treat these simply as 33 separate event types and ignore the hierarchical structure among them.

3 The Proposed Approach

Similar to existing work, we model ED as a multi-class classification task. In detail, given a sentence, we treat every token in that sentence as a trigger candidate, and our goal is to classify each of these candidates into one of 34 classes (33 event types plus an NA class).

In our approach, every word along with its context, which includes the contextual words and entities, constitute an event trigger candidate. Figure 1 describes the architecture of the proposed approach, which involves two components: (i) Context Representation Learning (CRL), which reveals the representation of both contextual words and entities via attention mechanisms; (ii) Event Detector (ED), which assigns an event type (including the NA type) to each candidate based on the learned contextual representations.

3.1 Context Representation Learning

In order to prepare for Context Representation Learning (CRL), we limit the context to a fixed length by trimming longer sentences and padding shorter sentences with a special token when necessary. Let n be the fixed length and w_0 be the current candidate trigger word, then its contextual words $\mathbf{C_w}$ is $[w_{-\frac{n}{2}}, w_{-\frac{n}{2}+1}, ..., w_{-1}, w_1, ..., w_{\frac{n}{2}-1}, w_{\frac{n}{2}}]$[2], and its contextual entities, which is the corresponding entity types (including an NA type) of $\mathbf{C_w}$, is $[e_{-\frac{n}{2}}, e_{-\frac{n}{2}+1}, ..., e_{-1}, e_1, ..., e_{\frac{n}{2}-1}, e_{\frac{n}{2}}]$. For convenience, we use w to denote the current word, $[w_1, w_2, ..., w_n]$ to denote the contextual words $\mathbf{C_w}$ and $[e_1, e_2, ..., e_n]$ to denote the contextual entities $\mathbf{C_e}$ in figure 1. Note that, both w, $\mathbf{C_w}$ and $\mathbf{C_e}$ mentioned above are originally in symbolic representation. Before entering CRL component, we transform them into real-valued vector by looking up word embedding table and entity type embedding table. Then we calculate attention vectors for both contextual words and entities by performing operations between the current word w and its contexts. Finally, the contextual words representation c_w and contextual entities representation c_e are formed by the weighted sum of the corresponding embeddings of each word and entity in $\mathbf{C_w}$ and $\mathbf{C_e}$, respectively. We will give the details in the fol-

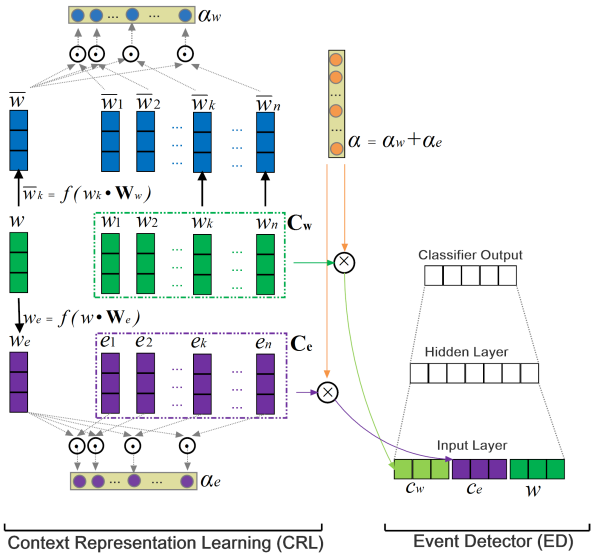


Figure 1: The architecture of the proposed approach for event detection. In this figure, w is the candidate word, $[w_1, ..., w_n]$ is the contextual words of w, and $[e_1, ..., e_n]$ is the corresponding entity types of $[w_1, ..., w_n]$.

[2]The current candidate trigger word w_0 is not included in the context.

lowing subsections.

3.1.1 Word Embedding Table

Word embeddings learned from a large amount of unlabeled data have been shown to be able to capture the meaningful semantic regularities of words (Bengio et al., 2003; Erhan et al., 2010). This paper uses the learned word embeddings as the source of basic features. Specifically, we use the Skip-gram model (Mikolov et al., 2013) to learn word embeddings on the NYT corpus[3].

3.1.2 Entity Type Embedding Table

The ACE 2005 corpus annotated not only events but also entities for each given sentence. Following existing work (Li et al., 2013; Chen et al., 2015; Nguyen and Grishman, 2015), we exploit the annotated entity information in our ED system. We randomly initialize embedding vector for each entity type (including the NA type) and update it in training procedure.

3.1.3 Representation Learning

In this subsection, we illustrate our proposed approach to learn representations of both contextual words and entities, which serve as inputs to the following event detector component. Recall that, we use the matrix $\mathbf{C_w}$ and $\mathbf{C_e}$ to denote contextual words and contextual entities, respectively.

As illustrated in figure 1, the CRL component needs three inputs: the current candidate trigger word w, the contextual words $\mathbf{C_w}$ and the contextual entities $\mathbf{C_e}$. Then, two attention vectors, which reflect different aspects of the context, are calculated in the next step.

The contextual word attention vector α_w is computed based on the current word w and its contextual words $\mathbf{C_w}$. We firstly transform each word w_k (including w and every word in $\mathbf{C_w}$) into a hidden representation $\overline{w}_k$ by the following equation:

$$\overline{w}_k = f(w_k \cdot W_w) \tag{1}$$

where $f(\cdot)$ is a non-linear function such as the hyperbolic tangent, and W_w is the transformation matrix. Then, we use the hidden representations to compute the attention value for each word in $\mathbf{C_w}$:

$$\alpha_w^k = \frac{\exp(\overline{w} \cdot \overline{w}_k^{\mathrm{T}})}{\sum_i \exp(\overline{w} \cdot \overline{w}_i^{\mathrm{T}})} \tag{2}$$

The contextual entity attention vector α_e is calculated with a similar method to α_w.

$$\alpha_e^k = \frac{\exp(w_e \cdot e_k^{\mathrm{T}})}{\sum_i \exp(w_e \cdot e_i^{\mathrm{T}})} \tag{3}$$

Note that, we do not use the entity information of the current candidate token to compute the attention vector. The reason is that only a small percentage of true event triggers are entities[4]. Therefore, the entity type of a candidate trigger is meaningless for ED. Instead, we use w_e, which is calculated by transforming w from the word space into the entity type space, as the attention source.

We combine α_w and α_e to obtain the final attention vector, $\alpha = \alpha_w + \alpha_e$. Finally, the contextual words representation c_w and the contextual entities representation c_e are formed by weighted sum of $\mathbf{C_w}$ and $\mathbf{C_e}$, respectively:

$$c_w = \mathbf{C_w}\alpha^{\mathrm{T}} \tag{4}$$

$$c_e = \mathbf{C_e}\alpha^{\mathrm{T}} \tag{5}$$

3.2 Event Detector

As illustrated in figure 1, we employ a three-layer (an input layer, a hidden layer and a softmax output layer) Artificial Neural Networks (ANNs) (Hagan et al., 1996) to model the ED task, which has been demonstrated very effective for event detection by Liu et al. (2016a).

3.2.1 Basic ED Model

Given a sentence, as illustrated in figure 1, we concatenate the embedding vectors of the context (including contextual words and entities) and the current candidate trigger to serve as the input to ED model. Then, for a given input sample $\mathbf{x}$, ANN with parameter θ outputs a vector $\mathbf{O}$, where the i-th value o_i of $\mathbf{O}$ is the confident score for classifying $\mathbf{x}$ to the i-th event type. To obtain the conditional probability $p(i|\mathbf{x}, \theta)$, we apply a softmax operation over all event types:

$$p(i|\mathbf{x}, \theta) = \frac{e^{o_i}}{\sum_{k=1}^{m} e^{o_k}} \tag{6}$$

[3] https://catalog.ldc.upenn.edu/LDC2008T19

[4] Only 10% of triggers in ACE 2005 are entities.

Given all of our (suppose T) training instances $(\mathbf{x}^{(\mathbf{i})}; y^{(i)})$, we can then define the negative log-likelihood loss function:

$$J(\theta) = -\sum_{i=1}^{T} \log p(y^{(i)}|\mathbf{x}^{(\mathbf{i})}, \theta) \quad (7)$$

We train the model by using a simple optimization technique called stochastic gradient descent (SGD) over shuffled mini-batches with the Adadelta rule (Zeiler, 2012). Regularization is implemented by a dropout (Kim, 2014; Hinton et al., 2012) and L_2 norm.

3.2.2 Supervised Attention

In this subsection, we introduce supervised attention to explicitly use annotated argument information to improve ED. Our basic idea is simple: argument words should acquire more attention than other words. To achieve this goal, we first construct vectors using annotated arguments as the gold attentions. Then, we employ them as supervision to train the attention mechanism.

Constructing Gold Attention Vectors

Our goal is to encourage argument words to obtain more attention than other words. To achieve this goal, we propose two strategies to construct gold attention vectors:

S1: only pay attention to argument words. That is, all argument words in the given context obtain the same attention, whereas other words get no attention. For candidates without any annotated arguments in context (such as negative samples), we force all entities to average the whole attention. Figure 2 illustrates the details, where α^* is the final gold attention vector.

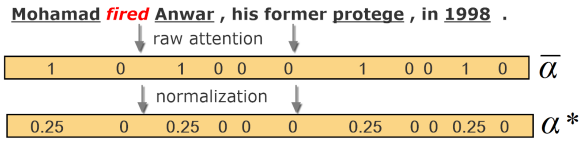


Figure 2: An example of S1 to construct gold attention vector. The word *fired* is the trigger candidate, and underline words are arguments of *fired* annotated in the corpus.

S2: pay attention to both arguments and the words around them. The assumption is that, not only arguments are important to ED, the words around them are also helpful. And the nearer a word is to arguments, the more attention it should obtain. Inspired by Mi et al. (2016), we use a gaussian distribution $g(\cdot)$ to model the attention distribution of words around arguments. In detail, given an instance, we first obtain the raw attention vector $\overline{\alpha}$ in the same manner as S1 (see figure 2). Then, we create a new vector $\alpha^{'}$ with all points initialized with zero, and for each $\overline{\alpha}_i = 1$, we update $\alpha^{'}$ by the following algorithm:

Algorithm 1: Updating $\alpha^{'}$

for $k \in \{-w, ..., 0, ..., w\}$ **do**
  $\alpha^{'}_{i+k} = \alpha^{'}_{i+k} + g(|k|, \mu, \sigma)$
end

where w is the window size of the attention mechanism and μ, σ are hyper-parameters of the gaussian distribution. Finally, we normalize $\alpha^{'}$ to obtain the target attention vector α^*. Similar with S1, we treat all entities in the context as arguments if the current candidate does not has any annotated arguments (such as netative samples).

Jointly Training ED and Attention

Given the gold attention α^* (see subsection 3.2.2) and the machine attention α produced by our model (see subsection 3.1.3), we employ the square error as the loss function of attentions:

$$D(\theta) = \sum_{i=1}^{T} \sum_{j=1}^{n} ({\alpha^*}^i_j - \alpha^i_j)^2 \quad (8)$$

Combining equation 7 and equation 8, we define the joint loss function of our proposed model as follows:

$$J^{'}(\theta) = J(\theta) + \lambda D(\theta) \quad (9)$$

where λ is a hyper-parameter for trade-off between J and D. Similar to basic ED model, we minimize the loss function $J^{'}(\theta)$ by using SGD over shuffled mini-batches with the Adadelta update rule.

4 Experiments

4.1 Dataset and Experimental Setup

Dataset

We conducted experiments on ACE 2005 dataset. For the purpose of comparison, we fol-

lowed the evaluation of (Li et al., 2013; Chen et al., 2015; Liu et al., 2016b): randomly selected 30 articles from different genres as the development set, and subsequently conducted a blind test on a separate set of 40 ACE 2005 newswire documents. We used the remaining 529 articles as our training set.

Hyper-parameter Setting

Hyper-parameters are tuned on the development dataset. We set the dimension of word embeddings to 200, the dimension of entity type embeddings to 50, the size of hidden layer to 300, the output size of word transformation matrix W_w in equation 1 to 200, the batch size to 100, the hyper-parameter for the L_2 norm to 10^{-6} and the dropout rate to 0.6. In addition, we use the standard normal distribution to model attention distributions of words around arguments, which means that $\mu = 0.0, \sigma = 1.0$, and the window size is set to 3 (see Subsection 3.2.2). The hyper-parameter λ in equation 9 is various for different attention strategies, we will give its setting in the next section.

4.2 Correctness of Our Assumption

In this section, we conduct experiments on ACE 2005 corpus to demonstrate the correctness of our assumption that argument information is crucial to ED. To achieve this goal, we design a series of systems for comparison.

ANN is the basic event detection model, in which the hyper-parameter λ is set to 0. This system does not employ argument information and computes attentions without supervision (see Subsection 3.1.3).

ANN-ENT assigns λ with 0, too. The difference is that it constructs the attention vector α by forcing all entities in the context to average the attention instead of computing it in the manner introduced in Subsection 3.1.3. Since all arguments are entities, this system is designed to investigate the effects of entities.

ANN-Gold1 uses the gold attentions constructed by strategy S1 in both **training** and **testing** procedure.

ANN-Gold2 is akin to *ANN-Gold1*, but uses the second strategy to construct its gold attentions.

Note that, in order to avoid the interference of attention mechanisms, the last two systems are designed to use argument information (via gold attentions) in both training and testing procedure. Thus both *ANN-Gold1* and *ANN-Gold2* assign λ with 0.

Methods	P	R	F_1
ANN	69.9	60.8	65.0
ANN-ENT	79.4	60.7	68.8
ANN-Gold1†	**81.9**	65.1	72.5
ANN-Gold2†	81.4	**66.9**	**73.4**

Table 2: Experimental results on ACE 2005 corpus. † designates the systems that employ argument information.

Table 2 compares these systems on ACE 2005 corpus. From the table, we observe that systems with argument information (the last two systems) significantly outperform systems without argument information (the first two systems), which demonstrates that argument information is very useful for this task. Moreover, since all arguments are entities, for preciseness we also investigate that whether *ANN-Gold1/2* on earth benefits from entities or arguments. Compared with *ANN-ENT* (revising that this system only uses entity information), *ANN-Gold1/2* performs much better, which illustrates that entity information is not enough and further demonstrates that argument information is necessary for ED.

4.3 Results on ACE 2005 Corpus

In this section, we conduct experiments on ACE 2005 corpus to demonstrate the effectiveness of the proposed approach. Firstly, we introduce systems implemented in this work.

ANN-S1 uses gold attentions constructed by strategy **S1** as supervision to learn attention. In our experiments, λ is set to 1.0.

ANN-S2 is akin to *ANN-S1*, but use strategy **S2** to construct gold attentions and the hyper-parameter λ is set to 5.0.

These two systems both employ supervised attention mechanisms. For comparison, we use an unsupervised-attention system *ANN* as our baseline, which is introduced in Subsection 4.2. In addition, we select the following state-of-the-art methods for comparison.

1). *Li's joint model* (Li et al., 2013) extracts events based on structure prediction. It is the best structure-based system.

Methods	P	R	F_1
Li's joint model (2013)	73.7	62.3	67.5
Liu's PSL (2016)	75.3	64.4	69.4
Liu's FN-Based (2016)	77.6	65.2	70.7
Ngyuen's joint (2016)	66.0	**73.0**	69.3
Skip-CNN (2016)	N/A		71.3
ANN	69.9	60.8	65.0
ANN-S1†	**81.4**	62.4	70.8
ANN-S2†	78.0	66.3	**71.7**

Table 3: Experimental results on ACE 2005. The first group illustrates the performances of state-of-the-art approaches. The second group illustrates the performances of the proposed approach. † designates the systems that employ arguments information.

2). *Liu's PSL* (Liu et al., 2016b) employs both latent local and global information for event detection. It is the best-reported feature-based system.

3). *Liu's FN-Based approach* (Liu et al., 2016a) leverages the annotated corpus of FrameNet to alleviate data sparseness problem of ED based on the observation that frames in FN are analogous to events in ACE.

4). *Ngyen's joint model* (Nguyen et al., 2016) employs a bi-directional RNN to jointly extract event triggers and arguments. It is the best-reported representation-based joint approach proposed on this task.

5). *Skip-CNN* (Nguyen and Grishman, 2016) introduces the non-consecutive convolution to capture non-consecutive k-grams for event detection. It is the best reported representation-based approach on this task.

Table 3 presents the experimental results on ACE 2005 corpus. From the table, we make the following observations:

1). *ANN* performs unexpectedly poorly, which indicates that unsupervised-attention mechanisms do not work well for ED. We believe the reason is that the training data of ACE 2005 corpus is insufficient to train a precise attention in an unsupervised manner, considering that data sparseness is an important issue of ED (Zhu et al., 2014; Liu et al., 2016a).

2). With argument information employed via supervised attention mechanisms, both *ANN-S1* and *ANN-S2* outperform *ANN* with remarkable gains, which illustrates the effectiveness of the proposed approach.

3). *ANN-S2* outperforms *ANN-S1*, but the latter achieves higher precision. It is not difficult to understand. On the one hand, strategy **S1** only focuses on argument words, which provides accurate information to identify event type, thus *ANN-S1* could achieve higher precision. On the other hand, **S2** focuses on both arguments and words around them, which provides more general but noised clues. Thus, *ANN-S2* achieves higher recall with a little loss of precision.

4). Compared with state-of-the-art approaches, our method *ANN-S2* achieves the best performance. We also perform a t-test ($p \leqslant 0.05$), which indicates that our method significantly outperforms all of the compared methods. Furthermore, another noticeable advantage of our approach is that it achieves much higher precision than state-of-the-arts.

4.4 Augmentation with FrameNet

Recently, Liu et al. (2016a) used events automatically detected from FN as extra training data to alleviate the data-sparseness problem for event detection. To further demonstrate the effectiveness of the proposed approach, we also use the events from FN to augment the performance of our approach.

In this work, we use the events published by Liu et al. (2016a)[5] as extra training data. However, their data can not be used in the proposed approach without further processing, because it lacks of both argument and entity information. Figure 3 shows several examples of this data.

##001 **Grandad was [dead] by then and so were the two great-uncles .**
Event Type: Die

##002 **I [divorced] my wife for smoking in the toilet !**
Event Type: Divorce

##003 **He was [executed] yesterday.**
Event Type: Execute

Figure 3: Examples of events detected from FrameNet (published by Liu et al. (2016a)).

Processing of Events from FN

Liu et al. (2016a) detected events from FrameNet based on the observation that frames in FN are analogous to events in ACE

[5]https://github.com/subacl/acl16

(lexical unit of a frame ↔ trigger of an event, frame elements of a frame ↔ arguments of an event). All events they published are also frames in FN. Thus, we treat frame elements annotated in FN corpus as event arguments. Since frames generally contain more frame elements than events, we only use core[6] elements in this work. Moreover, to obtain entity information, we use RPI Joint Information Extraction System[7] (Li et al., 2013, 2014; Li and Ji, 2014) to label ACE entity mentions.

Experimental Results

We use the events from FN as extra training data and keep the development and test datasets unchanged.Table 4 presents the experimental results.

Methods	P	R	F_1
ANN	69.9	60.8	65.0
ANN-S1	**81.4**	62.4	70.8
ANN-S2	78.0	66.3	71.7
ANN +FrameNet	72.5	61.7	66.7
ANN-S1 +FrameNet	80.1	63.6	70.9
ANN-S2 +FrameNet	76.8	**67.5**	**71.9**

Table 4: Experimental results on ACE 2005 corpus. "+FrameNet" designates the systems that are augmented by events from FrameNet.

From the results, we observe that:

1). With extra training data, *ANN* achieves significant improvements on F_1 measure (66.7 vs. 65.0). This result, to some extent, demonstrates the correctness of our assumption that the data sparseness problem is the reason that causes unsupervised attention mechanisms to be ineffective to ED.

2). Augmented with external data, both *ANN-S1* and *ANN-S2* achieve higher recall with a little loss of precision. This is to be expected. On the one hand, more positive training samples consequently make higher recall. On the other hand, the extra event samples are automatically extracted from FN, thus false-positive samples are inevitable to be involved, which may result in hurting the precision. Anyhow, with events from FN, our approach achieves higher F_1 score.

[6]FrameNet classifies frame elements into three groups: core, peripheral and extra-thematic.

[7]http://nlp.cs.rpi.edu/software/

5 Related Work

Event detection is an increasingly hot and challenging research topic in NLP. Generally, existing approaches could roughly be divided into two groups.

The first kind of approach tackled this task under the supervision of annotated triggers and entities, but totally ignored annotated arguments. The majority of existing work followed this paradigm, which includes feature-based methods and representation-based methods. Feature-based methods exploited a diverse set of strategies to convert classification clues (i.e., POS tags, dependency relations) into feature vectors (Ahn, 2006; Ji and Grishman, 2008; Patwardhan and Riloff, 2009; Gupta and Ji, 2009; Liao and Grishman, 2010; Hong et al., 2011; Liu et al., 2016b). Representation-based methods typically represent candidate event mentions by embeddings and feed them into neural networks (Chen et al., 2015; Nguyen and Grishman, 2015; Liu et al., 2016a; Nguyen and Grishman, 2016).

The second kind of approach, on the contrast, tackled event detection and argument extraction simultaneously, which is called joint approach (Riedel et al., 2009; Poon and Vanderwende, 2010; Li et al., 2013, 2014; Venugopal et al., 2014; Nguyen et al., 2016). Joint approach is proposed to capture internal and external dependencies of events, including trigger-trigger, argument-argument and trigger-argument dependencies. Theoretically, both ED and AE are expected to benefit from joint methods because triggers and arguments are jointly considered. However, in practice, existing joint methods usually only make remarkable improvements to AE, but insignificant to ED. Different from them, this work investigates the exploitation of argument information to improve the performance of ED.

6 Conclusions

In this work, we propose a novel approach to model argument information explicitly for ED via supervised attention mechanisms. Besides, we also investigate two strategies to construct gold attentions using the annotated arguments. To demonstrate the effectiveness of the proposed method, we systematically conduc-

t a series of experiments on the widely used benchmark dataset ACE 2005. Moreover, we also use events from FN to augment the performance of the proposed approach. Experimental results show that our approach outperforms state-of-the-art methods, which demonstrates that the proposed approach is effective for event detection.

Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 61533018) and the National Basic Research Program of China (No. 2014CB340503). And this research work was also supported by Google through focused research awards program.

References

David Ahn. 2006. Proceedings of the workshop on annotating and reasoning about time and events. Association for Computational Linguistics, pages 1–8. http://aclweb.org/anthology/W06-0901.

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

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via a dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 167–176. https://doi.org/10.3115/v1/P15-1017.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research* 11:625–660.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*. http://aclweb.org/anthology/C98-1013.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, pages 369–372. http://aclweb.org/anthology/P09-2093.

Martin T Hagan, Howard B Demuth, Mark H Beale, et al. 1996. *Neural network design*. Pws Pub. Boston.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* https://arxiv.org/abs/1207.0580.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1127–1136. http://aclweb.org/anthology/P11-1113.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, pages 254–262. http://aclweb.org/anthology/P08-1030.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 1746–1751. http://www.anthology.aclweb.org/D14-1181.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 402–412. https://doi.org/10.3115/v1/P14-1038.

Qi Li, Heng Ji, Yu HONG, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1846–1851. https://doi.org/10.3115/v1/D14-1198.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 73–82. http://aclweb.org/anthology/P13-1008.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 789–797. http://aclweb.org/anthology/P10-1081.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016a. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2134–2143. https://doi.org/10.18653/v1/P16-1201.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016b. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of the thirtieth AAAI Conference on Artificail Intelligence*. pages 2993–2999. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11990/12052.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. *arXiv preprint arXiv:1608.00112* https://arxiv.org/abs/1608.00112.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* https://arxiv.org/abs/1301.3781.

Huu Thien Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 300–309. https://doi.org/10.18653/v1/N16-1034.

Huu Thien Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 365–371. https://doi.org/10.3115/v1/P15-2060.

Huu Thien Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 886–891. http://aclweb.org/anthology/D16-1085.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 151–160. http://aclweb.org/anthology/D09-1016.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 813–821. http://aclweb.org/anthology/N10-1123.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. Proceedings of the bionlp 2009 workshop companion volume for shared task. Association for Computational Linguistics, pages 41–49. http://aclweb.org/anthology/W09-1406.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 831–843. https://doi.org/10.3115/v1/D14-1090.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701* https://arxiv.org/abs/1212.5701.

Zhu Zhu, Shoushan Li, Guodong Zhou, and Rui Xia. 2014. Bilingual event extraction: a case study on trigger type determination. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 842–847. https://doi.org/10.3115/v1/P14-2136.

Topical Coherence in LDA-based Models through Induced Segmentation

Hesam Amoualian
Univ. Grenoble Alps, CNRS, Grenoble INP - LIG
hesam.amoualian@imag.fr

Wei Lu
Singapore University of Technology and Design
luwei@sutd.edu.sg

Eric Gaussier
Univ. Grenoble Alps, CNRS, Grenoble INP - LIG
eric.gaussier@imag.fr

Georgios Balikas
Univ. Grenoble Alps, CNRS, Grenoble INP - LIG
georgios.balikas@imag.fr

Massih-Reza Amini
Univ. Grenoble Alps, CNRS, Grenoble INP - LIG
massih-reza.amini@imag.fr

Marianne Clausel
Univ. Grenoble Alps, CNRS, Grenoble INP - LJK
marianne.clausel@imag.fr

Abstract

This paper presents an LDA-based model that generates topically coherent segments within documents by jointly segmenting documents and assigning topics to their words. The coherence between topics is ensured through a copula, binding the topics associated to the words of a segment. In addition, this model relies on both document and segment specific topic distributions so as to capture fine grained differences in topic assignments. We show that the proposed model naturally encompasses other state-of-the-art LDA-based models designed for similar tasks. Furthermore, our experiments, conducted on six different publicly available datasets, show the effectiveness of our model in terms of perplexity, Normalized Pointwise Mutual Information, which captures the coherence between the generated topics, and the Micro F1 measure for text classification.

1 Introduction

Since the seminal works of Hofmann (1999) and Blei et al. (2003), there have been several developments in probabilistic topic models. Many extensions have indeed been proposed for different applications, including ad-hoc information retrieval (Wei and Croft, 2006), clustering search results (Zeng et al., 2004) and driving faceted browsing (Mimno and McCallum, 2007). In most of these studies, the initial exchangeability assumptions of PLSA and LDA, stipulating that words within a document are interdependent, has led to incoherent topic assignments within semantically meaningful text units, even though the importance of having topically coherent phrases is generally admitted (Griffiths et al., 2005). More recently, (Balikas et al., 2016b) has shown that binding topics, so as to obtain more coherent topic assignments, within such text segments as noun phrases improves the performance (*e.g.* in terms of perplexity) of LDA-based models. The question nevertheless remains as to which segmentation one should rely on.

Furthermore, text segments can refer to topics that are barely present in other parts of the document. For example, the segment "*the Kurdish regional capital*" in the sentence[1] "*A thousand protesters took to the main street in Erbil, the Kurdish regional capital, to condemn a new law requiring all public demonstrations to have government permits.*" refers to geography in a document that is mainly devoted to politics. Relying on a single topic distribution, as done in most previous studies including (Balikas et al., 2016b), may prevent one from capturing those segment specific topics.

In this paper, we propose a novel LDA-based model that automatically segments documents into topically coherent sequences of words. The coherence between topics is ensured through *copulas* (Elidan, 2013) that bind the topics associated to the words of a segment. In addition, this model relies on both document and segment specific topic distri-

[1] This sentence is taken from New York Times news (NYT) collection described in Section 4.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1799–1809
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1165

butions so as to capture fine grained differences in topic assignments. A simple switching mechanism is used to select the appropriate distribution (document or segment specific) for assigning a topic to a word. We show that this model naturally encompasses other state-of-the-art LDA-based models proposed to accomplish the same task, and that it outperforms these models over six publicly available collections in terms of perplexity, Normalized Pointwise Mutual Information (NPMI), a measure used to assess the coherence of topics with documents, and the Micro F1-measure in a text classification context.

2 Related work

Probabilistic Latent Semantic Analysis (PLSA) proposed by (Hofmann, 1999) is the first probabilistic model that explains the generation of co-occurrence data using latent random topics and, the EM algorithm for parameter estimation. The model was found more flexible and scalable than the Latent Semantic Analysis (Deerwester et al., 1990), which is based on the singular value decomposition of the document-term matrix, however PLSA is not a generative model as parameter estimation should be performed at each addition of new documents. To overcome this drawback, Blei et al. (2003) proposed the Latent Dirichlet Allocation (LDA) by assuming that the latent topics are random variables sampled from a Dirichlet distribution and that the generated words, occurring in a document, are exchangeable. The interdependence assumption allows the parameter estimation and the inference of the LDA model to be carried out efficiently, but it is not realistic in the sense that topics assigned to similar words of a text span are generally incoherent.

Different studies, presented in the following sections, attempted to remedy this problem and they can be grouped in two broad families depending on whether they make use of external knowledge-based tools or not in order to exhibit text structure for word-topic assignment.

2.1 Knowledge-based topic assignments

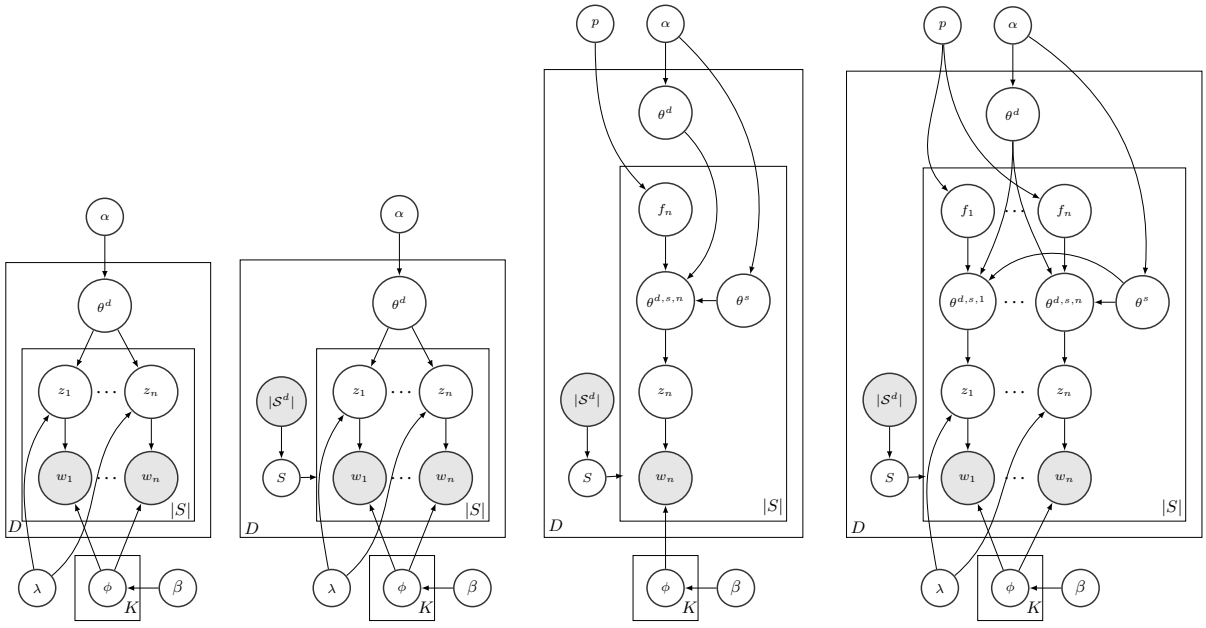
The main assumption behind these models are that text-spans such as sentences, phrases or segments are related in their content. Therefore, the integration of these dependent structures can help to discover coherent latent topics for words. Different attempts to combine LDA-based models with statistical tools to discover document structures have been successfully proposed, such as the study of Griffiths et al. (2005) who investigated the effect of combining a Hidden Markov Model with LDA to capture long and short distance dependencies. Similarly, (Boyd-Graber and Blei, 2008; Balikas et al., 2016a,b) integrated text structure exhibited by a parser or a chunker in their topic models. In this line, Du et al. (2013) following (Du et al., 2010) presented a hierarchical Bayesian model for unsupervised topic segmentation. This model integrates a boundary sampling method used in a Bayesian segmentation model introduced by Purver et al.(2006) to the topic model. For inference, a non-parametric Markov Chain inference is used that splits and merges the segments while a Pitman-Yor process (Teh, 2006) binds the topics. Recently, Tamura and Sumita (2016) extended this idea to the bilingual setting. They assume that documents consist of segments and the topic distribution of each segment is generated using a Pitman-Yor process (Teh, 2006).

Though, the topic assignments follow the structure of the text; these models suffer from the bias of statistical or linguistic tools they rely on. To overpass this limitation, other systems integrated automatically the extraction of text structure, in the form of phrases, in their process.

2.2 Knowledge-free topic assignments

This type of models extract text-spans using n-gram counts and word collections and use bigrams to integrate the order of words as well as to capture the topical content of a phrase (Lau et al., 2013). In (Wang et al., 2007), depending on the topic a particular bigram can be either considered as a single token or as two unigrams. Further, Wang et al. (2009) merged topic models with a unigram model over sentences that assigns topics to the sentences instead of the words.

Our proposed approach also does not make use of external statistical tools to find text segments. The main difference with the previous knowledge-free topic model approaches is that the proposed approach assigns topics to words based on two, segment-specific and document-specific distributions selected from a Bernoulli law. Topics within segments are then constrained using copulas that bind their distributions. In this way, segmentation is embedded in the model and it naturally comes along with the topic assignment.



(a) copLDA (b) segLDAcop$_{p=0}$ (c) segLDAcop$_{\lambda=0}$ (d) segLDAcop

Figure 1: Graphical model for Copula LDA (copLDA), extension of Copula LDA with segmentation (segLDAcop$_{p=0}$), LDA with segmentation and topic shift (segLDAcop$_{\lambda=0}$) and complete model (segLDAcop).

3 Joint latent model for topics and segments

We define here a *segment* as a topically coherent sequence of contiguous words. By topically coherent, we mean that, even though words in a segment can be associated to different topics, these topics are usually related. This view is in line with the one expressed in (Balikas et al., 2016b), in which a latent topic model, referred to as copLDA in the remainder, includes a binding mechanism between topics within coherent text spans, defined in their study as noun phrases (NPs). The relation between topics is captured through a copula that provides a joint probability for all the topics used in a segment. That is, to generate words in a segment, one first jointly generates all the word specific topics z via a copula, and then generates each word in the segment from its word specific topic and the word-topic distribution ϕ. Figure 1(a) illustrates this.

Copulas are particularly useful when modeling dependencies between random variables, as the joint cumulative distribution function (CDF) $F_{X_1,\cdots,X_n}$ of any random vector $\mathbf{X} = (X_1, \cdots, X_n)$ can be written as a function of its marginals, according to Sklar's Theorem (Nelsen, 2006):

$$F_{X_1,\cdots,X_n}(x_1, \cdots, x_p) = C(F_{X_1}(x_1), \cdots, F_{X_n}(x_n))$$

where C is a copula. For latent topic models, as discussed in (Amoualian et al., 2016), Frank's copula is particularly interesting as (a) it is invariant by permutations and associative, as are the words and topics z in each segment due to the exchangeability assumption, and (b) it relies on a single parameter (denoted λ here) that controls the strength of dependence between the variables and is thus easy to implement. In Frank's copula, when the parameter λ approaches 0, the variables are independent of each other, whereas when λ approaches $+\infty$, the variables take the same value. For further details on copulas, we refer the reader to (Nelsen, 2006).

One important problem, however, with copLDA is its reliance on a predefined segmentation. Although the information brought by the segmentation based on NPs helps to improve topic assignment, it may not be flexible enough to capture all the possible segments of a text. It is easy to correct this problem by considering all possible segmentations of a document and by choosing the most appropriate one at the same time that topics are assigned to words. This is illustrated in Figure 1(b), where a segmentation S is chosen from the set $\mathcal{S}^d$ of possible segmentations for a document d, and where each segment in S are generated in turn. We refer to the associated model as segLDAcop$_{p=0}$ for reasons that will become clear later.

Another point to be noted about copLDA (and

segLDAcop$_{p=0}$) is that the topics used in each segment come from the same document specific topic distribution θ^d. This entails that, in these models, one cannot differentiate the main topics of a document from potential segment specific topics that can explain some parts of it. Indeed, some text segments can refer to topics that are barely present in other parts of the document; relying on a single topic distribution may prevent one from capturing those segment specific topics.

It is possible to overcome this difficulty by generating a segment specific topic distribution as illustrated in Figure 1(c) (this model is referred to as segLDAcop$_{\lambda=0}$, again for reasons that will become clear later). However, as some words in a segment can be associated to the general topics of a document, we introduce a mechanism to choose, for each word in a segment, a topic either from the segment specific topic distribution θ^s or from the document specific topic distribution θ^d (this mechanism is similar to the one used for routes and levels in (Paul and Girju, 2010)). The choice between them is based on the Bernoulli variable f, as explained in the generative story given below.

The above developments can be combined in a single, complete model, illustrated in Figure 1(d) and detailed below. We will simply refer to this model as segLDAcop.

3.1 Complete generative model

As in standard LDA based models, with V denoting the size of the vocabulary of the collection and K the number of latent topics, β and ϕ^k, $1 \leq k \leq K$, are V dimensional vectors, α and θ (*i.e.*, $\theta^d, \theta^s, \theta^{d,s,n}$) are K dimensional vectors, whereas z_n takes value in $\{1, \cdots, K\}$. Lower indices are used to denote coordinates of the above vectors. Lastly, Dir denotes the Dirichlet distribution, Cat the categorical distribution (which is a multinomial distribution with one draw) and we omit, as is usual, the generation of the length of the document. The complete model segLDAcop is then based on the following generative process:

1. Generate, for each topic $k, 1 \leq k \leq K$, a distribution over the words: $\phi^k \sim Dir(\beta)$;
2. For each document $d, 1 \leq d \leq D$:
   (a) Choose a document specific topic distribution: $\theta^d \sim Dir(\alpha)$;
   (b) Choose a segmentation S of the document uniformly from the set of all possible segmentations $\mathcal{S}^d$: $P(S) = \frac{1}{|\mathcal{S}^d|}$;
   (c) For each segment s in S:
      (i) Choose a segment specific topic distribution: $\theta^s \sim Dir(\alpha)$;
      (ii) For each position n in s, choose $f_n \sim Ber(p)$ and set:

$$\theta^{d,s,n} = \begin{cases} \theta^s & \text{if } f_n = 1 \\ \theta^d & \text{otherwise} \end{cases}$$

      (iii) Choose topics $\mathcal{Z}^s = \{z_1, \ldots, z_n\}$ from Frank's copula with parameter λ and marginals $Cat(\theta^{d,s,n})$;
      (iv) For each position n in s, choose word w_n: $w_n \sim Cat(\phi^{z_n})$.

As on can note, the generative process relies on a segmentation uniformly chosen from the set of possible segmentations (step *2.b*) to generate related topics within each segment (Frank's copula in step *2.c.(iii)*), the distribution underlying each word specific topic z_n being either specific to the segment or general to the document (steps *2.c.(i)* and *2.c.(ii)*). The other steps are similar to the standard LDA steps.

As in almost all previous studies on LDA, α and β are considered fixed and symmetric, each coordinate of the vector being equal: $\alpha_1 = \cdots = \alpha_K$. The hyperparameters p ($\in [0, 1]$) of the Bernoulli distribution and λ ($\in [0, +\infty]$) of Frank's copula respectively regulate the choice between the segment specific and the document specific topic distributions and the strength of the dependence between topics in a segment. As for the other hyperparameters, we consider them fixed here (the values for all hyperparameters are given in Section 4).

As mentioned before, all the models presented in Figure 1 are special cases of the complete model segLDAcop: hence segLDAcop$_{\lambda=0}$ is obtained by dropping the topic dependencies, which amounts to setting λ to (a value close to) 0, segLDAcop$_{p=0}$ is obtained by relying only on the topic distribution obtained for the document, which amounts to setting p to 0, and the previously introduced copLDA model is obtained by setting p to 0, and fixing the segmentation.

3.2 Inference with Gibbs sampling

The parameters of the complete model can be directly estimated through Gibbs sampling. The Gibbs updates for the parameters ϕ and θ are the same as the ones for standard LDA (Blei et al.,

2003). The parameters f_n are directly estimated through: $f_n \sim Ber(p)$. Lastly, for the variables z, we follow the same strategy as the one described in (Balikas et al., 2016b) and based on (Amoualian et al., 2016), leading to:

$$P(\mathcal{Z}^s|\mathcal{Z}^{-s}, W, \Theta, \Phi, \lambda) = p(\mathcal{Z}^s|\Theta, \lambda) \prod_n \phi_{w_n}^{z_n}$$

where W denotes the document collection, and Θ and Φ the sets of all θ and ϕ^k, $1 \leq k \leq K$, vectors. $p(\mathcal{Z}^s|\Theta, \lambda)$ is obtained by Frank's copula with parameter λ and marginals $Cat(\theta^{d,s,n})$. As is standard in topic models, the notation $-s$ means excluding the information from s.

From the above equation, one can formulate an acceptance/rejection algorithm based on the following steps: (a) sample $\mathcal{Z}^s$ from $p(\mathcal{Z}^s|\Theta, \lambda)$ using Frank's copula, and (b) accept the sample with probability $\prod_n \phi_{w_n}^{z_n}$, where n runs over all the positions in segment s.

3.3 Efficient segmentation

As topics may change from one sentence to another, we assume here that segments cannot overlap sentence boundaries. The different segmentations of a document are thus based on its sentence segmentations. In the remainder, we use L to denote the maximum length of a segment and $g(M; L)$ to denote the number of segmentations in a sentence of length M, each segment comprising at most L words.

Generating all possible segmentations of a sentence and then selecting one at random is not an efficient process as the number of segments rapidly grows with the length of the sentence. In practice, however, one can define an efficient segmentation on the basis of the following proposition, the proof of which is given in Appendix A:

Proposition 3.1. *Let l_i^s be the random variable associated to the length of the segment starting at position i in a sentence of length M (positions go from 1 to M and l_i^s takes value in $\{1, \cdots, L\}$). Then $P(l_i^s = l) := \frac{g(M+1-i-l);L)}{g(M+1-i;L)}$ defines a probability distribution over l_i^s.*

Furthermore, the following process is equivalent to choosing sentence segmentations uniformly from the set of possible segmentations.

```
From pos. 1, repeat till end of sentence:
(a) Generate segment length acc. to P;
(b) Add segment to current segmentation;
(c) Move to position after the segment.
```

In practice, we thus replace steps *2.b* and *2.c* of the generative story by a loop over all sentences, and in each sentence use the process described in Prop, 3.1. Furthermore, as described in Appendix A, the values of g needed to compute $P(l_i^s = l)$ can be efficiently computed by recurrence.

4 Experiments

We conducted a number of experiments aimed at studying the impact of simultaneously segmenting and assigning topics to words within segments using the proposed `segLDAcop` model.

Datasets: We considered six publicly available datasets derived from Pubmed[2] (Tsatsaronis et al., 2015), Wikipedia (Partalas et al., 2015), Reuters[3] and New York Times (NYT)[4] (Yao et al., 2016). The first two collections were considered in (Balikas et al., 2016a), we followed their setup by considering 3 subsets of Wikipedia with different number of classes (namely, Wiki0, Wiki1 and Wiki2). The Reuters dataset comes from Reuters-21578, Distribution 1.0 as investigated in (Bird et al., 2009) and the NYT dataset is collected from full text of New York Times global news, from January 1st to December 31st, 2011.

These collections were processed following (Blei et al., 2003) by removing a standard list of 50 stop words, lemmatizing, lowercasing and keeping only words made of letters. To deal with relatively homogeneous collections, we also removed documents that are too long. The statistics of these datasets, as well as the admissible maximal length for documents, in terms of the number of words they contain, can be found in Table 1.

Settings: We compared our models (`segLDAcop`$_{p=0}$, `segLDAcop`$_{\lambda=0}$, `segLDAcop`) with three models, namely the standard `LDA` model, and two previously introduced models aiming at binding topics within segments:

1. `LDA`: Standard Latent Dirichlet Allocation implemented using collapsed Gibbs sampling inference (Griffiths and Steyvers, 2004)[5]. Note

[2] https://github.com/balikasg/topicModelling/tree/master/data

[3] https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection

[4] https://github.com/yao8839836/COT/tree/master/data

[5] http://gibbslda.sourceforge.net

	Wiki0	Wiki1	Wiki2
# words	32,354	70,954	103,308
– *vocabulary size*	7,853	12,689	14,715
# docs	1,014	2,138	3,152
– *maximal length*	100	100	100
# labels	17	42	53
	Pubmed	Reuters	NYT
# words	104,683	192,562	237,046
– *vocabulary size*	12,779	10,479	17,773
# docs	2,059	6,708	2,564
– *maximal length*	75	50	200
# labels	50	83	-

Table 1: Dataset statistics.

that there are neither segmentation nor topic binding mechanisms in this model;

2. `senLDA`: Sentence LDA, introduced in (Balikas et al., 2016a), which forces all words within a sentence to be assigned to the same topic. The segments considered thus correspond to sentences, and the binding between topics within segments is maximal as all word specific topics are equal;
3. `copLDA`: Copula LDA, introduced in (Balikas et al., 2016b) already discussed before, which relies on two types of segments, namely NPs (extracted with the `nltk.chunk` package (Bird et al., 2009)) and single words. In addition, a copula is also used to bind topics within NPs, from the document specific topic distribution.

Both `senLDA` and `copLDA` implementations, can be found in `https://github.com/balikasg/topicModelling`.

In all models α and β play a symmetric role and are respectively fixed to $1/K$, following (Asuncion et al., 2009). For copula based models, λ is set to 5, following (Balikas et al., 2016b). As already discussed, p is set to 0 for $\texttt{segLDAcop}_{p=0}$; it is set to 0.5 for `segLDAcop` so as not to privilege *a priori* one topic distribution (document or segment specific) over the other. For sampling from Frank's copula, we relied on the R copula package (Hofert and Maechler, 2011) [6]. We chose L (the maximum length of a segment) using line search for $L \in [2, 5]$ and used $L = 3$ in all our experiments. Finally, to illustrate the behaviors of the different models with different number of topics, we present here the results obtained with $K = 20$ and $K = 100$.

We now compare the different models along three main dimensions: perplexity, use of topic representations for classification and topic coherence.

[6] Our complete code will be available for research purposes.

4.1 Perplexity

We first randomly split here all the collections, using 75% of them for training, and 25% for testing.

In order to see how well the models fit the data and following (Blei et al., 2003), we first evaluated the methods in terms of perplexity defined as:

$$Perplexity = \exp\left(\frac{-\sum_{d\in D}\sum_{w\in d}\log\sum_{k=1}^{K}\theta_k^d\phi_w^k}{\sum_{d\in D}|d|}\right),$$

where d is a test document from the test set D, and $|d|$ is the total number of words in d, and K is the total number of topics. The lower the perplexity is, the better the model fits the test data. Table 2 shows perplexities of different methods for $K = 20$ and $K = 100$ topics.

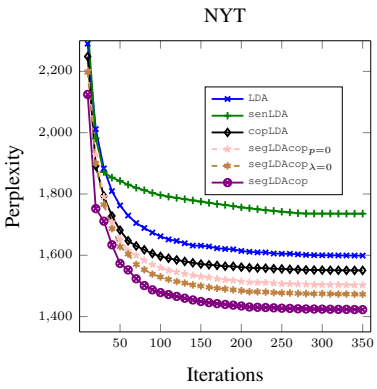


Figure 2: Perplexity with respect to training iteration on NYT collection (20 topics).

From Table 2, it comes out that the best performing model in terms of perplexity over all datasets and for different number of topics is `segLDAcop`. Further, $\texttt{segLDAcop}_{\lambda=0}$, that uses both document and segment specific topic distributions, performs better than $\texttt{segLDAcop}_{p=0}$, which in turn outperforms `copLDA`, bringing evidence that using all possible segmentations rather than only NPs unit extracted using a chunker yields a more flexible and natural topic assignment.

`segLDAcop` also converges faster than the other methods to its minimum as it is shown in Figure 2, depicting the evolution of perplexity of different models over the number of iterations on the NYT collection (a similar behavior is observed on the other collections).

Models	Wiki0		Wiki1		Wiki2		Pubmed		Reuters		NYT	
	20	100	20	100	20	100	20	100	20	100	20	100
LDA	853.7	370.9	1144.6	541.1	1225.2	570.6	1267.8	628.7	210.6	118.8	1600.1	1172.1
senLDA	958.4	420.5	1236.7	675.3	1253.1	625.2	1346.3	674.3	254.3	173.6	1735.9	1215.3
copLDA	753.1	264.3	954.1	411.5	1028.6	420.6	1031.5	483.2	206.3	101.3	1551.5	1063.2
segLDAcop$_{p=0}$	670.2	235.4	904.2	382.4	975.7	409.2	985.5	459.3	194.2	96.7	1504.2	1033.2
segLDAcop$_{\lambda=0}$	655.1	222.1	890.3	370.2	949.2	404.3	971.3	451.2	190.1	91.3	1474.6	1014.3
segLDAcop	**621.2**	**213.5**	**861.2**	**358.6**	**934.7**	**394.4**	**960.4**	**442.1**	**182.1**	**87.5**	**1424.2**	**992.3**

Table 2: Perplexity with respect to different number of topics (20 and 100).

Models	Wiki0		Wiki1		Wiki2		Pubmed		Reuters	
	20	100	20	100	20	100	20	100	20	100
LDA	55.3	63.5	42.4	51.4	41.2	48.7	54.1	63.5	75.5	82.7
senLDA	41.4	53.2	33.5	44.5	36.4	40.9	50.2	62.5	69.4	74.2
copLDA	51.2	62.7	43.4	52.1	40.8	46.5	53.5	63.1	75.2	81.5
segLDAcop$_{p=0}$	59.1	64.2	44.8	51.2	42.3	50.1	55.4	63.1	76.8	82.5
segLDAcop$_{\lambda=0}$	61.1	67.4	46.5	53.8	44.1	52.2	57.1	65.2	79.6	84.4
segLDAcop	**62.3**	**68.4**	**48.4**	**55.2**	**44.8**	**53.5**	**59.3**	**66.5**	**80.2**	**85.1**

Table 3: MiF score (percent) with respect to different number of topics (20 and 100).

4.2 Topical induced representation for classification

Some studies compare topic models using extrinsic tasks such as document classification. In this case, it is possible to reduce the dimensionality of the representation space by using the induced topics (Blei et al., 2003). In this study, we first randomly splitted the datasets, except NYT that does not contain class information, into training (75%) and test (25%) sets. We then applied SVMs with a linear kernel; the value of the hyperparameter C was found by cross-validation over the training set $\{0.01, 0.1, 1, 10, 100\}$. For datasets where certain documents have more than one label (Pubmed, Reuters), we used the one-versus-all approach for performing multi-label classification.

In Table 3, we report the Micro F1 (MiF) score of different models on the test sets. Again, the best results are obtained with segLDAcop, followed by segLDAcop$_{\lambda=0}$. This shows the importance of relying on both document and segment specific topic distributions. As conjectured before, our model is able to captures fine grained topic assignments within documents. In addition, all models relying on an inferred segmentation (segLDAcop$_{p=0}$, segLDAcop$_{\lambda=0}$, segLDAcop) outperform the models relying on fixed segmentations (sentences or NPs). This shows the importance of being able to discover flexible segmentations for assigning topics within documents.

4.3 Topic coherence

Another common way to evaluate topic models is by examining how coherent the produced topics are. Doing this manually is a time consuming process and cannot scale. To overcome this limitation the task of automatically evaluating the coherence of topics produced by topic models received a lot of attention (Mimno et al., 2011). It has been found that scoring the topics using co-occurrence measures, such as the pointwise mutual information (PMI) between the top-words of a topic, correlates well with human judgments (Newman et al., 2010). For this purpose an external, large corpus is used as a meta-document where the PMI scores of pairs of words are estimated using a sliding window. As discussed above, calculating the co-occurrence measures requires selecting the top-N words of a topic and performing the manual or automatic evaluation. Hence, N is a hyper-parameter to be chosen and its value can impact the results. Very recently, Lau and Baldwin (2016) showed that N actually impacts the quality of the obtained results and, in particular, the correlation with human judgments. In their work, they found that aggregating the topic coherence scores over several topic cardinalities leads to a substantially more stable and robust evaluation.

Following the findings of Lau and Baldwin (2016) and using (Newman et al., 2010)'s equation, we present in Figure 3 the topic coherence scores as measured by the Normalized Pointwise Mutual Information (NPMI) . Their values are in [-1,1], where in the limit of -1 two words w_1 and w_2 never occur together, while in the limit of +1 they always occur together (complete co-occurrence). For the reported scores, we aggregate the topic coherence scores over three different topic cardinalities: $N \in \{5, 10, 15\}$. segLDAcop model which

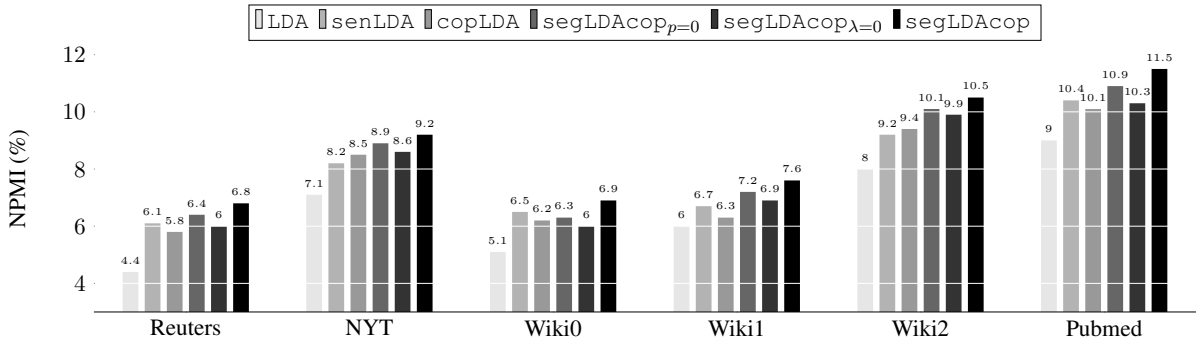


Figure 3: Topic coherence (NPMI) score with respect to 100 of topics.

uses copulas and segmentation together, shows the best score for the given reference meta-data (Wikipedia) in all of the datasets. It should be noted that segLDAcop$_{\lambda=0}$ which has not copula binder inside the model has less improvement against the segLDAcop$_{p=0}$ which has the copula. This means using copula has more effect on the topic coherence than only the segment-specific topic distribution.

4.4 Visualization

In order to illustrate the results obtained by segLDAcop, we display in Figure 4 the top 10 most probable words over 5 topics ($K = 20$) for the Reuters dataset, for both segLDAcop and LDA. In segLDAcop, topic 1, the top-ranked words are mostly relevant to the topic "date" (*e.g.*, march, january, year, fall, february, week). However, a similar topic learned by LDA appears to involve less such words (year, january, february), indicating a less coherent topic.

Topic1	march, fell, rose, january, rise, year, fall, february, pct, week	fell, mln, year, january, dlrs, rise, rose, pct, billion, february
Topic2	currency, bank, pct, cut, rate, day, prime, exchange, interest, national	billion, prime, day, rate, dlrs, pct, reserve, federal, fed, bank
Topic3	term, agreement, acquire, buy, sell, unit, acquisition, corp, company, sale	term, dlrs, buy, company, sell, unit, corp, acquisition, sale, mln
Topic4	approved, american, common, split, merger, company, board, stock, share, shareholder	acquire, mln, company, common, stock, shareholder, share, corp, merger, dlrs
Topic5	tokyo, life, intent, letter, buy, insurance, yen, japan, dealer, dollar	central, european, japan, yen, ec, dollar, bank, rate, dealer, market

Figure 4: Top-10 words of segLDAcop (left) vs LDA (right) for the Reuters (5 out of 20 topics).

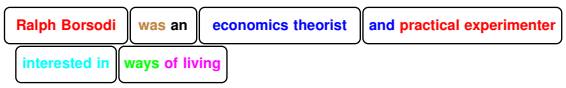


Figure 5: Topic assignments with segmentation boundaries using segLDAcop. Colors are topics (examples from Wiki0 including stopwords with 20 topics).

Figure 5 illustrates another aspect of our model, namely the possibility to detect topically coherent segments. In particular, as one can note, the sentence is segmented in six parts by our model, the first one is a NP, *Ralph Borsodi* where one single topic is assigned to both words. We observe a similar coherence in topic assignments on other NPs and segments, in which a single topic is used for the words involved. The data-driven approach we have adopted here can discover such fine grained differences, something the approaches based on fixed segmentations (either based on sentences or NPs), are less likely to achieve.

5 Discussion

In this paper, we have introduced an LDA-based model that generates topically coherent segments within documents by jointly segmenting documents and assigning topics to their words. The coherence between topics is ensured through Frank's copula, that binds the topics associated to the words of a segment. In addition, this model relies on both document and segment specific topic distributions so as to capture fine grained differences in topic assignments. We have shown that this model naturally encompasses other state-of-the-art LDA-based models proposed to accomplish the same task, and that it outperforms these models over six publicly available collections in terms of perplexity, Normalized Pointwise Mutual Information (NPMI), a measure used to assess the coherence of topics with documents, and the Micro F1-measure in a text classification context. Our results confirm the importance of a flexible segmentation as well as a

binding mechanism to produce topically coherent segments.

As regards complexity, it is true that more complex models, as the one we are considering, are more prone to underfitting (when data is scarce) and overfitting than simpler models. This said, the experimental results on perplexity (in which the word-topic distributions are fixed) and on classification (based on the topical induced representations) suggest that our model neither underfits nor overfits compared to simpler models. We believe that this is due to the fact that the main additional parameters in our model (the segment specific topic distribution) do not really add complexity as they are drawn from the same distribution as the standard document specific topics. Furthermore, the parameters p and f are simple parameters to choose between these two distributions.

The comparison with other segmentation methods is also an important point. While state-of-the-art supervised segmentation models can be used before applying the `LDA` model, we note such a pipeline approach comes with several limitations. The approach requires external annotated data to train the segmentation models, where certain domain and language specific information need to be captured. By contrast, our unsupervised approach learns both segmentations and topics jointly in a domain and language independent manner. Furthermore, existing supervised segmentation models are largely designed for a very different purpose with strong linguistic motivations, which may not align well with our main goal in this paper which is improving topic coherence in topic modeling. Similarly, unsupervised approaches, used for example in the TDT (Topic Detection and Tracking) campaigns or more recently in Du et al. (2013), usually consider coarse-grained topics, that can encompass several sentences. In contrast, our approach aims at identifying fine-grained topics associated with coherent segments that do not overlap sentence boundaries. These considerations, explain the choice of the baselines retained: they are based on segments of different granularities (words, NPs, sentences) that do not overlap sentence boundaries.

In the future, we plan on relying on other inference approaches, based for example on variational Bayes known to yield better estimates for perplexity (Asuncion et al., 2009); it is however not certain that the gain in perplexity one can expect from the use of variational Bayes approaches will necessarily result in a gain in, say, topic coherence. Indeed, the impact of the inference approach on the different usages of latent topic models for text collections remains to be better understood.

Acknowledgments

We would like to thank the reviewers for their helpful comments. Most of this work was done when Hesam Amoualian was visiting Singapore University of Technology and Design. This work is supported by MOE Tier 1 grant SUTDT12015008, also partly supported by the LabEx PERSYVAL-Lab ANR-11-LABX-0025.

References

Hesam Amoualian, Marianne Clausel, Eric Gaussier, and Massih-Reza Amini. 2016. Streaming-lda: A copula-based approach to modeling topic dependencies in document streams. In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, SIGKDD, pages 695–704. https://doi.org/10.1145/2939672.2939781.

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2009. On smoothing and inference for topic models. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, Arlington, Virginia, United States, UAI, pages 27–34. http://dl.acm.org/citation.cfm?id=1795114.1795118.

Georgios Balikas, Massih-Reza Amini, and Marianne Clausel. 2016a. On a topic model for sentences. In *Proceedings of the 39th International Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR, pages 921–924. https://doi.org/10.1145/2911451.2914714.

Georgios Balikas, Hesam Amoualian, Marianne Clausel, Eric Gaussier, and Massih R Amini. 2016b. Modeling topic dependencies in semantically coherent text spans with copulas. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, COLING, pages 1767–1776. http://aclweb.org/anthology/C16-1166.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly, Beijing. http://www.nltk.org/book/.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning* 3:993–1022. http://dl.acm.org/citation.cfm?id=944919.944937.

Jordan Boyd-Graber and David Blei. 2008. Syntactic topic models. In *Proceedings of the*

21st International Conference on Neural Information Processing Systems. Curran Associates Inc., USA, NIPS, pages 185–192. http://dl.acm.org/citation.cfm?id=2981780.2981804.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41(6):391–407. http://dx.doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.

Lan Du, Wray Buntine, and Huidong Jin. 2010. A Segmented Topic Model Based on the Two-parameter Poisson-Dirichlet Process. *Journal of Machine learning* 81(1):5–19. https://doi.org/10.1007/s10994-010-5197-4.

Lan Du, Wray Buntine, and Mark Johnson. 2013. Topic Segmentation with a Structured Topic Model. In *Proceedings of The Annual Conference of the North American Chapter of the Association for Computational Linguistics, Human Language Technologies*. HLT-NAACL, pages 190–200. http://dblp.uni-trier.de/db/conf/naacl/naacl2013.html/DuBJ13.

Gal Elidan. 2013. *Copulas in Machine Learning*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 39–60. https://doi.org/10.1007/978-3-642-35407-6_3.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Journal of the National Academy of Sciences* 101(suppl 1):5228–5235. https://doi.org/10.1073/pnas.0307752101.

Thomas L Griffiths, Mark Steyvers, David M Blei, and Joshua B Tenenbaum. 2005. Integrating topics and syntax. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in the International Conference on Neural Information Processing Systems*. MIT Press, NIPS, pages 537–544. http://papers.nips.cc/paper/2587-integrating-topics-and-syntax.pdf.

Marius Hofert and Martin Maechler. 2011. Nested Archimedean Copulas Meet R: The nacopula Package. *Journal of Statistical Software* 39(i09):–. https://doi.org/http://hdl.handle.net/10.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR, pages 50–57. https://doi.org/10.1145/312624.312649.

Jey Han Lau and Timothy Baldwin. 2016. The sensitivity of topic coherence evaluation to topic cardinality. In *Proceedings of The Annual Conference of the North American Chapter of the Association for Computational Linguistics, Human Language Technologies, San Diego California, USA, June 12-17, 2016*. NAACL, pages 483–487. http://aclweb.org/anthology/N/N16/N16-1057.pdf.

Jey Han Lau, Timothy Baldwin, and David Newman. 2013. On collocations and topic models. *Journal of ACM Trans. Speech Lang. Process.* 10(3):10:1–10:14. https://doi.org/10.1145/2483969.2483972.

David Mimno and Andrew McCallum. 2007. Organizing the oca: Learning faceted subjects from a library of digital books. In *Proceedings of the 7th Joint Conference on Digital Libraries*. ACM, New York, NY, USA, JCDL '07, pages 376–385. https://doi.org/10.1145/1255175.1255249.

David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP, pages 262–272. http://dl.acm.org/citation.cfm?id=2145432.2145462.

Roger B. Nelsen. 2006. *An Introduction to Copulas (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. http://www.springer.com/gp/book/9780387286594.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Proceedings of The Annual Conference of the North American Chapter of the Association for Computational Linguistics, Human Language Technologies*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL, pages 100–108. http://dl.acm.org/citation.cfm?id=1857999.1858011.

Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, et al. 2015. LSHTC: A Benchmark for Large-Scale Text Classification. *Journal of CoRR* abs/1503.08581. http://arxiv.org/abs/1503.08581.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Proceedings of the 24th Conference on Artificial Intelligence*. AAAI Press, AAAI, pages 545–550. http://dl.acm.org/citation.cfm?id=2898607.2898695.

Matthew Purver, Thomas L Griffiths, Konrad P. Körding, and Joshua B. Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL, pages 17–24. https://doi.org/10.3115/1220175.1220178.

Akihiro Tamura and Eiichiro Sumita. 2016. Bilingual segmented topic model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. ACL. http://aclweb.org/anthology/P/P16/P16-1120.pdf.

Yee Whye Teh. 2006. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL, pages 985–992. https://doi.org/10.3115/1220175.1220299.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, et al. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *Journal of BMC Bioinformatics* 16(1):138. https://doi.org/10.1186/s12859-015-0564-6.

Dingding Wang, Shenghuo Zhu, Tao Li, and Yihong Gong. 2009. Multi-document summarization using sentence-based topic models. In *Proceedings of the Conference on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL-IJCNLP, pages 297–300. http://dl.acm.org/citation.cfm?id=1667583.1667675.

Xuerui Wang, Andrew McCallum, and Xing Wei. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Proceedings of the 7th International Conference on Data Mining*. IEEE Computer Society, Washington, DC, USA, ICDM, pages 697–702. https://doi.org/10.1109/ICDM.2007.86.

Xing Wei and W. Bruce Croft. 2006. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR, pages 178–185. https://doi.org/10.1145/1148170.1148204.

Liang Yao, Yin Zhang, Baogang Wei, Lei Li, Fei Wu, Peng Zhang, and Yali Bian. 2016. Concept over time: the combination of probabilistic topic model with wikipedia knowledge. *Journal of Expert Systems with Applications* 60:27 – 38. https://doi.org/10.1016/j.eswa.2016.04.014.

Hua-Jun Zeng, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. 2004. Learning to cluster web search results. In *Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR, pages 210–217. https://doi.org/10.1145/1008992.1009030.

A Efficient segmentation

Let us recall the property presented before:

Proposition A.1. *Let l_i^s be the random variable associated to the length of the segment starting at position i in a sentence of length M (positions go from 1 to M and l_i^s takes value in $\{1, \cdots, L\}$). Then $P(l_i^s = l) := \frac{g(M+1-i-l);L)}{g(M+1-i;L)}$ defines a probability distribution over l_i^s.*

Furthermore, the following process is equivalent to choosing sentence segmentations uniformly from the set of possible segmentations.

```
From pos. 1, repeat till end of sentence:
(a) Generate segment length acc. to P;
(b) Add segment to current segmentation;
(c) Move to position after the segment.
```

Proof Any segmentation of the sentence of length M starts with either a segment of length 1, a segment of length 2, $\cdots$, or a segment of length L. Thus, $g(M; L)$ can be defined through the following recurrence relation:

$$g(M; L) = \sum_{l=1}^{L} g(M - l; L) \tag{1}$$

together with the initial values $g(1; L), g(2; L), \cdots, g(L; L)$, which can be computed offline (for example, for $L = 3$, one has: $g(1; 3) = 1, g(2; 3) = 2, g(3; 3) = 4$). Note that $g(1; L) = 1$ for all L.

Thus:

$$\sum_{l=1}^{L} P(l_i^s = l) = \sum_{l=1}^{L} \frac{g(M + 1 - i - l); L)}{g(M + 1 - i; L)} = 1$$

due to the recurrence relation on g. This proves the first part of the proposition.

Using the process described above where segments are generated one after another according to P, for a segmentation S, comprising $|S|$ segments, let us denote by $l_1, l_2, \cdots, l_{|S|}$ the lengths of each segment and by $i_1, i_2, \cdots, i_{|S|}$ the starting positions of each segment (with $i_1 = 1$). One has, as segments are independent of each other:

$$P(S) = \prod_{j=1}^{|S|} P(l_{i_j}^s = l_j) = \prod_{j=1}^{|S|} \frac{g(M + 1 - (i_j + l_j); L)}{g(M + 1 - i_j; L)}$$
$$= \frac{g(M - l_1; L)}{g(M; L)} \frac{g(M - l_1 - l_2; L)}{g(M - l_1; L)} \cdots = \frac{1}{g(M; L)}$$

as $g(1; L) = 1$. This concludes the proof of the proposition. □

Furthermore, as one can note from Eq. 1, the various elements needed to compute $P(l_i^s = l)$ can be efficiently computed, the time complexity being equal to $O(M)$. In addition, as the number of different sentence lengths is limited, one can store the values of g to reuse them during the segmentation phase.

Jointly Extracting Relations with Class Ties via Effective Deep Ranking

Hai Ye[1], **Wenhan Chao**[1], **Zhunchen Luo**[2*], **Zhoujun Li**[1]
[1]School of Computer Science and Engineering, Beihang University, Beijing 100191, China
{yehai, chaowenhan, lizj}@buaa.edu.cn
[2]China Defense Science and Technology Information Center, Beijing 100142, China
zhunchenluo@gmail.com

Abstract

Connections between relations in relation extraction, which we call *class ties*, are common. In distantly supervised scenario, one entity tuple may have multiple relation facts. Exploiting class ties between relations of one entity tuple will be promising for distantly supervised relation extraction. However, previous models are not effective or ignore to model this property. In this work, to effectively leverage class ties, we propose to make joint relation extraction with a unified model that integrates convolutional neural network (CNN) with a general pairwise ranking framework, in which three novel ranking loss functions are introduced. Additionally, an effective method is presented to relieve the severe class imbalance problem from NR (not relation) for model training. Experiments on a widely used dataset show that leveraging class ties will enhance extraction and demonstrate the effectiveness of our model to learn class ties. Our model outperforms the baselines significantly, achieving state-of-the-art performance.

1 Introduction

Relation extraction (RE) aims to classify the relations between two given named entities from natural-language text. Supervised machine learning methods require numerous labeled data to work well. With the rapid growth of volume of relation types, traditional methods can not keep up with the step for the limitation of labeled data. In order to narrow down the gap of data sparsity, Mintz et al. (2009) propose *distant supervision (DS)* for relation extraction, which automatically generates training data by aligning a knowledge facts database (ie. Freebase (Bollacker et al., 2008)) with texts.

place_lived (Patsy Ramsey, Atlanta)
place_of_birth (Patsy Ramsey, Atlanta)

	Sentence	Latent Label
#1	*Patsy Ramsey* has been living in *Atlanta* since she was born.	*place_of_birth*
#2	*Patsy Ramsy* always loves *Atlanta* since it is her hometown.	*place_lived*

Table 1: Training instances generated by freebase.

Class ties mean the connections between relations in relation extraction. In general, we conclude that class ties can have two types: weak class ties and strong class ties. Weak class ties mainly involve the co-occurrence of relations such as *place_of_birth* and *place_lived*, *CEO_of* and *founder_of*. On the contrary, strong class ties mean that relations have latent logical entailments. Take the two relations of *capital_of* and *city_of* for example, if one entity tuple has the relation of *capital_of*, it must express the relation fact of *city_of*, because the two relations have the entailment of *capital_of* $\Rightarrow$ *city_of*. Obviously the opposite induction is not correct. Further take the sentence of *"Jonbenet told me that her mother [Patsy Ramsey]*$_{e_1}$ *never left [Atlanta]*$_{e_2}$ *since she was born."* in DS scenario for example. This sentence expresses two relation facts which are *place_of_birth* and *place_lived*. However, the word "born" is a strong bios to extract *place_of_birth*, so it may not be easy to predict the relation of *place_lived*, but if we can incorporate the weak ties between the two relations, extracting *place_of_birth* will provide evidence for prediction of *place_lived*.

Exploiting class ties is necessary for DS based relation extraction. In DS scenario, there is a challenge that one entity tuple can have multiple rela-

* Corresponding author.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1810–1820
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1166

tion facts as shown in Table 1, which is called *relation overlapping* (Hoffmann et al., 2011; Surdeanu et al., 2012). However, the relations of one entity tuple can have class ties mentioned above which can be leveraged to enhance relation extraction for it narrowing down potential searching spaces and reducing uncertainties between relations when predicting unknown relations. If one pair entities has *CEO_of*, it will contain *founder_of* with high possibility.

To exploit class ties between relations, we propose to make joint extraction for all positive labels of one entity tuple with considering *pairwise* connections between positive and negative labels inspired by (Fürnkranz et al., 2008; Zhang and Zhou, 2006). As the two relations with class ties shown in Table 1, by joint extraction of two relations, we can maintain the *class ties* (co-occurrence) of them from training samples to be learned by potential model, and then leverage this learned information to extract instances with unknown relations, which can not be achieved by separated extraction for it dividing labels apart losing information of co-occurrence. To classify positive labels from negative ones, we adopt pairwise ranking to rank positive ones higher than negative ones, exploiting pairwise connections between them. In a word, joint extraction exploits class ties between relations and pairwise ranking classify positive labels from negative ones. Furthermore, combining information across sentences will be more appropriate for joint extraction which provides more information from other sentences to extract each relation (Zheng et al., 2016; Lin et al., 2016). In Table 1, sentence #1 is the evidence for *place_of_birth*, but it also expresses the meaning of "living in someplace", so it can be aggregated with sentence #2 to extract *place_lived*. Meanwhile, the word of "hometown" in sentence #2 can provide evidence for *place_of_birth* which should be combined with sentence #1 to extract *place_of_birth*.

In this work, we propose a unified model that integrates pairwise ranking with CNN to exploit class ties. Inspired by the effectiveness of deep learning for modeling sentences (LeCun et al., 2015), we use CNN to encode sentences. Similar to (Santos et al., 2015; Lin et al., 2016), we use class embeddings to represent relation classes. The whole model architecture is presented in Figure 1. We first use CNN to embed sentences, then we introduce two variant methods to combine the

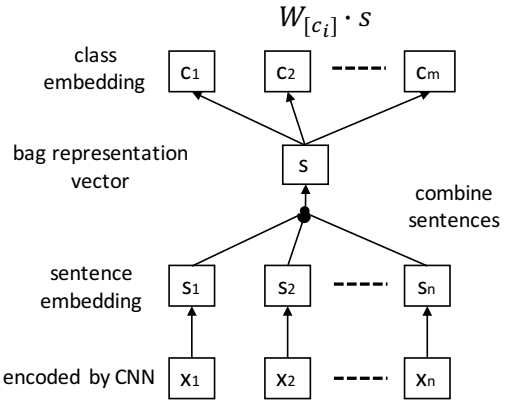


Figure 1: The main architecture of our model.

embedded sentences into one bag representation vector aiming to aggregate information across sentences, after that we measure the similarity between bag representation and relation class in real-valued space. With two variants for combining sentences, three novel pairwise ranking loss functions are proposed to make joint extraction. Besides, to relieve the bad impact of class imbalance from NR (not relation) (Japkowicz and Stephen, 2002) for training our model, we cut down loss propagation from NR class during training.

Our experimental results on dataset of Riedel et al. (2010) are evident that: (1) Our model is much more effective than the baselines; (2) Leveraging class ties will enhance relation extraction and our model is efficient to learn class ties by joint extraction; (3) A much better model can be trained after relieving class imbalance from NR.

Our contributions in this paper can be encapsulated as follows:

- We propose to leverage class ties to enhance relation extraction. An effective deep ranking model which integrates CNN and pairwise ranking framework is introduced to exploit class ties.
- We propose an effective method to relieve the impact of data imbalance from NR for model training.
- Our method achieves state-of-the-art performance.

2 Related Work

We summarize related works on two main aspects:

2.1 Distant Supervision Relation Extraction

Previous works on DS based RE ignore or are not effective to leverage class ties between rela-

tions.

Riedel et al. (2010) introduce multi-instance learning to relieve the wrong labelling problem, ignoring class ties. Afterwards, Hoffmann et al. (2011) and Surdeanu et al. (2012) model this problem by multi-instance multi-label learning to extract overlapping relations. Though they also propose to make joint extraction of relations, they only use information from single sentence losing information from other sentences. Han and Sun (2016) try to use *Markov logic* model to capture consistency between relation labels, on the contrary, our model leverages deep ranking to learn class ties automatically.

With the remarkable success of deep learning in CV and NLP (LeCun et al., 2015), deep learning has been applied to relation extraction (Zeng et al., 2014, 2015; Santos et al., 2015; Lin et al., 2016), the specific deep learning architecture can be CNN (Zeng et al., 2014), RNN (Zhou et al., 2016), etc. Zeng et al. (2015) propose a piecewise convolutional neural network with multi-instance learning for DS based relation extraction, which improves the precision and recall significantly. Afterwards, Lin et al. (2016) introduce the mechanism of attention (Luong et al., 2015; Bahdanau et al., 2014) to select the sentences to relieve the wrong labelling problem and use all the information across sentences. However, the two deep learning based models only make separated extraction thus can not model class ties between relations.

2.2 Deep Learning to Rank

Deep learning to rank has been widely used in many problems to serve as a classification model. In image retrieval, Zhao et al. (2015) apply deep semantic ranking for multi-label image retrieval. In text matching, Severyn and Moschitti (2015) adopt learning to rank combined with deep CNN for short text pairs matching. In traditional supervised relation extraction, Santos et al. (2015) design a pairwise loss function based on CNN for single label relation extraction. Based on the advantage of deep learning to rank, we propose pairwise learning to rank (LTR) (Liu, 2009) combined with CNN in our model aiming to jointly extract multiple relations.

3 Proposed Model

In this section, we first conclude the notations used in this paper, then we introduce the used CNN for sentence embedding, afterwards, we present our algorithm of how to learn class ties between relations of one entity tuple.

3.1 Notation

We define the relation classes as $\mathcal{L} = \{1, 2, \cdots, C\}$, entity tuples as $\mathcal{T} = \{t_i\}_{i=1}^{M}$ and mentions[1] as $\mathcal{X} = \{x_i\}_{i=1}^{N}$. Dataset is constructed as follows: for entity tuple $t_i \in \mathcal{T}$ and its relation class set $L_i \subseteq \mathcal{L}$, we collect all the mentions X_i that contain t_i, the dataset we use is $\mathcal{D} = \{(t_i, L_i, X_i)\}_{i=1}^{H}$. Given a data $(t_k, L_k, X_k) \in \{(t_i, L_i, X_i)\}_{i=1}^{H}$, the sentence embeddings of X_k encoded by CNN are defined as $S_k = \{s_i\}_{i=1}^{|X_k|}$ and we use class embeddings $W \in R^{|\mathcal{L}| \times d}$ to represent the relation classes.

3.2 CNN for Sentence Embedding

We take the effective CNN architecture adopted from (Zeng et al., 2015; Lin et al., 2016) to encode sentence and we briefly introduce CNN in this section. More details of our CNN can be obtained from previous work.

3.2.1 Words Representations

• **Word Embedding** Given a word embedding matrix $V \in \mathbb{R}^{l^w \times d^1}$ where l^w is the size of word dictionary and d^1 is the dimension of word embedding, the words of a mention $x = \{w_1, w_2, \cdots, w_n\}$ will be represented by real-valued vectors from V.

• **Position Embedding** The position embedding of a word measures the distance from the word to entities in a mention. We add position embeddings into words representations by appending position embedding to word embedding for every word. Given a position embedding matrix $P \in \mathbb{R}^{l^p \times d^2}$ where l^p is the number of distances and d^2 is the dimension of position embeddings, the dimension of words representations becomes $d^w = d^1 + d^2 \times 2$.

3.2.2 Convolution, Piecewise max-pooling

After transforming words in x to real-valued vectors, we get the sentence $q \in \mathbb{R}^{n \times d^w}$. The set of kernels K is $\{K_i\}_{i=1}^{d^s}$ where d^s is the number of kernels. Define the window size as d^{win} and given one kernel $K_k \in \mathbb{R}^{d^{win} \times d^w}$, the convolution operation is defined as follows:

$$m_{[i]} = q_{[i:i+d^{win}-1]} \odot K_k + b_{[k]} \quad (1)$$

[1]The sentence containing one certain entity is called mention.

where m is the vector after conducting convolution along q for $n - d^{win} + 1$ times and $b \in \mathbb{R}^{d^s}$ is the bias vector. For these vectors whose indexes out of range of $[1, n]$, we replace them with zero vectors.

By piecewise max-pooling, when pooling, the sentence is divided into three parts: $m_{[p_0:p_1]}$, $m_{[p_1:p_2]}$ and $m_{[p_2:p_3]}$ (p_1 and p_2 are the positions of entities, p_0 is the beginning of sentence and p_3 is the end of sentence). This piecewise max-pooling is defined as follows:

$$z_{[j]} = max(m_{[p_{j-1}:p_j]}) \tag{2}$$

where $z \in \mathbb{R}^3$ is the result of mention x processed by kernel K_k; $1 \leq j \leq 3$. Given the set of kernels K, following the above steps, the mention x can be embedded to o where $o \in R^{d^s * 3}$.

3.2.3 Non-Linear Layer, Regularization

To learn high-level features of mentions, we apply a non-linear layer after pooling layer. After that, a dropout layer is applied to prevent overfitting. We define the final fixed sentence representation as $s \in \mathbb{R}^{d^f}$ ($d^f = d^s * 3$).

$$s = g(o) \circ h \tag{3}$$

where $g(\cdot)$ is a non-linear function and we use $tanh(\cdot)$ in this paper; h is a Bernoulli random vector with probability p to be 1.

3.3 Learning Class Ties by Joint Extraction with Pairwise Ranking

As mentioned above, to learn class ties, we propose to make joint extraction with considering pairwise connections between positive labels and negative ones. Pairwise ranking is applied to achieve this goal. Besides, combining information across sentences is necessary for joint extraction. More specifically, as shown in Figure 2, from down to top, all information from sentences is pre-propagated to provide enough information for joint extraction. From top to down, pairwise ranking jointly extracting positive relations by combining losses, which are back-propagated to CNN to learn class ties.

3.3.1 Combining Information across Sentences

We propose two options to combine sentences to provide enough information for joint extraction.

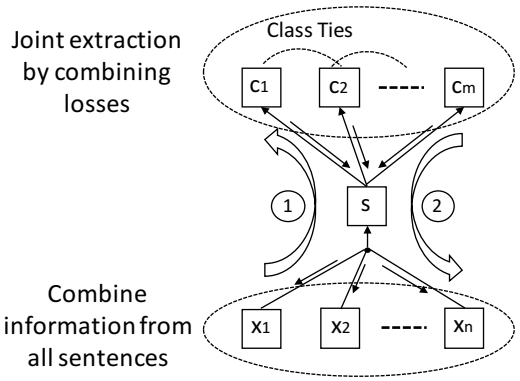


Figure 2: Illustration of mechanism of our model to model class ties between relations.

• **AVE** The first option is average method. This method regards all the sentences equally and directly average the values in all dimensions of sentence embedding. This **AVE** function is defined as follows:

$$s = \frac{1}{n} \sum_{s_i \in S_k} s_i \tag{4}$$

where n is the number of sentences and s is the representation vector combining all sentence embeddings. Because it weights the importance of sentences equally, this method may bring much noise data from two aspects: (1) the wrong labelling data; (2) irrelated mentions for one relation class, for all sentences containing the same entity tuple being combined together to construct the bag representation.

• **ATT** The second one is a sentence-level attention algorithm used by Lin et al. (2016) to measure the importance of sentences aiming to relieve the wrong labelling problem. For every sentence, **ATT** will calculate a weight by comparing the sentence to one relation. We first calculate the similarity between one sentence embedding and relation class as follows:

$$e_j = a \cdot W_{[c]} \cdot s_j \tag{5}$$

where e_j is the similarity between sentence embedding s_j and relation class c and a is a bias factor. In this paper, we set a as 0.5. Then we apply $\mathrm{Softmax}$ to rescale $\mathbf{e}$ ($\mathbf{e} = \{e_i\}_{i=1}^{|X_k|}$) to $[0, 1]$. We get the weight α_j for s_j as follows:

$$\alpha_j = \frac{\exp(e_j)}{\sum_{e_i \in \mathbf{e}} \exp(e_i)} \tag{6}$$

so the function to merge s with **ATT** is as follows:

$$s = \sum_{i=1}^{|X_k|} \alpha_i \cdot s_i \tag{7}$$

3.3.2 Joint Extraction by Combining Losses to Learn Class Ties

Firstly, we have to present the score function to measure the similarity between s and relation c.

• **Score Function** We use dot function to produce score for s to be predicted as relation c. The score function is as follows:

$$\mathcal{F}(s, c) = W_{[c]} \cdot s \tag{8}$$

There are other options for score function. In Wang et al. (2016), they propose a margin based loss function that measures the similarity between s and $W_{[c]}$ by distance. Because score function is not an important issue in our model, we adopt dot function, also used by Santos et al. (2015) and Lin et al. (2016), as our score function.

Now we start to introduce the ranking loss function.

Pairwise ranking aims to learn the score function $\mathcal{F}(s, c)$ that ranks positive classes higher than negative ones. This goal can be summarized as follows:

$$\forall c^+ \in L_k, \forall c^- \in \mathcal{L} - L_k : \mathcal{F}(s, c^+) > \mathcal{F}(s, c^-) + \beta \tag{9}$$

where β is a margin factor which controls the minimum margin between the positive scores and negative scores.

To learn class ties between relations, we extend the formula (9) to make joint extraction and we propose three ranking loss functions with variants of combining sentences. Followings are the proposed loss functions:

• **with AVE (Variant-1)** We define the margin-based loss function with option of **AVE** to aggregate sentences as follows:

$$G_{[ave]} = \sum_{c^+ \in L_k} \rho[0, \sigma^+ - \mathcal{F}(s, c^+)]_+ + \rho|L_k|[0, \sigma^- + \mathcal{F}(s, c^-)]_+ \tag{10}$$

where $[0, \cdot\,]_+ = max(0, \cdot\,)$; ρ is the rescale factor, σ^+ is positive margin and σ^- is negative margin. Similar to Santos et al. (2015) and Wang et al. (2016), this loss function is designed to rank positive classes higher than negative ones controlled by the margin of $\sigma^+ - \sigma^-$. In reality, $\mathcal{F}(s, c^+)$ will be higher than σ^+ and $\mathcal{F}(s, c^-)$ will be lower than σ^-. In our work, we set ρ as 2, σ^+ as 2.5 and σ^- as 0.5 adopted from Santos et al. (2015).

Similar to Weston et al. (2011) and Santos et al. (2015), we update one negative class at every training round but to balance the loss between positive classes and negative ones, we multiply $|L_k|$ before the right term in function (10) to expand the negative loss. We apply mini-bach based stochastic gradient descent (SGD) to minimize the loss function. The negative class is chosen as the one with highest score among all negative classes (Santos et al., 2015), i.e.:

$$c^- = \underset{c \in \mathcal{L} - L_k}{\operatorname{argmax}} \mathcal{F}(s, c) \tag{11}$$

• **with ATT (Variant-2)** Now we define the loss function for the option of **ATT** to combine sentences as follows:

$$G_{[att]} = \sum_{c^+ \in L_k} (\rho[0, \sigma^+ - \mathcal{F}(s^{c^+}, c^+)]_+ + \rho[0, \sigma^- + \mathcal{F}(s^{c^+}, c^-)]_+) \tag{12}$$

where s^c means the attention weights of representation s are merged by comparing sentence embeddings with relation class c and c^- is chosen by the following function:

$$c^- = \underset{c \in \mathcal{L} - L_k}{\operatorname{argmax}} \mathcal{F}(s^{c^+}, c) \tag{13}$$

which means we update one negative class in every training round. We keep the values of ρ, σ^+ and σ^- same as values in function (10).

According to this loss function, we can see that: for each class $c^+ \in L_k$, it will capture the most related information from sentences to merge s^{c^+}, then rank $\mathcal{F}(s^{c^+}, c^+)$ higher than all negative scores which each is $\mathcal{F}(s^{c^+}, c^-)$ ($c^- \in \mathcal{L} - L_k$). We use the same update algorithm to minimize this loss.

• **Extended with ATT (Variant-3)** According to function (12), for each c^+, we only select one negative class to update the parameters, which only considers the connections between positive classes and negative ones, ignoring connections between positive classes, so we extend function (12) to better exploit class ties by considering the connections between positive classes. We give out the extended loss function as follows:

$$G_{[Exatt]} = \sum_{c^* \in L_k} (\sum_{c^+ \in L_k} \rho[0, \sigma^+ - \mathcal{F}(s^{c^*}, c^+)]_+ + \rho[0, \sigma^- + \mathcal{F}(s^{c^*}, c^-)]_+) \tag{14}$$

Pro.	Training	Test
SemE.	17.63%	16.71%
Riedel	72.52%	96.26%

Table 2: The proportions of NR samples from SemEval-2010 Task 8 dataset and Riedel dataset.

Similar to function (13), we select c^- as follows:

$$c^- = \underset{c \in \mathcal{L} - L_k}{\operatorname{argmax}} \mathcal{F}(s^{c^*}, c) \tag{15}$$

and we use the same method to update this loss function as discussed above. From the function (14), we can see that: for $c^* \in L_k$, after merging the bag representation s with c^*, we share s with all the other positive classes and update the class embeddings of other positive classes with s, in this way, the connections between positive classes can be captured and learned by our model.

In loss function (10), (12) and (14), we combine losses from all positive labels to make joint extraction to capture the class ties among relations. Suppose we make separated extraction, the losses from positive labels will be divided apart and will not get enough information of connections between positive labels, comparing to joint extraction. Connections between positive labels and negative ones are exploited by controlling margins: σ^+ and σ^-.

3.4 Relieving Impact of NR

In relation extraction, the dataset will always contain certain negative samples which do not express relations classified as NR (not relation). Table 2 presents the proportion of NR samples in SemEval-2010 Task 8 dataset[2] (Erk and Strapparava, 2010) and dataset from Riedel et al. (2010), which shows almost data is about NR in the latter dataset. Data imbalance will severely affect the model training and cause the model only sensitive to classes with high proportion (He and Garcia, 2009).

In order to relieve the impact of NR in DS based relation extraction, we cut the propagation of loss from NR, which means if relation c is NR, we set its loss as 0. Our method is similar to Santos et al. (2015) with slight variance. Santos et al. (2015) directly omit the NR class embedding, but we keep it. If we use **ATT** method to combine information across sentences, we can not omit NR class embedding according to function (6) and (7), on the contrary, it will be updated from the negative classes' loss.

Algorithm 1: Merging loss function of **Variant-3**

input : $\mathcal{L}$, (t_k, L_k, X_k) and S_k;
output: $G_{[Exatt]}$;

```
G_[Exatt] ← 0;
for c* ∈ L_k do
    Merge representation s^{c*} by function (5), (6), (7);
    for c+ ∈ L_k do
        if c+ is not NR then
            G_[Exatt] ← G_[Exatt] + ρ[0, σ+ − F(s^{c*}, c+)]+;
    c− ← argmax_{c∈L−L_k} F(s^{c*}, c);
    G_[Exatt] ← G_[Exatt] + ρ[0, σ− + F(s^{c*}, c−)]+;
return G_[Exatt];
```

In Algorithm 1, we give out the pseudocodes of merging loss with **Variant-3** and considering to relieve the impact of NR.

4 Experiments

4.1 Dataset and Evaluation Criteria

We conduct our experiments on a widely used dataset, developed by Riedel et al. (2010) and has been used by Hoffmann et al. (2011), Surdeanu et al. (2012), Zeng et al. (2015) and Lin et al. (2016). The dataset aligns Freebase relation facts with the New York Times corpus, in which training mentions are from 2005-2006 corpus and test mentions from 2007.

Following Mintz et al. (2009), we adopt held-out evaluation framework in all experiments. Aggregated precision/recall curves are drawn and precision@N (P@N) is reported to illustrate the model performance.

4.2 Experimental Settings

Word Embeddings. We use a word2vec tool that is gensim[3] to train word embeddings on NYT corpus. Similar to Lin et al. (2016), we keep the words that appear more than 100 times to construct word dictionary and use "UNK" to represent the other ones.

[2]This is a dataset for relation extraction in traditional supervision framework.

[3]http://radimrehurek.com/gensim/models/word2vec.html

Parameter Name	Symbol	Value
Window size	d^{win}	3
Sentence. emb. dim.	d^f	690
Word. emb. dim.	d^1	50
Position. emb. dim.	d^2	5
Batch size	$\mathcal{B}$	160
Learning rate	λ	0.03
Dropout pos.	p	0.5

Table 3: Hyper-parameter settings.

Hyper-parameter Settings. Three-fold validation on the training dataset is adopted to tune the parameters following Surdeanu et al. (2012). We use grid search to determine the optimal hyper-parameters. We select word embedding size from $\{50, 100, 150, 200, 250, 300\}$. Batch size is tuned from $\{80, 160, 320, 640\}$. We determine learning rate among $\{0.01, 0.02, 0.03, 0.04\}$. The window size of convolution is tuned from $\{1, 3, 5\}$. We keep other hyper-parameters same as Zeng et al. (2015): the number of kernels is 230, position embedding size is 5 and dropout rate is 0.5. Table 3 shows the detailed parameter settings.

4.3 Comparisons with Baselines

Baseline. We compare our model with the following baselines:

- **Mintz** (Mintz et al., 2009) the original distantly supervised model.
- **MultiR** (Hoffmann et al., 2011) a multi-instance learning based graphical model which aims to address overlapping relation problem.
- **MIML** (Surdeanu et al., 2012) also solving overlapping relations in a multi-instance multi-label framework.
- **PCNN+ATT** (Lin et al., 2016) the state-of-the-art model in dataset of Riedel et al. (2010) which applies **ATT** to combine the sentences.

Results and Discussion. We compare our three variants of loss functions with the baselines and the results are shown in Figure 3. From the results we can see that: (1) Rank + AVE (Variant-1) achieves comparable results with PCNN+ATT; (2) Rank + ATT (Variant-2) and Rank + ExATT (Variant-3) significantly outperform PCNN + ATT with much higher precision and slightly higher recall in whole view; (3) Rank + ExATT (Variant-3) exhibits the best performances comparing with all the other methods including PCNN + ATT, Rank + AVE and Rank + ATT.

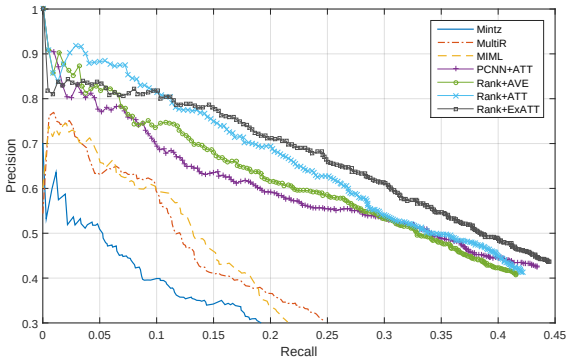


Figure 3: Performance comparison of our model and the baselines.

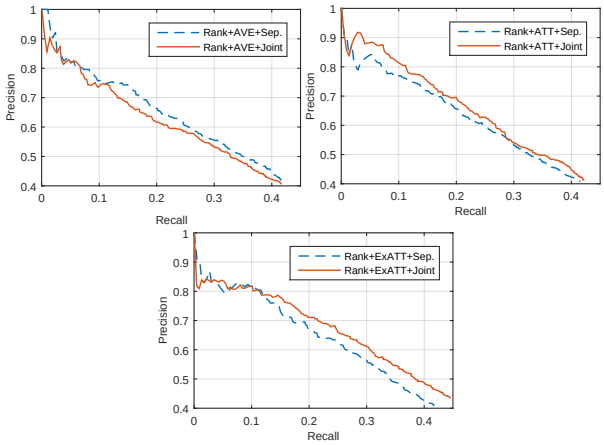


Figure 4: Results for impact of joint extraction and class ties with methods of Rank + AVE, Rank + ATT and Rank + ExATT under the setting of relieving impact of NR.

4.4 Impact of Joint Extraction and Class Ties

In this section, we conduct experiments to reveal the effectiveness of our model to learn class ties with three variant loss functions mentioned above, and the impact of class ties for relation extraction. As mentioned above, we make joint extraction to learn class ties, so to achieve the goal of this set of experiments, we compare joint extraction with separated extraction. To make separated extraction, we divide the labels of entity tuple into single label and for one relation label we only select the sentences expressing this relation, then we use this dataset to train our model with the three variant loss functions. We conduct experiments with Rank + AVE (Variant-1), Rank + ATT (Variant-2) and Rank + ExATT (Variant-3) relieving impact of NR. Aggregated P/R curves are drawn and precisions@N (100, 200, $\cdots$, 500) are reported to show the model performances.

P@N(%)	100	200	300	400	500	Ave.
R.+AVE+J.	81.3	76.4	74.6	69.6	66.0	73.6
R.+AVE+S.	**82.4**	**79.6**	74.6	**74.4**	**69.9**	**76.2**
R.+ATT+J.	**87.9**	**84.3**	**78.0**	**74.9**	**70.3**	**79.1**
R.+ATT+S.	82.4	79.1	75.9	71.9	69.5	75.7
R.+ExATT+J.	**83.5**	82.2	78.7	**77.2**	**73.1**	**79.0**
R.+ExATT+S.	82.4	**82.7**	**79.4**	74.2	69.2	77.6

Table 4: Precisions for top 100, 200, 300, 400, 500 and average of them for impact of joint extraction and class ties.

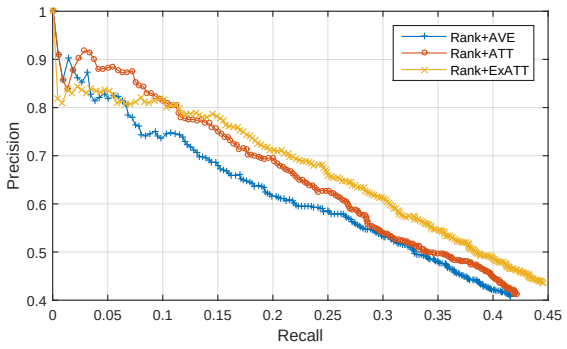


Figure 5: Results for comparisons of variant joint extractions.

P@N(%)	100	200	300	400	500	Ave.
R.+AVE	81.3	76.4	74.6	69.6	66.0	73.6
R.+ATT	**87.9**	**84.3**	78.0	74.9	70.3	**79.1**
R.+ExATT	83.5	82.2	**78.7**	**77.2**	**73.1**	79.0

Table 5: Precisions for top 100, 200, 300, 400, 500 and average of them for Rank + AVE, Rank + ATT and Rank + ExATT.

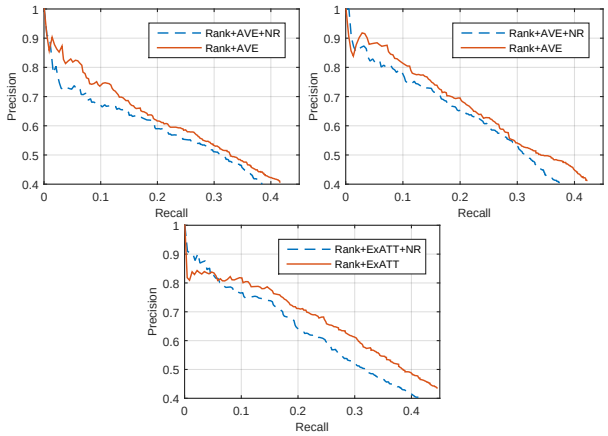


Figure 6: Results for impact of relation NR with methods of Rank + AVE, Rank + ATT and Rank + ExATT. "+NR" means not relieving impact of NR.

Experimental results are shown in Figure 4 and Table 4. From the results we can see that: (1) For Rank + ATT and Rank + ExATT, joint extraction exhibits better performance than separated extraction, which demonstrates class ties will improve relation extraction and the two methods are effective to learn class ties; (2) For Rank + AVE, surprisingly joint extraction does not keep up with separated extraction. For the second phenomenon, the explanation may lie in the **AVE** method to aggregate sentences will incorporate noise data consistent with the finding in Lin et al. (2016). When make joint extraction, we will combine all sentences containing the same entity tuple no matter which class type is expressed, so it will engender much noise if we only combine them equally.

4.5 Comparisons of Variant Joint Extractions

To make joint extraction, we have proposed three variant loss functions including Rank + AVE, Rank + ATT and Rank + ExATT in the above discussion and Figure 3 shows that the three variants achieve different performances. In this experiment, we aim to compare the three variants in detail. We conduct the experiments with the three variants under the setting of relieving impact of NR and joint extraction. We draw the P/R curves and report the top N (100, 200, $\cdots$, 500) precisions to compare model performance with the three variants.

From the results as shown in Figure 5 and Table 5 we can see that: (1) Comparing Rank + AVE with Rank + ATT, from the whole view, they can achieve the similar maximal recall point, but Rank + ATT exhibits higher precision in all range of recall; (2) Comparing Rank + ATT with Rank + ExATT, Rank + ExATT achieves much better performance with broader range of recall and higher precision in almost range of recall.

4.6 Impact of NR Relation

The goal of this experiment is to inspect how much relation of NR can affect the model performance. We use Rank + AVE, Rank + ATT, Rank + ExATT under the setting of relieving impact of NR or not to conduct experiments. We draw the aggregated P/R curves as shown in Figure 6, from which we can see that after relieving the impact of NR, the model performance can be improved significantly.

Then we further evaluate the impact of NR for convergence behavior of our model in model train-

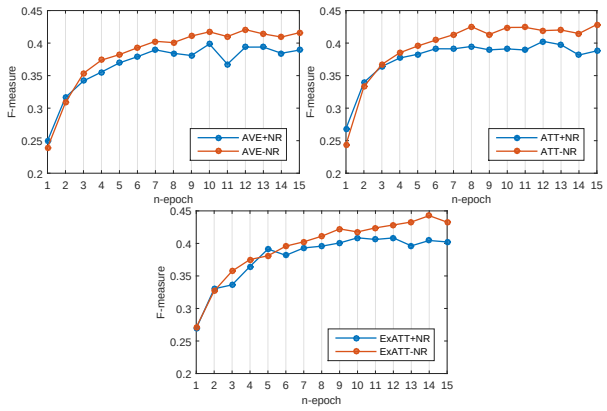


Figure 7: Impact of NR for model convergence. "+NR" means not relieving NR impact; "-NR" is opposite.

ing. Also with the three variant loss functions, in each iteration, we record the maximal value of F-measure [4] to represent the model performance at current epoch. Model parameters are tuned for 15 times and the convergence curves are shown in Figure 7. From the result, we can find out: "+NR" converges quicker than "-NR" and arrives to the final score at the around 11 or 12 epoch. In general, "-NR" converges more smoothly and will achieve better performance than "+NR" in the end.

4.7 Case Study

Joint vs. Sep. Extraction (Class Ties). We randomly select an entity tuple *(Cuyahoga County, Cleveland)* from test set to see its scores for every relation class with the method of Rank + ATT under the setting of relieving impact of NR with joint extraction and separated extraction. This entity tuple have two relations: */location/./county_seat* and */location/./contains*, which derive from the same root class and they have weak class ties for they all relating to topic of "location". We rescale the scores by adding value 10. The results are shown in Figure 8, from which we can see that: under joint extraction setting, the two gold relations have the highest scores among the other relations but under separated extraction setting, only */location/./contains* can be distinguished from the negative relations, which demonstrates that joint extraction is better than separated extraction by capturing the class ties between relations.

[4] $F = 2 * P * R/(P + R)$

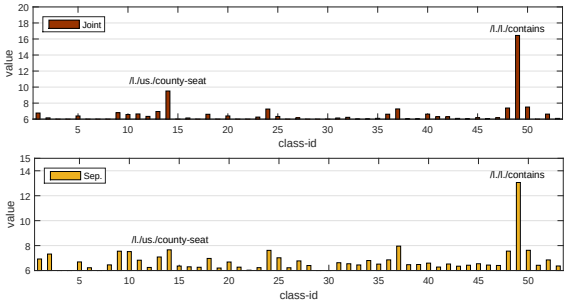


Figure 8: The output scores for every relation with method of Rank + ATT. The top is under joint extraction setting; the bottom is under separated extraction.

5 Conclusion and Future Works

In this paper, we leverage class ties to enhance relation extraction by joint extraction using pairwise ranking combined with CNN. An effective method is proposed to relieve the impact of NR for model training. Experimental results on a widely used dataset show that leveraging class ties will enhance relation extraction and our model is effective to learn class ties. Our method significantly outperforms the baselines.

In the future, we will focus on two aspects: (1) Our method in this paper considers pairwise intersections between labels, so to better exploit class ties, we will extend our method to exploit all other labels' influences on each relation for relation extraction, transferring *second-order* to *high-order* (Zhang and Zhou, 2014); (2) We will focus on other problems by leveraging class ties between labels, specially on multi-label learning problems (Zhou et al., 2012) such as multi-category text categorization (Rousu et al., 2005) and multi-label image categorization (Zha et al., 2008).

Acknowledgments

Firstly, we would like to thank Xianpei Han and Kang Liu for their valuable suggestions on the initial version of this paper, which have helped a lot to improve the paper. Secondly, we also want to express gratitudes to the anonymous reviewers for their hard work and kind comments, which will further improve our work in the future. This work was supported by the National High-tech Research and Development Program (863 Program) (No. 2014AA015105) and National Natural Science Foundation of China (No. 61602490).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of KDD*. pages 1247–1250.

Katrin Erk and Carlo Strapparava, editors. 2010. *Proceedings of SemEval*. The Association for Computer Linguistics.

Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. 2008. Multilabel classification via calibrated label ranking. *Machine learning* 73(2):133–153.

Xianpei Han and Le Sun. 2016. Global distant supervision for relation extraction. In *Proceedings of AAAI*. pages 2950–2956.

Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* 21(9):1263–1284.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL-HLT*. Association for Computational Linguistics, pages 541–550.

Nathalie Japkowicz and Shaju Stephen. 2002. The class imbalance problem: A systematic study. *Intelligent data analysis* 6(5):429–449.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521(7553):436–444.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*. volume 1, pages 2124–2133.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3(3):225–331.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*. pages 1412–1421.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*. Association for Computational Linguistics, pages 1003–1011.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*. Springer, pages 148–163.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. 2005. Learning hierarchical multi-category text classification models. In *Proceeding of ICML*. ACM, pages 744–751.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceeding of ACL*. pages 626–634.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 373–382.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*. Association for Computational Linguistics, pages 455–465.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of ACL, Volume 1: Long Papers*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. WSABIE: scaling up to large vocabulary image annotation. In *Proceedings of IJCAI*. pages 2764–2770.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*. pages 17–21.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceeding of COLING*. pages 2335–2344.

Zheng-Jun Zha, Xian-Sheng Hua, Tao Mei, Jingdong Wang, Guo-Jun Qi, and Zengfu Wang. 2008. Joint multi-label multi-instance learning for image classification. In *CVPR*. IEEE, pages 1–8.

Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering* 18(10):1338–1351.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering* 26(8):1819–1837.

Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. 2015. Deep semantic ranking based hashing for multi-label image retrieval. In *Proceedings of CVPR*. pages 1556–1564.

Hao Zheng, Zhoujun Li, Senzhang Wang, Zhao Yan, and Jianshe Zhou. 2016. Aggregating inter-sentence information to enhance relation extraction. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceeding of ACL*. page 207.

Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. 2012. Multi-instance multi-label learning. *Artificial Intelligence* 176(1):2291–2320.

Search-based Neural Structured Learning for Sequential Question Answering

Mohit Iyyer*
Department of Computer Science and UMIACS
University of Maryland, College Park
miyyer@umd.edu

Wen-tau Yih, Ming-Wei Chang
Microsoft Research
Redmond, WA 98052
{scottyih,minchang}@microsoft.com

Abstract

Recent work in semantic parsing for question answering has focused on long and complicated questions, many of which would seem unnatural if asked in a normal conversation between two humans. In an effort to explore a conversational QA setting, we present a more realistic task: answering sequences of simple but inter-related questions. We collect a dataset of 6,066 question sequences that inquire about semi-structured tables from Wikipedia, with 17,553 question-answer pairs in total. To solve this sequential question answering task, we propose a novel dynamic neural semantic parsing framework trained using a weakly supervised reward-guided search. Our model effectively leverages the sequential context to outperform state-of-the-art QA systems that are designed to answer highly complex questions.

1 Introduction

Semantic parsing, which maps natural language text to meaning representations in formal logic, has emerged as a key technical component for building question answering systems (Liang, 2016). Once a natural language question has been mapped to a formal query, its answer can be retrieved by executing the query on a back-end structured database.

One of the main focuses of semantic parsing research is how to address *compositionality* in language, and complicated questions have been specifically targeted in the design of a recently-released QA dataset (Pasupat and Liang, 2015). Take for example the following question: "*of those actresses who won a Tony after 1960, which one took the most amount of years after winning the Tony to win an Oscar?*" The corresponding logical form is highly compositional; in order to answer it, many sub-questions must be implicitly answered in the process (e.g., *"who won a Tony after 1960?"*).

While we agree that semantic parsers *should* be able to answer very complicated questions, in reality these questions are rarely issued by users.[1] Because users can interact with a QA system repeatedly, there is no need to assume a single-turn QA setting where the exact *question intent* has to be captured with just one complex question. The same intent can be more naturally expressed through a sequence of simpler questions, as shown below:

1. *What actresses won a Tony after 1960?*
2. *Of those, who later won an Oscar?*
3. *Who had the biggest gap between their two award wins?*

Decomposing complicated intents into multiple related but simpler questions is arguably a more effective strategy to explore a topic of interest, and it reduces the cognitive burden on both the person who asks the question and the one who answers it.[2]

In this work, we study semantic parsing for answering *sequences* of simple related questions. We collect a dataset of question sequences called SequentialQA (SQA; Section 2)[3] by asking crowdsourced workers to decompose complicated questions sampled from the WikiTableQuestions dataset (Pasupat and Liang, 2015) into multiple easier ones. SQA, which contains 6,066 question sequences with 17,553 total question-answer pairs, is to the best of our knowledge the first semantic parsing dataset for sequential question answering. Section 3 describes our novel dynamic neural semantic parsing framework (DynSP), a weakly su-

*Work done during an internship at Microsoft Research

[1] For instance, there are only 3.75% questions with more than 15 words in WikiAnswers (Fader et al., 2014).

[2] Studies have shown increased sentence complexity links to longer reading times (Hale, 2006; Levy, 2008; Frank, 2013).

[3] Available at http://aka.ms/sqa

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1821–1831
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1167

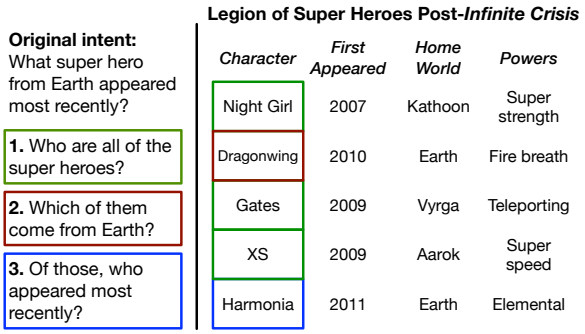


Figure 1: An example question sequence created from a compositional question intent. Workers must write questions whose answers are subsets of cells in the table.

pervised structured-output learning approach based on reward-guided search that is designed for solving sequential QA. We demonstrate in Section 4 that DynSP achieves higher accuracies than existing systems on SQA, and we offer a qualitative analysis of question types that our method answers effectively, as well as those on which it struggles.

2 A Dataset of Question Sequences

We collect the SequentialQA (SQA) dataset via crowdsourcing by leveraging WikiTableQuestions (Pasupat and Liang, 2015, henceforth WTQ), which contains highly compositional questions associated with HTML tables from Wikipedia. Each crowdsourcing task contains a long, complex question originally from WTQ as the question *intent*. The workers are asked to compose a sequence of simpler questions that lead to the final intent; an example of this process is shown in Figure 1.

To simplify the task for workers, we only use questions from WTQ whose answers are cells in the table, which excludes those involving arithmetic and counting. We likewise also restrict the questions our workers can write to those answerable by only table cells. These restrictions speed the annotation process because workers can just click on the table to answer their question. They also allow us to collect answer coordinates (row and column in the table) as opposed to answer text, which removes many normalization issues for answer string matching in evaluation. Finally, we only use long questions that contain nine or more words as intents; shorter questions tend to be simpler and are thus less amenable to decomposition.

2.1 Properties of SQA

In total, we used 2,022 question intents from the train and test folds of the WTQ for decomposition. We had three workers decompose each intent, resulting in 6,066 unique questions sequences containing 17,553 total question-answer pairs (for an average of 2.9 questions per sequence). We divide the dataset into train and test using the original WTQ folds, resulting in an 83/17 train/test split. Importantly, just like in WTQ, none of the tables in the test set are seen in the training set.

We identify three frequently-occurring question classes: *column selection*, *subset selection*, and *row selection*.[4] In column selection questions, the answer is an entire column of the table; these questions account for 23% of all questions in SQA. Subset and row selection are more complicated than column selection, as they usually contain coreferences to the previous question's answer. In subset selections, the answer is a subset of the previous question's answer; similarly, the answers to row selections occur in the same row(s) as the previous answer but in a different column. Subset selections make up 27% of SQA, while row selections are an additional 19%. The remaining 31% contains more complex combinations of these three types.

We also observe dramatic differences in the types of questions that are asked at each position of the sequence. For example, 51% of the first questions in the sequences are column selections (e.g., "*what are all of the teams?*"). This number dwindles to just 18% when we look at the second question of each sequence, which indicates that the collected sequences start with general questions and progress to more specific ones.

3 Dynamic Neural Semantic Parsing

The unique setting of SQA provides both opportunities and challenges. On the one hand, it contains short questions with less compositionality, which in theory should reduce the difficulty of the semantic parsing problem; on the other hand, the additional contextual dependencies of the preceding questions and their answers increase modeling complexity. These observations lead us to propose a dynamic neural semantic parsing framework (DynSP) trained using a reward-guided search pro-

[4] In the example sequence "*what are all of the tournaments? in which one did he score the least points? on what date was that?*", the first question is a column selection, the second is a subset selection, and the last one is a row selection.

cedure for solving SQA.

Given a question (optionally along with previous questions and answers) and a table, DynSP formulates the semantic parsing problem as a state–action search problem. Each state represents a complete or partial parse, while each action corresponds to an operation to extend a parse. The goal during inference is to find an end state with the highest score as the predicted parse.

The quality of the induced semantic parse obviously depends on the scoring function. In our design, the score of a state is determined by the scores of actions taken from the initial state to the target state, which are predicted by different neural network modules based on action type. By leveraging a margin-based objective function, the model learning procedure resembles several structured-output learning algorithms such as structured SVMs (Tsochantaridis et al., 2005), but can take either strong or weak supervision seamlessly.

DynSP is inspired by STAGG, a search-based semantic parser (Yih et al., 2015), as well as the dynamic neural module network (DNMN) of Andreas et al. (2016). Much like STAGG, DynSP chains together different modules as search progresses; however, these modules are implemented as neural networks, which enables end-to-end training as in DNMN. The key difference between DynSP and DNMN is that in DynSP the network structure of an example is not predetermined. Instead, different network structures are constructed dynamically as our learning procedure explores the state space. It is straightforward to answer *sequential* questions using our framework: we allow the model to take the previous question and its answers as input, with a slightly modified action space to reflect a *dependent* semantic parse. The same search / learning procedure is then able to effortlessly adapt to the new setting. In this section, we first describe the formal language underlying DynSP, followed by the model formulation and learning algorithm.

3.1 Semantic parse language

Because tables are used as the data source to answer questions in SQA, we decide to form our semantic parses in an SQL-like language[5]. Our parses consist of two parts: a ***select*** statement and conjunctions of zero or more ***conditions***.

A ***select*** statement is associated with a column name, which is referred to as the *answer* column. Conditions enforce additional constraints on which cells in the answer column can be chosen; a ***select*** statement without any conditions indicates that an entire column of the table is the answer to the question. In particular, each condition contains a column name as the *condition* column and an operator with zero or more arguments. The operators in this work include: $=, \neq, >, \geq, <, \leq$, $\arg\min, \arg\max$. A cell in the answer column is only a legitimate answer if the cell of the corresponding row in the condition column satisfies the constraint defined by the operator and its arguments. As a concrete example, suppose the data source is the same table in Fig. 1. The semantic parse of the question "*Which super heroes came from Earth and first appeared after 2009?*" is "Select **Character** Where {**Home World** = *Earth*} $\wedge$ {**First Appeared** $>$ *2009*}" and the answers are {Dragonwing, Harmonia}.

In order to handle the *sequential* aspect of SQA, we extend the semantic parse language by adding a preamble statement ***subsequent***. A ***subsequent*** statement contains only conditions, as it essentially adds constraints to the semantic parse of the previous question. For instance, if the follow-up question is "*Which of them breathes fire?*", then the corresponding semantic parse is "Subsequent Where {**Powers** = *Fire breath*}". The answer to this question is {Dragonwing}, a subset of the previous answer.

3.2 Model formulation

We start introducing our model design by first defining the state and action space. Let $\mathcal{S}$ be the set of states and $\mathcal{A}$ the set of all actions. A state $s \in \mathcal{S}$ is simply a sequence of variable length of actions $\{a_1, a_2, a_3, \cdots, a_t\}$, where $a_i \in \mathcal{A}$. An empty sequence, $s_0 = \phi$, is a special state used as the starting point of search.

As mentioned earlier, a state represents a (partial) semantic parse of *one* question. Each action is thus a legitimate operation that can be added to *grow* the semantic parse. Our action space design is tied closely to the statements defined by our parse language; in particular, an action *instance* is either a complete or partial statement, and action instances are grouped by *type*. For example, ***select*** and ***subsequent*** operations are two action types. A ***condition*** statement is formed by two different action types:

[5] Our framework is not restricted to the formal language we use in this work. In addition, the structured query can be straightforwardly represented in other formal languages, such as the lambda DCS logic used in (Pasupat and Liang, 2015).

Id	Type	# Action instances
$\mathcal{A}_1$	Select-column	# columns
$\mathcal{A}_2$	Cond-column	# columns
$\mathcal{A}_3$	Op-Equal (=)	# rows
$\mathcal{A}_4$	Op-NotEqual ($\neq$)	# rows
$\mathcal{A}_5$	Op-GT ($>$)	# numbers / datetimes
$\mathcal{A}_6$	Op-GE ($\geq$)	# numbers / datetimes
$\mathcal{A}_7$	Op-LT ($<$)	# numbers / datetimes
$\mathcal{A}_8$	Op-LE ($\leq$)	# numbers / datetimes
$\mathcal{A}_9$	Op-ArgMin	# numbers / datetimes
$\mathcal{A}_{10}$	Op-ArgMax	# numbers / datetimes
$\mathcal{A}_{11}$	Subsequent	1
$\mathcal{A}_{12}$	S-Cond-column	# columns
$\mathcal{A}_{13}$	S-Op-Equal (=)	# rows
$\mathcal{A}_{14}$	S-Op-NotEqual ($\neq$)	# rows
$\mathcal{A}_{15}$	S-Op-GT ($>$)	# numbers / datetimes
$\mathcal{A}_{16}$	S-Op-GE ($\geq$)	# numbers / datetimes
$\mathcal{A}_{17}$	S-Op-LT ($<$)	# numbers / datetimes
$\mathcal{A}_{18}$	S-Op-LE ($\leq$)	# numbers / datetimes
$\mathcal{A}_{19}$	S-Op-ArgMin	# numbers / datetimes
$\mathcal{A}_{20}$	S-Op-ArgMax	# numbers / datetimes

Table 1: Types of actions and the number of action instances in each type. Numbers / datetimes are the mentions discovered in the question (plus the previous question if it is a subsequent condition).

(1) selection of the condition column, and (2) the comparison operator. The instances of each action type differ in their arguments (e.g., column names, or specific cells in a column). Because conditions in a ***subsequent*** parse rely on previous questions and answers, they belong to different action types from regular conditions. Table 1 summarizes the action space defined in this work.

Any state that represents a complete and legitimate parse is an end state. Notice that search does not necessarily need to stop at an end state, because adding more actions (e.g., condition statements) can lead to another end state. Take the same example question from before: "*Which super heroes came from Earth and first appeared after 2009?*". One action sequence that represents the parse is {($\mathcal{A}_1$) select-column **Character**, ($\mathcal{A}_2$) cond-column **Home World**, ($\mathcal{A}_3$) op-equal *Earth*, ($\mathcal{A}_2$) cond-column **First Appeared**, ($\mathcal{A}_5$) op-gt *2009*}.

Notice that many states represent semantically equivalent parses (e.g., those with the same actions ordered differently, or states with repeated conditions). To prune the search space, we introduce the function $Act(s) \subset \mathcal{A}$, which defines the actions that can be taken when given a state s. Borrowing the idea of *staged* state generation in (Yih et al., 2015), we choose a default ordering of actions based on their types, dictating that a *select* action must be picked first and that a *condition-column* needs to be determined before the *operator* is chosen. The full transition diagram is presented in Fig. 2. Note that to implement this transition order, we only need to check the last action in the state. In addition, we also disallow adding duplicates of actions that already exist in the state.

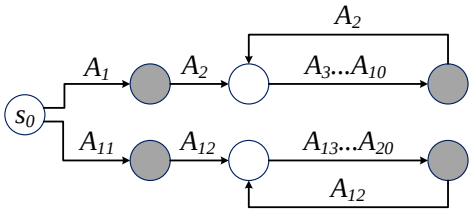


Figure 2: Possible action transitions based on their types (see Table 1). Shaded circles are end states.

We use beam search to find an end state with the highest score for inference. Let s_t be a state consisting of a sequence of actions $a_1, a_2, \cdots, a_t$. The state value function V is defined recursively as $V(s_t) = V(s_{t-1}) + \pi(s_{t-1}, a_t), V(s_0) = 0$, where the *policy* function $\pi(s, a)$ scores an action $a \in Act(s)$ given the current state.

3.3 Policy function

The intuition behind the policy function can be summarized as follows. Halfway through the construction of a semantic parse, the policy function measures the quality of an immediate action that can be taken next given the current state (i.e., the question and actions that have previously been chosen). To enable integrated, end-to-end learning, the policy function in our framework is parameterized using neural networks. Because each action type has very different semantics, we design different network structures (i.e., modules) accordingly.

Most of our network structures encourage learning semantic matching functions between the words in the question and table (either the column names or cells). Here we illustrate the design using the *select-column* action type ($\mathcal{A}_1$). Conceptually, the corresponding module is a combination of various matching scores. Let W_Q be the embeddings of words in the question and W_C be the embeddings of words in the target column name. The component matching functions are:

$$f_{max} = \frac{1}{|W_C|} \sum_{w_c \in W_C} \max_{w_q \in W_Q} w_q^T w_c$$

$$f_{avg} = \left(\frac{1}{|W_C|} \sum_{w_c \in W_C} w_c \right)^T \left(\frac{1}{|W_Q|} \sum_{w_q \in W_Q} w_q \right)$$

Essentially, for each word in the column name, f_{max} finds the highest matching question word and outputs the average score. Conversely, f_{avg} simply uses the average word vectors of the question and column name and returns their inner product. In another variant of f_{avg}, we replace the question representation with the output of a bi-directional LSTM model. These matching component functions are combined by a 2-layer feed-forward neural network, which outputs a scalar value as the action score. Details of the neural module design for other action types can be found in Appendix A.

3.4 Model learning

Because the state value function V is defined recursively as the sum of scores of actions in the sequence, the goal of model optimization is to learn the parameters in the neural networks behind the policy function. Let θ be the collection of all the model parameters. Then the state value function can be written as: $V_\theta(s_t) = \sum_{i=1}^{t} \pi_\theta(s_{i-1}, a_i)$.

In a fully supervised setting where the correct semantic parse of each question is available, learning the policy function can be reduced to a sequence prediction problem. However, while having full supervision leads to a better semantic parser, collecting the correct parses requires a much more sophisticated UI design (Yih et al., 2016). In many scenarios, such as the one in the SQA dataset, it is often the case that only the answers to the questions are available. Adapting a learning algorithm to this *weakly* supervised setting is thus critical.

Generally speaking, weakly supervised semantic parsers operate on one assumption — a candidate semantic parse is treated as a correct one if it results in answers that are identical to the gold answers. Therefore, a straightforward modification of existing structured learning algorithms in our setting is to use any semantic parse found to evaluate to the correct answers during beam search as a *reference* parse, and then update the model parameters accordingly. In practice, however, this approach is often problematic: the search space can grow enormously, and when coupled with poor model performance early during training, this leads to beams that contain no parses evaluating to the correct answer. As a result, learning becomes inefficient and takes a long time to converge.

In this work, we propose a conceptually simple learning algorithm for weakly supervised training that sidesteps the inefficient learning problem. Our key insight is to conduct inference using a beam search procedure guided by an *approximate reward* function. The search procedure is executed *twice* for each training example, one for finding the best possible reference semantic parse and the other for finding the predicted semantic parse to update the model. Our framework is suitable for learning from either implicit or explicit supervision, and is detailed in a companion paper (Peng et al., 2017). Below we describe how we adapt it to the semantic parsing problem in this work.

Approximate reward Let $A(s)$ be the answers retrieved by executing the semantic parse represented by state s, and let A^* be the set of gold answers of a given question. We define the *reward* $R(s; A^*) = \mathbb{1}[A(s) = A^*]$, or the accuracy of the retrieved answers. We use $R(s)$ as the abbreviation for $R(s; A^*)$. A state s with $R(s) = 1$ is called a goal state. Directly using this reward function in search of goal states can be difficult, as rewards of most states are 0. However, even when the answers from a semantic parse are not completely correct, some overlap with the gold answers can still hint that the state is close to a goal state, thus providing useful information to guide search. To formalize this idea, we define an *approximated reward* $\tilde{R}(s)$ in this work using the Jaccard coefficient ($\tilde{R}(s) = |A(s) \cap A^*|/|A(s) \cup A^*|$). If s is a goal state, then obviously $\tilde{R}(s) = R(s) = 1$. Also because our actions effectively add additional constraints to exclude some table cells, any succeeding states of s' with $\tilde{R}(s') = 0$ will also have 0 approximate reward and can be pruned from search immediately.

We use the approximate reward $\tilde{R}$ to guide our beam search to find the reference parses (i.e., goal states). Some variations of the approximate reward can be used to make learning more efficient. For instance, we use the model score for tie-breaking, effectively making the approximate reward function depend on the model parameters:

$$\tilde{R}_\theta(s) = |A(s) \cap A^*|/|A(s) \cup A^*| + \epsilon V_\theta(s), \quad (1)$$

where ϵ is a small constant. When a goal state is not found, the state with the highest approximate reward can still be used as a surrogate reference.

Updating parameters The model parameters are updated by first finding the most *violated* state $\hat{s}$ and then comparing $\hat{s}$ with a reference state s^* to compute a loss. The idea of finding the most violated state comes from Taskar et al. (2004), with the

Algorithm 1 Model parameter updates

1: **for** pick a labeled data (x, A^*) **do**
2: $\quad s^* \leftarrow \arg\max_{s \in \mathcal{E}(x)} \tilde{R}(s; A^*)$
3: $\quad \hat{s} \leftarrow \arg\max_{s \in \mathcal{E}(x)} V_\theta(s) - \tilde{R}(s; A^*)$
4: $\quad$ update θ by minimizing $\max(\mathcal{L}(s), 0)$
5: **end for**

intuition that the learning algorithm should make the state value function behave similarly to the reward. Formally, for every state s, we would like the value function to satisfy the following constraint:

$$V_\theta(s^*) - V_\theta(s) \geq R(s^*) - R(s) \quad (2)$$

$R(s^*) - R(s)$ is thus the margin. As discussed above, we use approximate reward function $\tilde{R}_\theta$ instead of the true reward. We want to update the model parameters θ to make sure that the constraint is satisfied. When the constraint is violated, the degree of violation can be written as:

$$\mathcal{L}(s) = V_\theta(s) - V_\theta(s^*) - \tilde{R}_\theta(s) + \tilde{R}_\theta(s^*) \quad (3)$$

In the algorithm, we want to find the state such that the corresponding constraint is most violated. Finding the most violated state is then equivalent to finding the state with the highest value of $V_\theta(s) - \tilde{R}_\theta(s)$ as the other two terms are constant.

Algorithm 1 sketches the key steps of our method in each iteration. It first picks a training instance (x and y), where x represents the table and the question, and y is the gold answer set. The approximate reward function $\tilde{R}$ is defined by y, while $\mathcal{E}(x)$ is the set of end states for this instance. Line 2 finds the best reference and Line 3 finds the most violated state, both relying on beam search for approximate inference. Line 4 computes the gradient of the loss in Eq. (3), which is then used in backpropagation to update the model parameters.

4 Experiments

Since the questions in SQA are decomposed from those in WTQ, we compare our method, `DynSP`, to two existing semantic parsers designed for WTQ: (1) the *floating parser* (`FP`) of Pasupat and Liang (2015), and (2) the *neural programmer* (`NP`) of Neelakantan et al. (2017). We describe below each system's configurations in more detail and qualitatively compare and contrast their performance on SQA.

Floating parser: The floating parser (Pasupat and Liang, 2015) maps questions to logical forms and then executes them on the table to retrieve the answers. It was designed specifically for the WTQ task (achieving 37.0% accuracy on the WTQ test set) and differs from other semantic parsers by not anchoring predicates to tokens in the question, relying instead on typing constraints to reduce the search space. Using `FP` as-is results in poor performance on SQA because the system is configured for questions with single answers, while SQA contains many questions with multiple-cell answers. We address this issue by removing a pruning hyperparameter (*tooManyValues*) and features that add bias on the denotation size.

Neural programmer: The neural programmer proposed by Neelakantan et al. (2017) has shown promising results on WTQ, achieving accuracies on par with those of `FP`. Similar to our method, `NP` contains specialized neural modules that perform discrete operations such as argmax and argmin, and it is able to chain together multiple modules to answer a single question. However, module selection in `NP` is computed via soft attention (Cho et al., 2014), and information is propagated from one module to the next using a recurrent neural network. Since module selection is not tied to a pre-defined parse language like `DynSP`, `NP` simply runs for a fixed number of recurrent timesteps per question rather than growing a parse until it is complete.

Comparing the baseline systems: `FP` and `NP` exemplify two very different paradigms for designing a semantic parsing system to answer questions using structured data. `FP` is a feature-rich system that aims to output the correct semantic parse (in a logical parse language) for a given question. On the other hand, the end-to-end neural network of `NP` relies on its modular architectures to output a probability distribution over cells in a table given a question. While `NP` can learn more powerful neural matching functions between questions and tables than `FP`'s simpler feature-based matching, `NP` cannot produce a complete, discrete semantic parse, which means that its actions can only be interpreted coarsely by looking at the order of the modules selected at each timestep.[6] Furthermore, `FP`'s design theoretically allows it to operate on partial tables

[6] Since `NP` uses a fixed number of timesteps for each question, the module order is not guaranteed to correspond to a complete parse.

indirectly through an API, which is necessary if tables are large and stored in a backend database, while NP requires upfront access to the full tables to facilitate end-to-end model differentiability.[7]

Even though FP and NP are powerful systems designed for the more difficult, compositional questions in WTQ, our method outperforms both systems on SQA when we consider all questions within a sequence independently of each other (a fair comparison), demonstrating the power of our search-based semantic parsing framework. More interestingly, when we leverage the sequential information by including the ***subsequent*** action, our method improves almost 3% in absolute accuracy.

DynSP combines the best parts of both FP and NP. Given a question, we try to generate its correct semantic parse in a formal language that can be predefined by the choice of structured data source (e.g., SQL). However, we push the burden of feature engineering to neural networks as in NP. Our framework is easier to extend to the sequential setting of SQA than either baseline system, requiring just the additional ***subsequent*** action. FP's reliance on a hand-designed grammar necessitates extra rules that operate over partial tables from the previous question, which if added would blow up the search space. Meanwhile, modifying NP to handle sequential QA is non-trivial due to soft module and answer selection; it is not immediately clear how to constrain predictions for one question based on the probability distribution over table cells from the previous question in the sequence.

To more fairly compare DynSP to the baseline systems, we also experiment with a "concatenated questions" setting, which allows the baselines to access sequential context. Here, we treat concatenated question prefixes of a sequence as additional training examples, where a question prefix includes all questions prior to the current question in the sequence.

For example, suppose the question sequence is: *1. what are all of the teams? 2. of those, which won championships?* For the second question, in addition to the original question–answer pair, we add the concatenated question sequence *"what are all of the teams? of those, which won championships?"* paired with the second question's answer. We refer to these concatenated question baselines as FP^+ and NP^+.

[7] In fact, NP is restricted during training to only questions whose associated tables have fewer than a certain threshold of rows and columns due to computational constraints.

4.1 DynSP implementation details

Unlike previous dynamic neural network frameworks (Andreas et al., 2016; Looks et al., 2017), where each example can have different but *predetermined* structure, DynSP needs to dynamically explores and constructs different neural network structures for each question. Therefore, we choose DyNet (Neubig et al., 2017) as our implementation platform for its flexibility in composing computation graphs. We optimize our model parameters using standard stochastic gradient descent. The word embeddings are initialized with 100-d pretrained GloVe vectors (Pennington et al., 2014) and fine-tuned during training with dropout rate 0.5. For follow-up questions, we choose uniformly at random to use either gold answers to the previous question or the model's previous predictions.[8] We constrain the maximum length of actions to 3 for computational efficiency and set the beam size to 15 in our reported models, as accuracy gains are negligible with larger beam sizes. We train our model for 30 epochs, although the best model on the validation set is usually found within the first 20 epochs. Only CPU is used in model training, and each epoch in the beam size 15 setting takes about 30 minutes to complete.

4.2 Results & Analysis

Table 2 shows the results of the baseline systems as well as our method on SQA's test set. For each system, we show both the overall accuracy, the sequence accuracy (the percentage of sequences for which every question was answered correctly), and the accuracy at each position in the sequence. Our method without any sequential information (DynSP) outperforms the standard baselines, and when the *subsequent* action is added (DynSP^*), we improve both overall and sequence accuracy over the concatenated-question baselines.

With that said, all of the systems struggle to answer all questions within a sequence correctly, despite the fact that each individual question is simpler on average than those in WTQ. Most of the errors made by our system are due to either semantic matching challenges or limitations of the underlying parse language. In the middle example of Figure 3, the first question asks for a list of super heroes; from the model's point of view, *Real name* is a more relevant column than *Character*, although the latter is correct. The second question also con-

[8] Only predicted answers are used at test time.

Model	All	Seq	Pos 1	Pos 2	Pos 3
FP	34.1	7.2	52.6	25.6	**25.9**
NP	39.4	10.8	58.9	35.9	24.6
DynSP	42.0	10.2	**70.9**	35.8	20.1
FP$^+$	33.2	7.7	51.4	22.2	22.3
NP$^+$	40.2	11.8	60.0	35.9	25.5
DynSP*	**44.7**	**12.8**	70.4	**41.1**	23.6

Table 2: Accuracies of all systems on SQA; the models in the first half of the table treat questions independently, while those in the second half consider sequential context. Our method outperforms existing ones both in terms of overall accuracy as well as sequence accuracy.

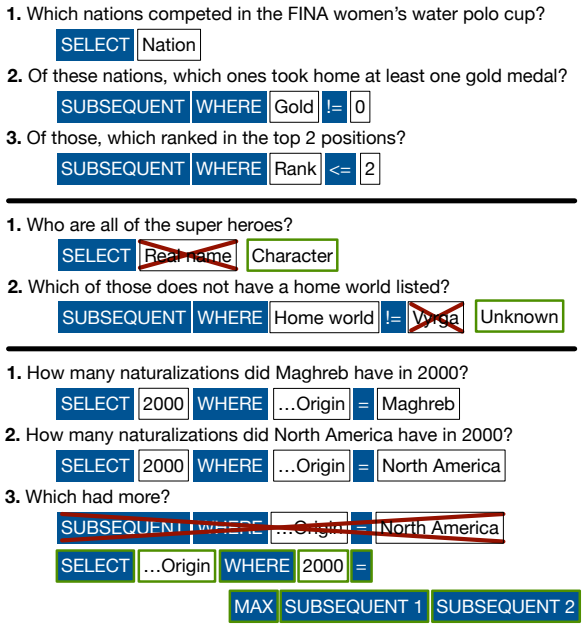


Figure 3: Parses computed by DynSP for three test sequences (actions in blue boxes, values from table in white boxes). *Top*: all three questions are parsed correctly. *Middle*: semantic matching errors cause the model to select incorrect columns and conditions. *Bottom*: The final question is unanswerable due to limitations of our parse language.

tains a challenging matching problem where the unlisted home worlds referred to in the question are marked as *Unknown* in the table. Many of these matching issues are resolved by humans using common sense, which for computers requires far more data than is available in SQA to learn.

Even when there are no tricky discrepancies between question and table text, questions are often complex enough that their semantic parses cannot be expressed in our parse language. Although trivial on the surface, the final question in the bottom sequence of Figure 3 is one such example; the correct semantic parse requires access to the answers of both the first and second question, actions that we have not currently implemented in our language due to concerns with the search space size. Increasing the number of complex actions requires designing smarter optimization procedures, which we leave to future work.

5 Related Work

Previous work on conversational QA has focused on small, single-domain datasets. Perhaps most related to our task is the context-dependent sentence analysis described in (Zettlemoyer and Collins, 2009), where conversations between customers and travel agents are mapped to logical forms after resolving referential expressions. Another dataset of travel booking conversations is used by Artzi and Zettlemoyer (2011) to learn a semantic parser for complicated queries given user clarifications. More recently, Long et al. (2016) collect three contextual semantic parsing datasets (from synthetic domains) that contain coreferences to entities and actions. We differentiate ourselves from these prior works in two significant ways: first, our dataset is not restricted to a particular domain, and second, a major goal of our work is to analyze the different types of sequence progressions people create when they are trying to express a complicated intent.

Complex, interactive QA tasks have also been proposed in the information retrieval community, where the data source is a corpus of newswire text (Kelly and Lin, 2007). We also build on aspects of some existing interactive question-answering systems. For example, the system of Harabagiu et al. (2005) includes a module that predicts what a user will ask next given their current question.

Other than FP and NP, the work of Neural Symbolic Machines (NSM) (Liang et al., 2017) is perhaps the closest to ours. NSM aims to generate formal semantic parses of questions that can be executed on Freebase to retrieve answers, and is trained using the REINFORCE algorithm (Williams, 1992) augmented with approximate gold parses found in a separate curriculum learning stage. In comparison, finding reference parses is an integral part of our algorithm. Our non-

probabilistic, margin-based objective function also helps avoid the need for empirical tricks to handle normalization and proper sampling, which are crucial when applying REINFORCE in practice.

6 Conclusion & Future Work

In this work we move towards a conversational, multi-turn QA scenario in which systems must rely on prior context to answer the user's current question. To this end, we introduce SQA, a dataset that consists of 6,066 unique sequences of inter-related questions about Wikipedia tables, with 17,553 questions-answer pairs in total. To the best of our knowledge, SQA is the first semantic parsing dataset that addresses sequential question answering. We propose `DynSP`, a dynamic neural semantic parsing framework, for solving SQA. By formulating semantic parsing as a state–action search problem, our method learns modular neural network models through reward-guided search. `DynSP` outperforms existing state-of-the-art systems designed for answering complex questions when applied to SQA, and increases the gain after incorporating the subsequent actions.

In the future, we plan to investigate several interesting research questions triggered by this work. For instance, although our current formal language design covers most question types in SQA, it is nevertheless important to extend it further to make the semantic parser more robust (e.g., by including UNION or allowing comparison of multiple previous answers). Practically, allowing a more complicated semantic parse structure—either by increasing the number of primitive statements or the length of the parse—poses serious computational challenges in both model learning and inference. Because of the dynamic nature of our framework, it is not trivial to leverage the computational capabilities of GPUs using minibatched training; we plan to investigate ways to take full advantage of modern computing machinery in the near future. Finally, better resolution of semantic matching errors is a top priority, and unsupervised learning from large external corpora is one way to make progress in this direction.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. We are also grateful to Panupong Pasupat for his help in configuring the floating parser baseline, and to Arvind Neelakantan for his help with the neural programmer model.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 1156–1165.

Stefan L Frank. 2013. Uncertainty reduction as a measure of cognitive load in sentence comprehension. *Topics in Cognitive Science* 5(3).

John Hale. 2006. Uncertainty about the rest of the sentence. *Cognitive Science* 30(4).

Sanda Harabagiu, Andrew Hickl, John Lehmann, and Dan Moldovan. 2005. Experiments with interactive question-answering. In *Proceedings of the Association for Computational Linguistics*.

Diane Kelly and Jimmy Lin. 2007. Overview of the trec 2006 ciqa task. In *ACM SIGIR Forum*. ACM, volume 41, pages 107–116.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition* 106(3).

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM* 59(9):68–76.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the Association for Computational Linguistics*.

Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. 2017. Deep learning with dynamic computation graphs. In *Proceedings of the International Conference on Learning Representations*.

Arvind Neelakantan, Quoc Le, Martin Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. In *Proceedings of the International Conference on Learning Representations*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the Association for Computational Linguistics*.

Haoruo Peng, Ming-Wei Chang, and Wen-tau Yih. 2017. Maximum margin reward networks for learning from explicit and implicit supervision. Manuscript Submitted for Publication.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin Markov networks. In *Proceedings of Advances in Neural Information Processing Systems*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of machine learning research* 6(Sep):1453–1484.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 1321–1331.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 201–206.

Luke Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Association for Computational Linguistics*.

A Action Neural Module Design

We describe here the neural module design for each action. As most actions try to match question text to column names or table entries, the neural network architectures are essentially various kinds of semantic similarity matching functions.

$\mathcal{A}_1$ **Select-column** Conceptually, the corresponding module is a combination of various matching scores. Let W_Q be the embeddings of words in the question and W_C be the embeddings of words in the target column name. The component matching functions are:

$$f_{max} = \frac{1}{|W_C|} \sum_{w_c \in W_C} \max_{w_q \in W_Q} w_q^T w_c$$

$$f_{avg} = \left(\frac{1}{|W_C|} \sum_{w_c \in W_C} w_c \right)^T \left(\frac{1}{|W_Q|} \sum_{w_q \in W_Q} w_q \right)$$

Essentially, for each word in the column name, f_{max} finds the highest matching question word and outputs the average score. Conversely, f_{avg} simply uses the average word vectors of the question and column name and returns their inner product. In another variant of f_{avg}, we replace the question representation with the output of a bidirectional LSTM model. These matching component functions are combined by a 2-layer feed-forward neural network, which outputs a scalar value as the action score.

$\mathcal{A}_2$ **Cond-column** Because this action also tries to find the correct column (but for conditions), we use the same matching scoring functions as in $\mathcal{A}_1$ module. However, a different 2-layer feed-forward neural network is used to combine the scores, as well as two binary features that indicate whether all the cells in this column are numeric values or not.

$\mathcal{A}_3$ **Op-Equal** This action checks whether a particular column value matches the question text. Suppose the average of the word vectors of the particular cell is w_x and the question word vectors are W_Q. Here the matching function is:

$$f_{max} = \max_{w_q \in W_Q} w_q^T w_x$$

$\mathcal{A}_4$ **Op-NotEqual** The neural module for this action extends the design for $\mathcal{A}_3$. It first uses a $\max$ function similar to f_{max} in $\mathcal{A}_3$ to compare the vector of the negation word "not", and the question words. This score is combined with the f_{max} score in $\mathcal{A}_3$ using a 2-layer feed-forward neural network as the final module score.

$\mathcal{A}_5$-$\mathcal{A}_8$ **Op-GT, Op-GE, Op-LT, Op-LE** The arguments of these comparison operations are extracted from question in advance. Therefore, the action modules just need to decide whether such relations are indeed used in the question. We take a simple strategy by initialing a special word vector that tries to capture the semantics of the relation. Take op-gt, *greater than*, for example. We use the average of the vectors of words like *more*, *greater* and *larger* to initialize the special word vector, denoted as w_{gt}. Let w_{arg} be the averaged vectors of words within a $[-2, +2]$ window centered at the argument in the question. The inner product of w_{gt} and w_{arg} is then used as the scoring function.

$\mathcal{A}_9$-$\mathcal{A}_{10}$ **Op-ArgMin, Op-ArgMax** We handle ArgMin and ArgMax similarly to the comparison operations. The difference is that we compare the special word vector to the averaged vector of all the question words, instead of a short subsequence of words.

Subsequent actions The modules in subsequent actions use basically the same design as their counterparts in the independent question setting. The main difference is that we extend the question representation to words from not just the target question, but also the question that immediately precedes it.

Gated-Attention Readers for Text Comprehension

Bhuwan Dhingra* **Hanxiao Liu*** **Zhilin Yang**
William W. Cohen **Ruslan Salakhutdinov**
School of Computer Science
Carnegie Mellon University
{bdhingra,hanxiaol,zhiliny,wcohen,rsalakhu}@cs.cmu.edu

Abstract

In this paper we study the problem of answering cloze-style questions over documents. Our model, the Gated-Attention (GA) Reader[1], integrates a multi-hop architecture with a novel attention mechanism, which is based on multiplicative interactions between the query embedding and the intermediate states of a recurrent neural network document reader. This enables the reader to build query-specific representations of tokens in the document for accurate answer selection. The GA Reader obtains state-of-the-art results on three benchmarks for this task–the CNN & Daily Mail news stories and the Who Did What dataset. The effectiveness of multiplicative interaction is demonstrated by an ablation study, and by comparing to alternative compositional operators for implementing the gated-attention.

1 Introduction

A recent trend to measure progress towards machine reading is to test a system's ability to answer questions about a document it has to comprehend. Towards this end, several large-scale datasets of cloze-style questions over a context document have been introduced recently, which allow the training of supervised machine learning systems (Hermann et al., 2015; Hill et al., 2016; Onishi et al., 2016). Such datasets can be easily constructed automatically and the unambiguous nature of their queries provides an objective benchmark to measure a system's performance at text comprehension.

Deep learning models have been shown to outperform traditional shallow approaches on text comprehension tasks (Hermann et al., 2015). The success of many recent models can be attributed primarily to two factors: (1) *Multi-hop architectures* (Weston et al., 2015; Sordoni et al., 2016; Shen et al., 2016), allow a model to scan the document and the question iteratively for multiple passes. (2) *Attention mechanisms*, (Chen et al., 2016; Hermann et al., 2015) borrowed from the machine translation literature (Bahdanau et al., 2014), allow the model to focus on appropriate subparts of the context document. Intuitively, the multi-hop architecture allows the reader to incrementally refine token representations, and the attention mechanism re-weights different parts in the document according to their relevance to the query.

The effectiveness of multi-hop reasoning and attentions have been explored orthogonally so far in the literature. In this paper, we focus on combining both in a complementary manner, by designing a novel attention mechanism which gates the evolving token representations across hops. More specifically, unlike existing models where the query attention is applied either token-wise (Hermann et al., 2015; Kadlec et al., 2016; Chen et al., 2016; Hill et al., 2016) or sentence-wise (Weston et al., 2015; Sukhbaatar et al., 2015) to allow weighted aggregation, the Gated-Attention (GA) module proposed in this work allows the query to directly interact with each dimension of the token embeddings at the semantic-level, and is applied layer-wise as information filters during the multi-hop representation learning process. Such a fine-grained attention enables our model to learn conditional token representations w.r.t. the given question, leading to accurate answer selections.

We show in our experiments that the proposed GA reader, despite its relative simplicity, consis-

*BD and HL contributed equally to this work.
[1]Source code is available on github: https://github.com/bdhingra/ga-reader

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1832–1846
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1168

tently improves over a variety of strong baselines on three benchmark datasets . Our key contribution, the GA module, provides a significant improvement for large datasets. Qualitatively, visualization of the attentions at intermediate layers of the GA reader shows that in each layer the GA reader attends to distinct salient aspects of the query which help in determining the answer.

2 Related Work

The cloze-style QA task involves tuples of the form $(d, q, a, \mathcal{C})$, where d is a document (context), q is a query over the contents of d, in which a phrase is replaced with a placeholder, and a is the answer to q, which comes from a set of candidates $\mathcal{C}$. In this work we consider datasets where each candidate $c \in \mathcal{C}$ has at least one token which also appears in the document. The task can then be described as: given a document-query pair (d, q), find $a \in \mathcal{C}$ which answers q. Below we provide an overview of representative neural network architectures which have been applied to this problem.

LSTMs with Attention: Several architectures introduced in Hermann et al. (2015) employ LSTM units to compute a combined document-query representation $g(d, q)$, which is used to rank the candidate answers. These include the **DeepLSTM Reader** which performs a single forward pass through the concatenated *(document, query)* pair to obtain $g(d, q)$; the **Attentive Reader** which first computes a document vector $d(q)$ by a weighted aggregation of words according to attentions based on q, and then combines $d(q)$ and q to obtain their joint representation $g(d(q), q)$; and the **Impatient Reader** where the document representation is built incrementally. The architecture of the Attentive Reader has been simplified recently in **Stanford Attentive Reader**, where shallower recurrent units were used with a bilinear form for the query-document attention (Chen et al., 2016).

Attention Sum: The **Attention-Sum (AS) Reader** (Kadlec et al., 2016) uses two bi-directional GRU networks (Cho et al., 2015) to encode both d and q into vectors. A probability distribution over the entities in d is obtained by computing dot products between q and the entity embeddings and taking a softmax. Then, an aggregation scheme named *pointer-sum attention* is further applied to sum the probabilities of the same entity, so that frequent entities the document will be favored compared to rare ones. Building on the AS Reader, the **Attention-over-Attention (AoA) Reader** (Cui et al., 2017) introduces a two-way attention mechanism where the query and the document are mutually attentive to each other.

Mulit-hop Architectures: **Memory Networks (MemNets)** were proposed in Weston et al. (2015), where each sentence in the document is encoded to a memory by aggregating nearby words. Attention over the memory slots given the query is used to compute an overall memory and to renew the query representation over multiple iterations, allowing certain types of reasoning over the salient facts in the memory and the query. **Neural Semantic Encoders (NSE)** (Munkhdalai & Yu, 2017a) extended MemNets by introducing a *write* operation which can evolve the memory over time during the course of reading. Iterative reasoning has been found effective in several more recent models, including the **Iterative Attentive Reader** (Sordoni et al., 2016) and **ReasoNet** (Shen et al., 2016). The latter allows dynamic reasoning steps and is trained with reinforcement learning.

Other related works include **Dynamic Entity Representation network (DER)** (Kobayashi et al., 2016), which builds dynamic representations of the candidate answers while reading the document, and accumulates the information about an entity by max-pooling; **EpiReader** (Trischler et al., 2016) consists of two networks, where one proposes a small set of candidate answers, and the other reranks the proposed candidates conditioned on the query and the context; **Bi-Directional Attention Flow network (BiDAF)** (Seo et al., 2017) adopts a multi-stage hierarchical architecture along with a flow-based attention mechanism; Bajgar et al. (2016) showed a 10% improvement on the CBT corpus (Hill et al., 2016) by training the AS Reader on an augmented training set of about 14 million examples, making a case for the community to exploit data abundance. The focus of this paper, however, is on designing models which exploit the available data efficiently.

3 Gated-Attention Reader

Our proposed GA readers perform multiple hops over the document (context), similar to the Memory Networks architecture (Sukhbaatar et al., 2015). Multi-hop architectures mimic the multi-step comprehension process of human readers, and have shown promising results in several recent models for text comprehension (Sordoni et al.,

2016; Kumar et al., 2016; Shen et al., 2016). The contextual representations in GA readers, namely the embeddings of words in the document, are iteratively refined across hops until reaching a final attention-sum module (Kadlec et al., 2016) which maps the contextual representations in the last hop to a probability distribution over candidate answers.

The attention mechanism has been introduced recently to model human focus, leading to significant improvement in machine translation and image captioning (Bahdanau et al., 2014; Mnih et al., 2014). In reading comprehension tasks, ideally, the semantic meanings carried by the contextual embeddings should be aware of the query across hops. As an example, human readers are able to keep the question in mind during multiple passes of reading, to successively mask away information irrelevant to the query. However, existing neural network readers are restricted to either attend to tokens (Hermann et al., 2015; Chen et al., 2016) or entire sentences (Weston et al., 2015), with the assumption that certain sub-parts of the document are more important than others. In contrast, we propose a finer-grained model which attends to components of the semantic representation being built up by the GRU. The new attention mechanism, called *gated-attention*, is implemented via *multiplicative* interactions between the query and the contextual embeddings, and is applied per hop to act as fine-grained information filters during the multi-step reasoning. The filters weigh individual components of the vector representation of *each* token in the document separately.

The design of gated-attention layers is motivated by the effectiveness of multiplicative interaction among vector-space representations, e.g., in various types of recurrent units (Hochreiter & Schmidhuber, 1997; Wu et al., 2016) and in relational learning (Yang et al., 2014; Kiros et al., 2014). While other types of compositional operators are possible, such as concatenation or addition (Mitchell & Lapata, 2008), we find that multiplication has strong empirical performance (section 4.3), where query representations naturally serve as information filters across hops.

3.1 Model Details

Several components of the model use a Gated Recurrent Unit (GRU) (Cho et al., 2015) which maps an input sequence $X = [x_1, x_2, \ldots, x_T]$ to an ouput sequence $H = [h_1, h_2, \ldots, h_T]$ as follows:

$$
\begin{aligned}
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r), \\
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z), \\
\tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h), \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t.
\end{aligned}
$$

where $\odot$ denotes the Hadamard product or the element-wise multiplication. r_t and z_t are called the *reset* and *update* gates respectively, and $\tilde{h}_t$ the *candidate output*. A Bi-directional GRU (Bi-GRU) processes the sequence in both forward and backward directions to produce two sequences $[h_1^f, h_2^f, \ldots, h_T^f]$ and $[h_1^b, h_2^b, \ldots, h_T^b]$, which are concatenated at the output

$$
\overleftrightarrow{\text{GRU}}(X) = [h_1^f \| h_T^b, \ldots, h_T^f \| h_1^b] \tag{1}
$$

where $\overleftrightarrow{\text{GRU}}(X)$ denotes the *full output* of the Bi-GRU obtained by concatenating each forward state h_i^f and backward state h_{T-i+1}^b at step i given the input X. Note $\overleftrightarrow{\text{GRU}}(X)$ is a matrix in $\mathbb{R}^{2n_h \times T}$ where n_h is the number of hidden units in GRU.

Let $X^{(0)} = [x_1^{(0)}, x_2^{(0)}, \ldots x_{|D|}^{(0)}]$ denote the token embeddings of the document, which are also inputs at layer 1 for the document reader below, and $Y = [y_1, y_2, \ldots y_{|Q|}]$ denote the token embeddings of the query. Here $|D|$ and $|Q|$ denote the document and query lengths respectively.

3.1.1 Multi-Hop Architecture

Fig. 1 illustrates the Gated-Attention (GA) reader. The model reads the document and the query over K horizontal layers, where layer k receives the contextual embeddings $X^{(k-1)}$ of the document from the previous layer. The document embeddings are transformed by taking the full output of a document Bi-GRU (indicated in blue in Fig. 1):

$$
D^{(k)} = \overleftrightarrow{\text{GRU}}_D^{(k)}(X^{(k-1)}) \tag{2}
$$

At the same time, a layer-specific query representation is computed as the full output of a separate query Bi-GRU (indicated in green in Figure 1):

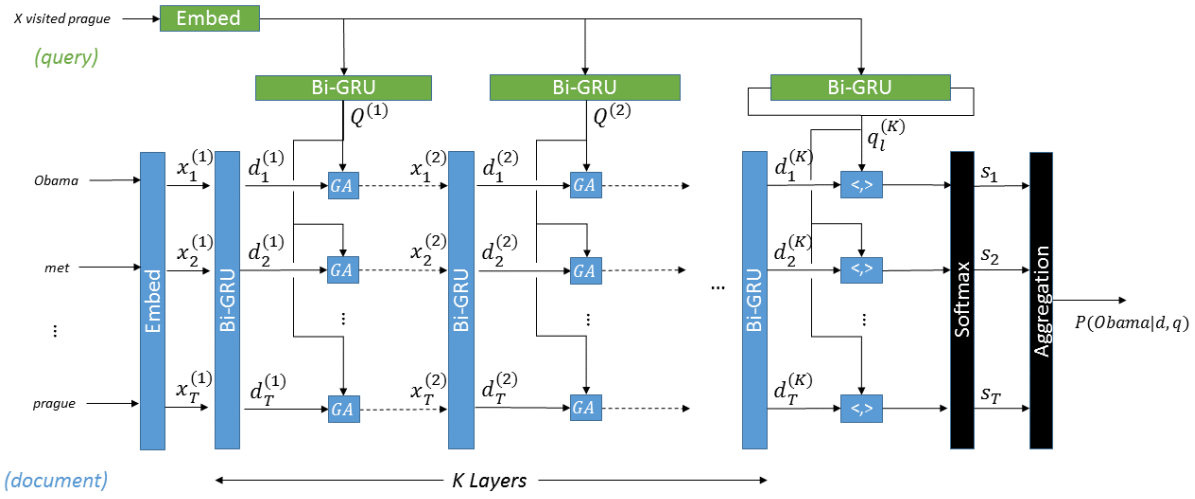
$$
Q^{(k)} = \overleftrightarrow{\text{GRU}}_Q^{(k)}(Y) \tag{3}
$$

Next, *Gated-Attention* is applied to $D^{(k)}$ and $Q^{(k)}$ to compute inputs for the next layer $X^{(k)}$.

$$
X^{(k)} = \text{GA}(D^{(k)}, Q^{(k)}) \tag{4}
$$

where GA is defined in the following subsection.

Figure 1: Gated-Attention Reader. Dashed lines represent dropout connections.



3.1.2 Gated-Attention Module

For brevity, let us drop the superscript k in this subsection as we are focusing on a particular layer. For each token d_i in D, the GA module forms a token-specific representation of the query $\tilde{q}_i$ using soft attention, and then multiplies the query representation element-wise with the document token representation. Specifically, for $i = 1, \ldots, |D|$:

$$\alpha_i = \text{softmax}(Q^\top d_i) \tag{5}$$
$$\tilde{q}_i = Q\alpha_i$$
$$x_i = d_i \odot \tilde{q}_i \tag{6}$$

In equation (6) we use the multiplication operator to model the interactions between d_i and $\tilde{q}_i$. In the experiments section, we also report results for other choices of gating functions, including addition $x_i = d_i + \tilde{q}_i$ and concatenation $x_i = d_i \| \tilde{q}_i$.

3.1.3 Answer Prediction

Let $q_\ell^{(K)} = q_\ell^f \| q_{T-\ell+1}^b$ be an intermediate output of the final layer query Bi-GRU at the location ℓ of the cloze token in the query, and $D^{(K)} = \overleftrightarrow{\text{GRU}}_D^{(K)}(X^{(K-1)})$ be the full output of final layer document Bi-GRU. To obtain the probability that a particular token in the document answers the query, we take an inner-product between these two, and pass through a softmax layer:

$$s = \text{softmax}((q_\ell^{(K)})^T D^{(K)}) \tag{7}$$

where vector s defines a probability distribution over the $|D|$ tokens in the document. The probability of a particular candidate $c \in \mathcal{C}$ as being the answer is then computed by aggregating the probabilities of all document tokens which appear in c and renormalizing over the candidates:

$$\Pr(c|d,q) \propto \sum_{i \in \mathbb{I}(c,d)} s_i \tag{8}$$

where $\mathbb{I}(c,d)$ is the set of positions where a token in c appears in the document d. This aggregation operation is the same as the *pointer sum attention* applied in the AS Reader (Kadlec et al., 2016).

Finally, the candidate with maximum probability is selected as the predicted answer:

$$a^* = \text{argmax}_{c \in \mathcal{C}} \ \Pr(c|d,q). \tag{9}$$

During the training phase, model parameters of GA are updated w.r.t. a cross-entropy loss between the predicted probabilities and the true answers.

3.1.4 Further Enhancements

Character-level Embeddings: Given a token w from the document or query, its vector space representation is computed as $x = L(w)||C(w)$. $L(w)$ retrieves the word-embedding for w from a lookup table $L \in \mathbb{R}^{|V| \times n_l}$, whose rows hold a vector for each unique token in the vocabulary. We also utilize a character composition model $C(w)$ which generates an orthographic embedding of the token. Such embeddings have been previously shown to be helpful for tasks like Named Entity Recognition (Yang et al., 2016) and dealing with OOV tokens at test time (Dhingra et al., 2016). The embedding $C(w)$ is generated by taking the final outputs $z_{n_c}^f$ and $z_{n_c}^b$ of a Bi-GRU applied to embeddings from

a lookup table of characters in the token, and applying a linear transformation:

$$z = z^f_{n_c} || z^b_{n_c}$$
$$C(w) = Wz + b$$

Question Evidence Common Word Feature (qe-comm): Li et al. (2016) recently proposed a simple token level indicator feature which significantly boosts reading comprehension performance in some cases. For each token in the document we construct a one-hot vector $f_i \in \{0, 1\}^2$ indicating its presence in the query. It can be incorporated into the GA reader by assigning a feature lookup table $F \in \mathbb{R}^{n_F \times 2}$ (we use $n_F = 2$), taking the feature embedding $e_i = f_i^T F$ and appending it to the inputs of the last layer document BiGRU as, $x_i^{(K)} \| f_i$ for all i. We conducted several experiments both with and without this feature and observed some interesting trends, which are discussed below. Henceforth, we refer to this feature as the *qe-comm feature* or just *feature*.

4 Experiments and Results

4.1 Datasets

We evaluate the GA reader on five large-scale datasets recently proposed in the literature. The first two, CNN and Daily Mail news stories[2] consist of articles from the popular CNN and Daily Mail websites (Hermann et al., 2015). A query over each article is formed by removing an entity from the short summary which follows the article. Further, entities within each article were anonymized to make the task purely a comprehension one. N-gram statistics, for instance, computed over the entire corpus are no longer useful in such an anonymized corpus.

The next two datasets are formed from two different subsets of the Children's Book Test (CBT)[3] (Hill et al., 2016). Documents consist of 20 contiguous sentences from the body of a popular children's book, and queries are formed by deleting a token from the 21st sentence. We only focus on subsets where the deleted token is either a common noun (CN) or named entity (NE) since simple language models already give human-level performance on the other types (cf. (Hill et al., 2016)).

The final dataset is Who Did What[4] (WDW) (Onishi et al., 2016), constructed from the LDC English Gigaword newswire corpus. First, article pairs which appeared around the same time and with overlapping entities are chosen, and then one article forms the document and a cloze query is constructed from the other. Missing tokens are always person named entities. Questions which are easily answered by simple baselines are filtered out, to make the task more challenging. There are two versions of the training set—a small but focused "Strict" version and a large but noisy "Relaxed" version. We report results on both settings which share the same validation and test sets. Statistics of all the datasets used in our experiments are summarized in the Appendix (Table 5).

4.2 Performance Comparison

Tables 1 and 3 show a comparison of the performance of GA Reader with previously published results on WDW and CNN, Daily Mail, CBT datasets respectively. The numbers reported for GA Reader are for single best models, though we compare to both ensembles and single models from prior work. GA Reader-- refers to an earlier version of the model, unpublished but described in a preprint, with the following differences—(1) it does not utilize token-specific attentions within the GA module, as described in equation (5), (2) it does not use a character composition model, (3) it is initialized with word embeddings pretrained on the corpus itself rather than GloVe. A detailed analysis of these differences is studied in the next section. Here we present 4 variants of the latest GA Reader, using combinations of whether the qe-comm feature is used (+feature) or not, and whether the word lookup table $L(w)$ is updated during training or fixed to its initial value. Other hyperparameters are listed in Appendix A.

Interestingly, we observe that feature engineering leads to significant improvements for WDW and CBT datasets, but not for CNN and Daily Mail datasets. We note that anonymization of the latter datasets means that there is already some feature engineering (it adds hints about whether a token is an entity), and these are much larger than the other four. In machine learning it is common to see the effect of feature engineering diminish with increasing data size. Similarly, fixing the word embeddings provides an improvement for the WDW

[2] https://github.com/deepmind/rc-data

[3] http://www.thespermwhale.com/jaseweston/babi/CBTest.tgz

[4] https://tticnlp.github.io/who_did_what/

Table 1: Validation/Test accuracy (%) on WDW dataset for both "Strict" and "Relaxed" settings. Results with "†" are cf previously published works.

Model	Strict		Relaxed	
	Val	Test	Val	Test
Human †	–	84	–	–
Attentive Reader †	–	53	–	55
AS Reader †	–	57	–	59
Stanford AR †	–	64	–	65
NSE †	66.5	66.2	67.0	66.7
GA-- †	–	57	–	60.0
GA (update $L(w)$)	67.8	67.0	67.0	66.6
GA (fix $L(w)$)	68.3	68.0	69.6	69.1
GA (+feature, update $L(w)$)	70.1	69.5	70.9	71.0
GA (+feature, fix $L(w)$)	**71.6**	**71.2**	**72.6**	**72.6**

Table 2: **Top:** Performance of different gating functions. **Bottom:** Effect of varying the number of hops K. Results on WDW without using the qe-comm feature and with fixed $L(w)$.

Gating Function	Accuracy	
	Val	Test
Sum	64.9	64.5
Concatenate	64.4	63.7
Multiply	**68.3**	**68.0**
K		
1 (AS) †	–	57
2	65.6	65.6
3	**68.3**	68.0
4	**68.3**	**68.2**

and CBT, but not for CNN and Daily Mail. This is not surprising given that the latter datasets are larger and less prone to overfitting.

Comparing with prior work, on the WDW dataset the basic version of the GA Reader outperforms all previously published models when trained on the Strict setting. By adding the qe-comm feature the performance increases by 3.2% and 3.5% on the Strict and Relaxed settings respectively to set a new state of the art on this dataset. On the CNN and Daily Mail datasets the GA Reader leads to an improvement of 3.2% and 4.3% respectively over the best previous single models. They also outperform previous ensemble models, setting a new state of that art for both datasets. For CBT-NE, GA Reader with the qe-comm feature outperforms all previous single and ensemble models except the AS Reader trained on the much larger BookTest Corpus (Bajgar et al., 2016). Lastly, on CBT-CN the GA Reader with the qe-comm feature outperforms all previously published single models except the NSE, and AS Reader trained on a larger corpus. For each of the 4 datasets on which GA achieves the top performance, we conducted one-sample proportion tests to test whether GA is significantly better than the second-best baseline. The p-values are 0.319 for CNN, <0.00001 for DailyMail, 0.028 for CBT-NE, and <0.00001 for WDW. In other words, GA statistically significantly outperforms all other baselines on 3 out of those 4 datasets at the 5% significance level. The results could be even more significant under paired tests, however we did not have access to the predictions from the baselines.

4.3 GA Reader Analysis

In this section we do an ablation study to see the effect of Gated Attention. We compare the GA Reader as described here to a model which is exactly the same in all aspects, except that it passes document embeddings $D^{(k)}$ in each layer directly to the inputs of the next layer without using the GA module. In other words $X^{(k)} = D^{(k)}$ for all $k > 0$. This model ends up using only one query GRU at the output layer for selecting the answer from the document. We compare these two variants both with and without the qe-comm feature on CNN and WDW datasets for three subsets of the training data - 50%, 75% and 100%. Test set accuracies for these settings are shown in Figure 2. On CNN when tested without feature engineering, we observe that GA provides a significant boost in performance compared to without GA. When tested with the feature it still gives an improvement, but the improvement is significant only with 100% training data. On WDW-Strict, which is a third of the size of CNN, without the feature we see an improvement when using GA versus without using GA, which becomes significant as the training set size increases. When tested with the feature on WDW, for a small data size without GA does better than with GA, but as the dataset size increases they become equivalent. We conclude that GA provides a boost in the absence of feature engineering, or as the training set size increases.

Next we look at the question of how to gate intermediate document reader states from the query, i.e. what operation to use in equation 6. Table

Table 3: Validation/Test accuracy (%) on CNN, Daily Mail and CBT. Results marked with "†" are cf previously published works. Results marked with "‡" were obtained by training on a larger training set. Best performance on standard training sets is in bold, and on larger training sets in italics.

Model	**CNN**		**Daily Mail**		**CBT-NE**		**CBT-CN**	
	Val	Test	Val	Test	Val	Test	Val	Test
Humans (query) †	–	–	–	–	–	52.0	–	64.4
Humans (context + query) †	–	–	–	–	–	81.6	–	81.6
LSTMs (context + query) †	–	–	–	–	51.2	41.8	62.6	56.0
Deep LSTM Reader †	55.0	57.0	63.3	62.2	–	–	–	–
Attentive Reader †	61.6	63.0	70.5	69.0	–	–	–	–
Impatient Reader †	61.8	63.8	69.0	68.0	–	–	–	–
MemNets †	63.4	66.8	–	–	70.4	66.6	64.2	63.0
AS Reader †	68.6	69.5	75.0	73.9	73.8	68.6	68.8	63.4
DER Network †	71.3	72.9	–	–	–	–	–	–
Stanford AR (relabeling) †	73.8	73.6	77.6	76.6	–	–	–	–
Iterative Attentive Reader †	72.6	73.3	–	–	75.2	68.6	72.1	69.2
EpiReader †	73.4	74.0	–	–	75.3	69.7	71.5	67.4
AoA Reader †	73.1	74.4	–	–	77.8	72.0	72.2	69.4
ReasoNet †	72.9	74.7	77.6	76.6	–	–	–	–
NSE †	–	–	–	–	78.2	73.2	74.3	**71.9**
BiDAF †	76.3	76.9	80.3	79.6	–	–	–	–
MemNets (ensemble) †	66.2	69.4	–	–	–	–	–	–
AS Reader (ensemble) †	73.9	75.4	78.7	77.7	76.2	71.0	71.1	68.9
Stanford AR (relabeling,ensemble) †	77.2	77.6	80.2	79.2	–	–	–	–
Iterative Attentive Reader (ensemble) †	75.2	76.1	–	–	76.9	72.0	74.1	71.0
EpiReader (ensemble) †	–	–	–	–	76.6	71.8	73.6	70.6
AS Reader (+BookTest) † ‡	–	–	–	–	80.5	76.2	83.2	80.8
AS Reader (+BookTest,ensemble) † ‡	–	–	–	–	*82.3*	*78.4*	*85.7*	*83.7*
GA--	73.0	73.8	76.7	75.7	74.9	69.0	69.0	63.9
GA (update $L(w)$)	**77.9**	**77.9**	**81.5**	**80.9**	76.7	70.1	69.8	67.3
GA (fix $L(w)$)	77.9	77.8	80.4	79.6	77.2	71.4	71.6	68.0
GA (+feature, update $L(w)$)	77.3	76.9	80.7	80.0	77.2	73.3	73.0	69.8
GA (+feature, fix $L(w)$)	76.7	77.4	80.0	79.3	**78.5**	**74.9**	**74.4**	70.7

2 (top) shows the performance on WDW dataset for three common choices – `sum` ($x = d + q$), `concatenate` ($x = d \| q$) and `multiply` ($x = d \odot q$). Empirically we find element-wise multiplication does significantly better than the other two, which justifies our motivation to "filter" out document features which are irrelevant to the query.

At the bottom of Table 2 we show the effect of varying the number of hops K of the GA Reader on the final performance. We note that for $K = 1$, our model is equivalent to the AS Reader without any GA modules. We see a steep and steady rise in accuracy as the number of hops is increased from $K = 1$ to 3, which remains constant beyond that. This is a common trend in machine learning as model complexity is increased, however we note that a multi-hop architecture is important to achieve a high performance for this task, and provide further evidence for this in the next section.

4.4 Ablation Study for Model Components

Table 4 shows accuracy on WDW by removing one component at a time. The steepest reduction is observed when we replace pretrained GloVe vectors with those pretrained on the corpus itself. GloVe vectors were trained on a large corpus of about 6 billion tokens (Pennington et al., 2014), and provide an important source of prior knowl-

Figure 2: Performance in accuracy with and without the Gated-Attention module over different training sizes. p-values for an exact one-sided Mcnemar's test are given inside the parentheses for each setting.

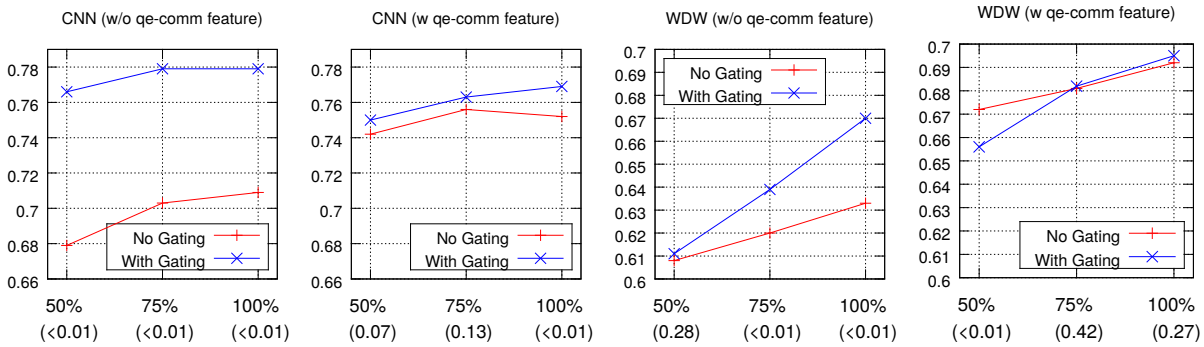


Table 4: Ablation study on WDW dataset, without using the qe-comm feature and with fixed $L(w)$. Results marked with † are cf Onishi et al. (2016).

Model	Accuracy	
	Val	Test
GA	**68.3**	**68.0**
−char	66.9	66.9
−token-attentions (eq. 5)	65.7	65.0
−glove, +corpus	64.0	62.5
GA--†	–	57

edge for the model. Note that the strongest baseline on WDW, NSE (Munkhdalai & Yu, 2017b), also uses pretrained GloVe vectors, hence the comparison is fair in that respect. Next, we observe a substantial drop when removing token-specific attentions over the query in the GA module, which allow gating individual tokens in the document only by parts of the query relevant to that token rather than the overall query representation. Finally, removing the character embeddings, which were only used for WDW and CBT, leads to a reduction of about 1% in the performance.

4.5 Attention Visualization

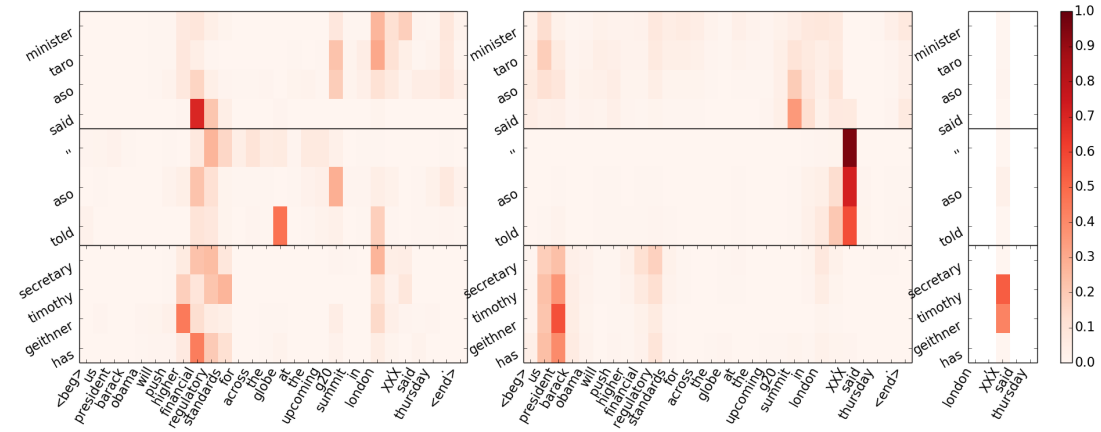
To gain an insight into the reading process employed by the model we analyzed the attention distributions at intermediate layers of the reader. Figure 3 shows an example from the validation set of WDW dataset (several more are in the Appendix). In each figure, the left and middle plots visualize attention over the query (equation 5) for candidates in the document after layers 1 & 2 respectively. The right plot shows attention over candidates in the document of cloze placeholder (XXX) in the query at the final layer. The full document, query and correct answer are shown at the bottom.

A generic pattern observed in these examples is that in intermediate layers, candidates in the document (shown along rows) tend to pick out salient tokens in the query which provide clues about the cloze, and in the final layer the candidate with the highest match with these tokens is selected as the answer. In Figure 3 there is a high attention of the correct answer on `financial regulatory standards` in the first layer, and on `us president` in the second layer. The incorrect answer, in contrast, only attends to one of these aspects, and hence receives a lower score in the final layer despite the n-gram overlap it has with the cloze token in the query. Importantly, different layers tend to focus on different tokens in the query, supporting the hypothesis that the multi-hop architecture of GA Reader is able to combine distinct pieces of information to answer the query.

5 Conclusion

We presented the Gated-Attention reader for answering cloze-style questions over documents. The GA reader features a novel multiplicative gating mechanism, combined with a multi-hop architecture. Our model achieves the state-of-the-art performance on several large-scale benchmark datasets with more than 4% improvements over competitive baselines. Our model design is backed up by an ablation study showing statistically significant improvements of using Gated Attention as information filters. We also showed empirically that multiplicative gating is superior to addi-

Figure 3: Layer-wise attention visualization of GA Reader trained on WDW-Strict. See text for details.



DOC: japanese prime minister taro aso said friday he would call for stronger monitoring of international finance at the g20 summit next week in london . `` we will h ave to emphatically argue that the foundation of the international monetary fund (imf) is weak and that we must establish financial regulations and supervisio n , '' aso told a legislative session . other world leaders have also pushed for stricter regulations of risky and unrestrained investment practices and instrum ents blamed for triggering the current global economic crisis . japan officially agreed in february to lend up to 100 billion dollars to the imf to provide fina ncial lifelines to emerging economies hit hard by the worldwide downturn . us treasury secretary timothy geithner has said president barack obama would discuss new global financial regulatory standards at the london summit .

QRY: <beg> us president barack obama will push higher financial regulatory standards for across the globe at the upcoming g20 summit in london , XXX said thursday . <end>
ANS: timothy geithner

tion and concatenation operations for implementing gated-attentions, though a theoretical justification remains part of future research goals. Analysis of document and query attentions in intermediate layers of the reader further reveals that the model iteratively attends to different aspects of the query to arrive at the final answer. In this paper we have focused on text comprehension, but we believe that the Gated-Attention mechanism may benefit other tasks as well where multiple sources of information interact.

Acknowledgments

This work was funded by NSF under CCF1414030 and Google Research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Ondrej Bajgar, Rudolf Kadlec, and Jan Kleindienst. Embracing data abundance: Booktest dataset for reading comprehension. *arXiv preprint arXiv:1610.00956*, 2016.

Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *ACL*, 2016.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *ACL*, 2015.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *ACL*, 2017.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. Tweet2vec: Character-based distributed representations for social media. *ACL*, 2016.

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

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *ICLR*, 2016.

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

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. *ACL*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

Ryan Kiros, Richard Zemel, and Ruslan R Salakhutdinov. A multiplicative model for learning distributed text-based attribute representations. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2014.

Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. Dynamic entity representations with max-pooling improves machine reading. In *NAACL-HLT*, 2016.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *ICML*, 2016.

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *arXiv preprint arXiv:1607.06275*, 2016.

Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, pp. 236–244, 2008.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pp. 2204–2212, 2014.

Tsendsuren Munkhdalai and Hong Yu. Neural semantic encoders. *EACL*, 2017a.

Tsendsuren Munkhdalai and Hong Yu. Reasoning with memory augmented neural networks for language comprehension. *ICLR*, 2017b.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Who did what: A large-scale person-centered cloze dataset. *EMNLP*, 2016.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *ICLR*, 2017.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. *arXiv preprint arXiv:1609.05284*, 2016.

Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245*, 2016.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pp. 2431–2439, 2015.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL http://arxiv.org/abs/1605.02688.

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. Natural language comprehension with the epireader. *EMNLP*, 2016.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *ICLR*, 2015.

Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan Salakhutdinov. On multiplicative integration with recurrent neural networks. *Advances in Neural Information Processing Systems*, 2016.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Learning multi-relational semantics using neural-embedding models. *NIPS Workshop on Learning Semantics*, 2014.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*, 2016.

A Implementation Details

Our model was implemented using the Theano (Theano Development Team, 2016) and Lasagne[5] Python libraries. We used stochastic gradient descent with ADAM updates for optimization, which combines classical momentum and adaptive gradients (Kingma & Ba, 2015). The batch size was 32 and the initial learning rate was 5×10^{-4} which was halved every epoch after the second epoch. The same setting is applied to all models and datasets. We also used gradient clipping with a threshold of 10 to stabilize GRU training (Pascanu et al., 2013). We set the number of layers K to be 3 for all experiments. The number of hidden units for the character GRU was set to 50. The remaining two hyperparameters—size of document and query GRUs, and dropout rate—were tuned on the validation set, and their optimal values are shown in Table 6. In general, the optimal GRU size increases and the dropout rate decreases as the corpus size increases.

The word lookup table was initialized with $100d$ GloVe vectors[6] (Pennington et al., 2014) and OOV tokens at test time were assigned unique random vectors. We empirically observed that initializing with pre-trained embeddings gives higher performance compared to random initialization for all

[5] https://lasagne.readthedocs.io/en/latest/

[6] http://nlp.stanford.edu/projects/glove/

Table 5: Dataset statistics.

	CNN	Daily Mail	CBT-NE	CBT-CN	WDW-Strict	WDW-Relaxed
# train	380,298	879,450	108,719	120,769	127,786	185,978
# validation	3,924	64,835	2,000	2,000	10,000	10,000
# test	3,198	53,182	2,500	2,500	10,000	10,000
# vocab	118,497	208,045	53,063	53,185	347,406	308,602
max doc length	2,000	2,000	1,338	1,338	3,085	3,085

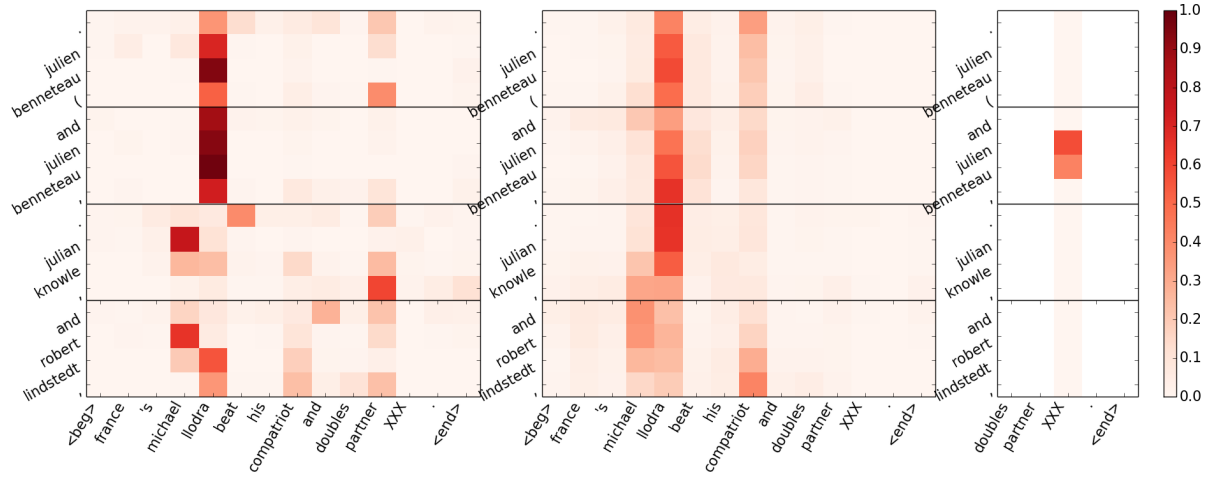
Table 6: Hyperparameter settings for each dataset. dim() indicates hidden state size of GRU.

Hyperparameter	CNN	Daily Mail	CBT-NE	CBT-CN	WDW-Strict	WDW-Relaxed
Dropout	0.2	0.1	0.4	0.4	0.3	0.3
$\dim(\overleftrightarrow{\text{GRU}}_*)$	256	256	128	128	128	128

datasets. Furthermore, for smaller datasets (WDW and CBT) we found that fixing these embeddings to their pretrained values led to higher test performance, possibly since it avoids overfitting. We do not use the character composition model for CNN and Daily Mail, since their entities (and hence candidate answers) are anonymized to generic tokens. For other datasets the character lookup table was randomly initialized with $25d$ vectors. All other parameters were initialized to their default values as specified in the Lasagne library.

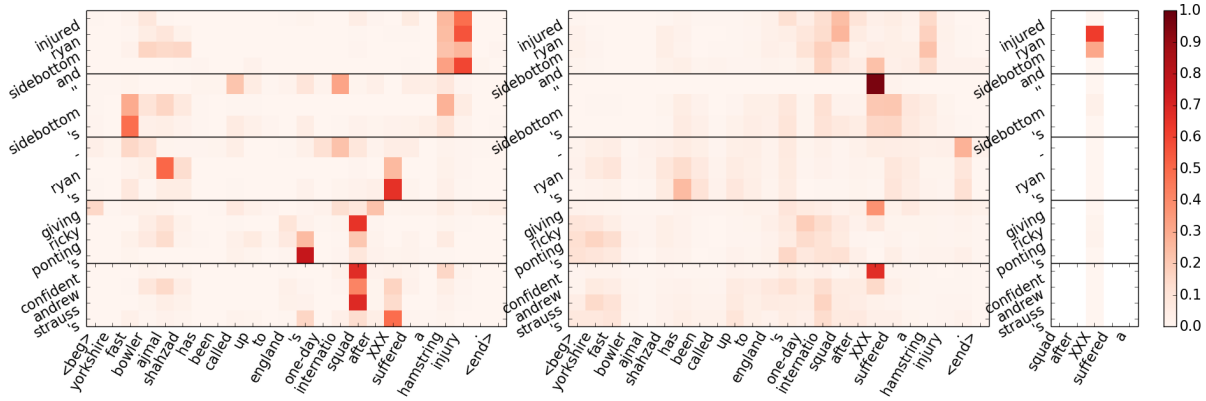
B Attention Plots

Figure 4: Layer-wise attention visualization of GA Reader trained on WDW-Strict. See text for details.



DOC: result sunday from the open 13 , a (euro) 512,750 ($ 697,400) atp world tour indoor hardcourt event at palais des sports (seedings in parentheses) : singl
es final michael llodra , france , def . julien benneteau (8) , france , 6-3 , 6-4. doubles final michael llodra and julien benneteau , france (2) , def . j
ulian knowle , austria and robert lindstedt , sweden (1) , 6-4 , 6-3 .

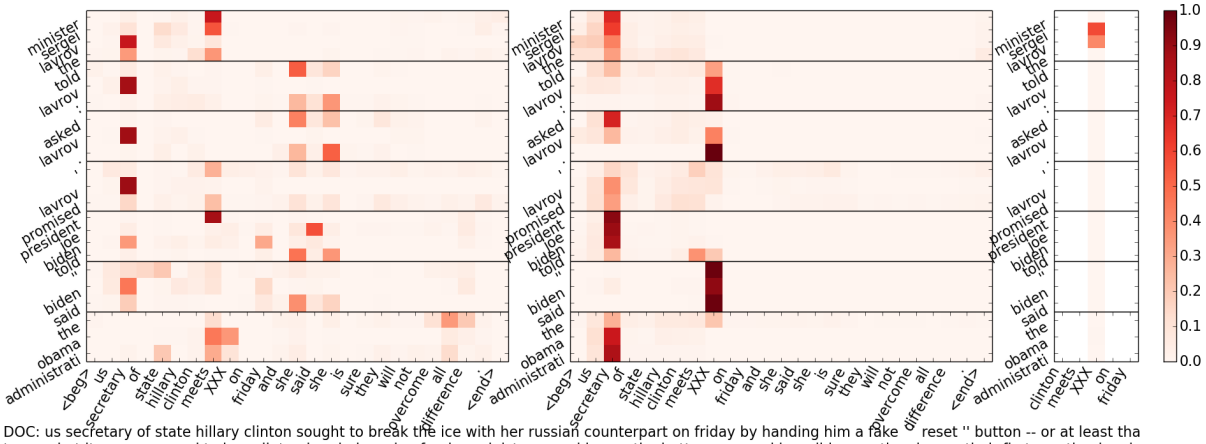
QRY: <beg> france 's michael llodra beat his compatriot and doubles partner XXX . <end>
ANS: julien benneteau



DOC: england pace bowler ajmal shahzad has warned australia that his `` fearless '' team- mates are ready to deal their rivals a psychological blow in the forthcomin
g one-day series . shahzad has been called into england 's squad for five matches against australia as cover for the injured ryan sidebottom and he senses an ex
tremely positive mood in the dressing room . the 24-year-old admits england are desperate to warm up for the ashes in australia later this year by giving ricky
ponting 's men another beating after defeating them in the recent icc world twenty20 final in the caribbean . and shahzad , the first british-born asian to play
for yorkshire , is confident andrew strauss 's side wo n't back down against the ultra-aggressive australians . `` it will be a step up for the lads . but ever
ybody is focused and ready for it , '' shahzad said . `` there 's no fear , no nerves - and it 's nice to be part of that kind of dressing room . '' sidebottom
's hamstring problem has given shahzad his opportunity in the one-day squad and he is determined to seize the opportunity to stake his claim for a permanent pla
ce by impressing against the australians . `` it 's just nice to be called up , and i hope if my chance comes i can grasp it with both hands , '' he said . `` a
lot 's happened to me in the last six months . if the chance comes - ryan 's got a niggle , but i do n't think it 's too bad - i 've just got to put in a decen
t performance and have my name in the hat for the games to come . `` but it 's just nice to be involved , and know you 're there or thereabouts . i just hope i
can get the nod . ''

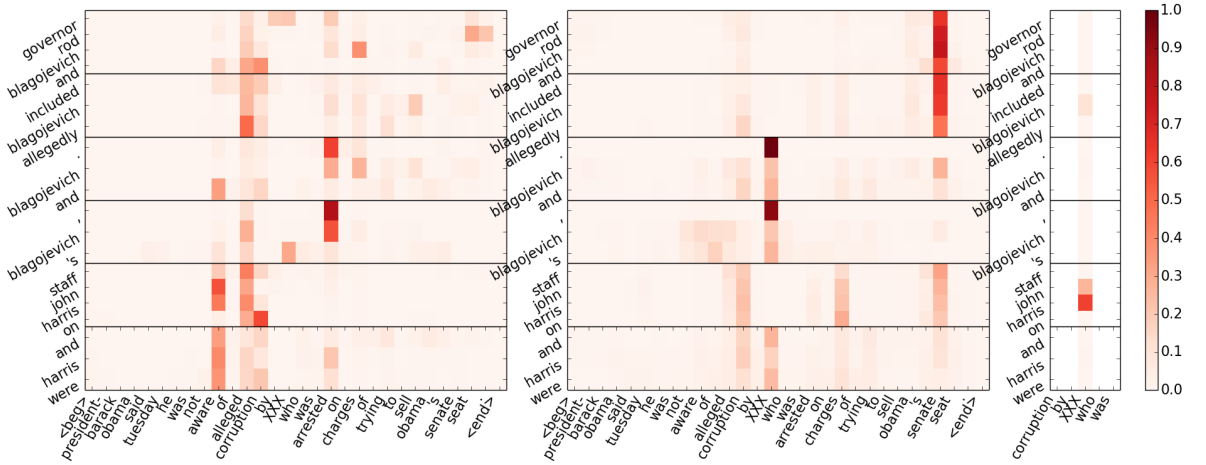
QRY: <beg> yorkshire fast bowler ajmal shahzad has been called up to england 's one-day international squad after XXX suffered a hamstring injury . <end>
ANS: ryan sidebottom

Figure 5: Layer-wise attention visualization of GA Reader trained on WDW-Strict. See text for details.



DOC: us secretary of state hillary clinton sought to break the ice with her russian counterpart on friday by handing him a fake reset '' button -- or at least tha t was what it was supposed to be . clinton handed russian foreign minister sergei lavrov the button wrapped in a ribbon as they began their first meeting in a l uxury hotel in geneva . earlier this year , us vice president joe biden told foreign leaders at an international security conference in munich , southern german y , that the obama administration wanted to improve ties with moscow . `` it is time to press the reset button and to revisit the many areas where we can and sh ould work together , '' biden said . as she proffered the red plastic button , clinton told lavrov : `` we want to reset our relationship . and so we will do it together , '' she said , laughing . but the button also bore a russian word that was meant to translate as `` reset '' . `` we worked hard to get the right rus sian word . do you think we got it ? '' clinton asked lavrov . `` you got it wrong , '' he responded as they both laughed . `` it should be 'perezagrouzka ' (t he russian word for 'reset ') , '' the russian foreign minister pointed out . `` this says , peregruzka , which means overcharged . '' `` we wo n't let you do that to us , '' clinton replied . russian speakers indicated that the mistaken word was better translated as 'overload . ' lavrov promised to keep it on his des k .

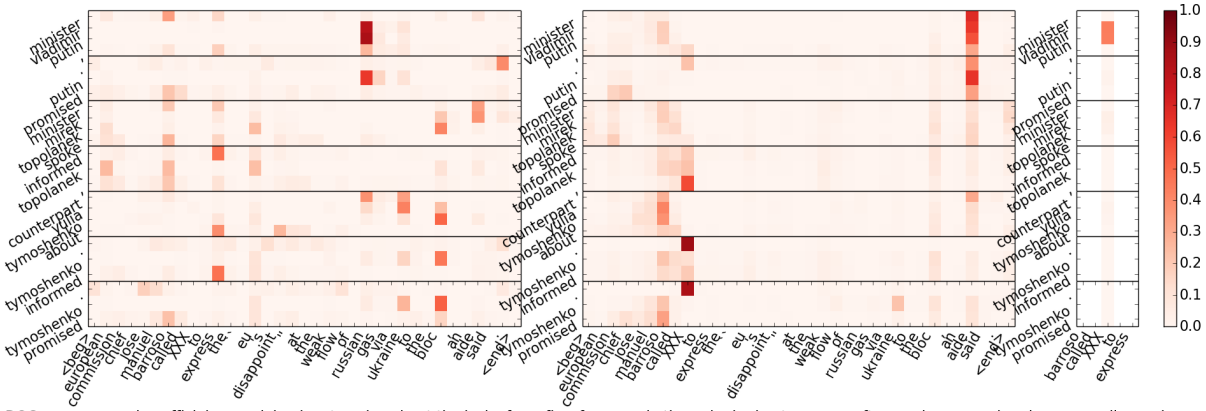
QRY: <beg> us secretary of state hillary clinton meets XXX on friday and she said she is sure they will not overcome all differences . <end>
ANS: sergei lavrov



DOC: illinois governor arrested on corruption charges chicago , dec. 9 (xinhua) -- u.s. federal prosecutors on tuesday arrested illinois governor rod blagojevich a nd his chief of staff john harris on corruption charges . the two were accused of a wide-ranging criminal conspiracy that included blagojevich allegedly conspir ing to sell or trade the senate seat left vacant by president-elect barack obama in exchange for financial benefits for his wife and himself . the governor was also charged with obtaining campaign contributions in exchange for other official actions . blagojevich and harris were arrested simultaneously at their homes a t about 6:15 a.m. , according to the fbi . they were transported to fbi headquarters in chicago , where they remained at 9 a.m. however , blagojevich 's spokesm an said he did not know the development .

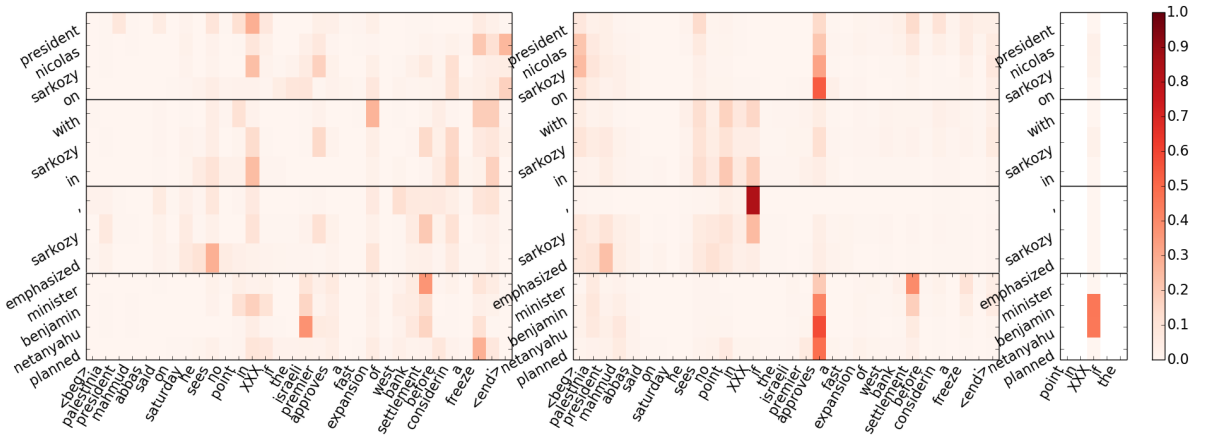
QRY: <beg> president-elect barack obama said tuesday he was not aware of alleged corruption by XXX who was arrested on charges of trying to sell obama 's senate seat . <end>
ANS: rod blagojevich

Figure 6: Layer-wise attention visualization of GA Reader trained on WDW-Strict. See text for details.



DOC: european union officials complained on tuesday about the lack of gas flow from russia through ukraine to europe after russia resumed early gas supplies under a three- way deal signed on the previous day . european commission president jose manuel barroso spoke by phone to russian prime minister vladimir putin , express ing disappointment over the lack of natural gas flowing to europe . eu monitors on the ground reported that only very little gas is flowing through the pipeline s . barroso voiced his `` disappointment with both the level of gas flowing to europe '' and the lack of access `` of our monitors to dispatch centers , '' acco rding to his aide . putin promised him to take a look into what he complained . on the same day , czech prime minister mirek topolanek spoke on the phone to his ukrainian counterpart yulia tymoshenko about the matter , said a press release from the czech eu presidency . tymoshenko informed topolanek , who asked about t he causes and circumstances of the delay in supplies , of some technical difficulties , saying that more specifically the pressure of gas arriving from the russ ia is too low . the czech prime minister recommended her to immediately contact the eurogaz experts who are ready to assist ukraine with technical problems . ty moshenko promised to act on this offer . russia reopened taps tuesday morning to let gas flow to europe via ukraine after cutting off gas supplies to europe on wednesday amid a pricing dispute with ukraine . the cutoff left a number of european countries in lack of heating gas amid freezing weather .

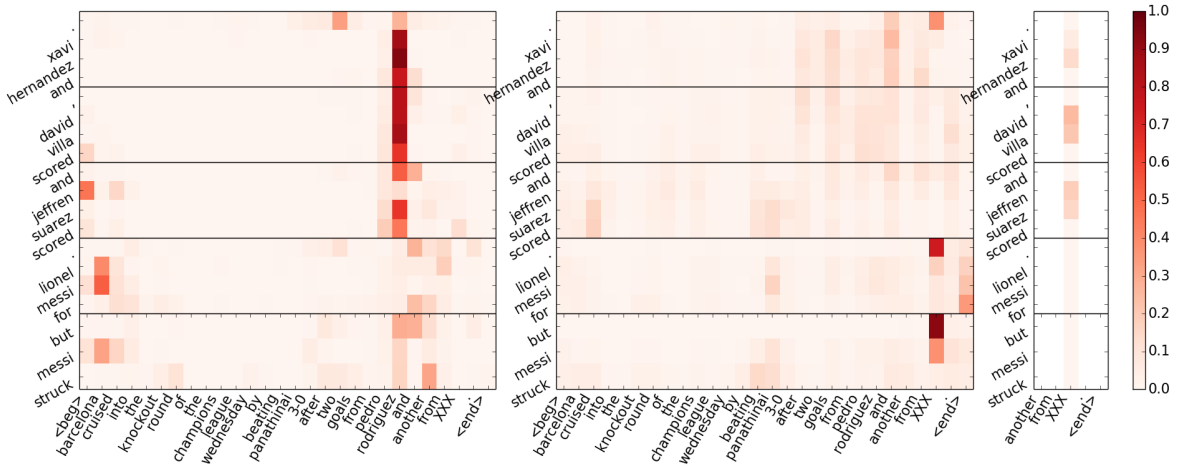
QRY: <beg> european commission chief jose manuel barroso called XXX to express the `` eu 's disappointment '' at the weak flow of russian gas via ukraine to the bloc , an aide said . <end>
ANS: vladimir putin



DOC: president of the palestinian authority mahmud abbas held talks with french president nicolas sarkozy on friday , and slammed the israeli plan of constructing mo re settlement buildings on the west bank as `` unacceptable . '' `` that is not acceptable , '' said abbas after meeting with sarkozy in the elysee palace . it was reported that israeli prime minister benjamin netanyahu planned to approval the construction of new home buildings on the west bank before considering a fre eze on settlement activities . `` we want a freeze on settlement and the launch of negotiations on the final phase of it , '' abbas said . `` this was the main subject of our talks . '' according to a statement from the elysee palace , talks between the two leaders were aimed at starting again the peace process within the palestinian territories , as well as discussing regional issues . during the meeting , sarkozy emphasized the urgency of the resumption of a negotiation pro cess between israel and palestine . this has been abbas ' third official visit to france since 2007 .

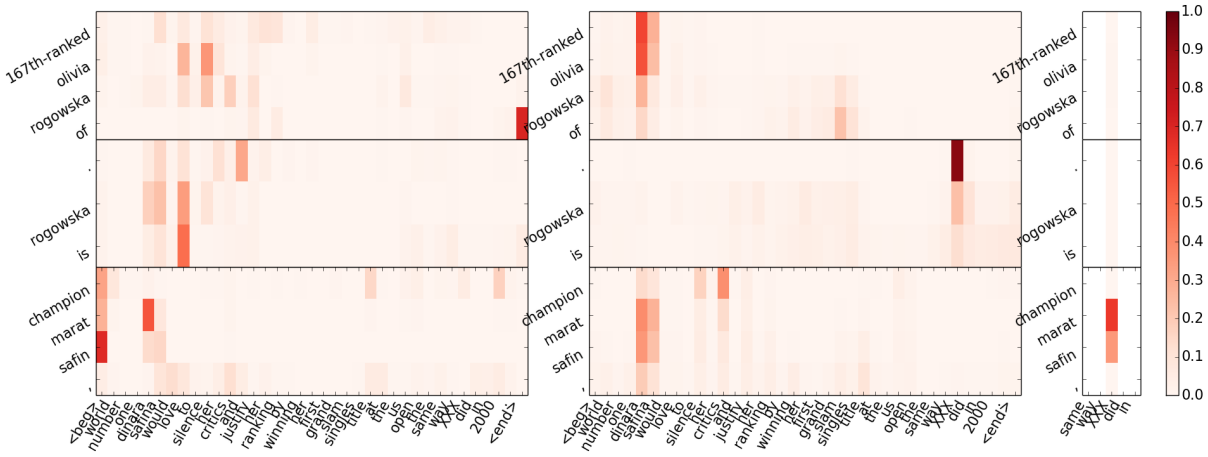
QRY: <beg> palestinian president mahmud abbas said on saturday he sees no point in XXX if the israeli premier approves a fast expansion of west bank settlements befo re considering a freeze . <end>
ANS: benjamin netanyahu

Figure 7: Layer-wise attention visualization of GA Reader trained on WDW-Strict. See text for details.



DOC: if there was a shred of doubt where the world cup was built and won for spain this year , it was removed monday night when barcelona destroyed real madrid , 5-0 . in teeming rain , host barcelona simply outplayed real , which led the spanish league until monday . barcelona 's lineup contained eight home-bred players , s even of them world champions . xavi hernandez and pedro rodriguez scored , david villa scored twice , and jeffren suarez scored the fifth as a substitute . the loss ended madrid 's 26-game unbeaten streak . lionel messi for once did not score . but messi struck a post , and he was involved in three of the goals . indee d , by taking a deeper role and taking considerable brutish tackles , he epitomized barcelona 's collective will to work for one another .

QRY: <beg> barcelona cruised into the knockout round of the champions league wednesday by beating panathinaikos 3-0 after two goals from pedro rodriguez and another from XXX . <end>
ANS: lionel messi



DOC: dinara safina has barely avoided becoming the first no . 1-seeded woman to lose in the first round at the u.s. open . safina overcame 11 double-faults and 48 un forced errors to come back and beat 167th-ranked olivia rogowska of australia 6-7 (5) , 6-2 , 6-4 in arthur ashe stadium on tuesday . safina , the younger sis ter of two-time major champion marat safin , moved up to no . 1 in the rankings in april -- and is assured of staying there no matter what happens at flushing m eadows . the russian reached the finals at the australian open and french open this year , losing both . rogowska is 18 , received a wild-card invitation into t he u.s. open and has won one grand slam match . she never has defeated anyone ranked better than 47th .

QRY: <beg> world number one dinara safina would love to silence her critics and justify her ranking by winning her first grand slam singles title at the us open the same way XXX did in 2000 . <end>
ANS: marat safin

Determining Gains Acquired from Word Embedding Quantitatively Using Discrete Distribution Clustering

Jianbo Ye[⋆], Yanran Li[†], Zhaohui Wu[‡], James Z. Wang[⋆], Wenjie Li[†] and Jia Li[⋆]
[⋆]The Pennsylvania State University, University Park, Pennsylvania
[†]The Hong Kong Polytechnic University, Hong Kong [‡]Microsoft

Abstract

Word embeddings have become widely-used in document analysis. While a large number of models for mapping words to vector spaces have been developed, it remains undetermined how much net gain can be achieved over traditional approaches based on bag-of-words. In this paper, we propose a new document clustering approach by combining any word embedding with a state-of-the-art algorithm for clustering empirical distributions. By using the Wasserstein distance between distributions, the word-to-word semantic relationship is taken into account in a principled way. The new clustering method is easy to use and consistently outperforms other methods on a variety of data sets. More importantly, the method provides an effective framework for determining when and how much word embeddings contribute to document analysis. Experimental results with multiple embedding models are reported.

1 Introduction

Word embeddings (a.k.a. word vectors) have been broadly adopted for document analysis (Mikolov et al., 2013a,b). The embeddings can be trained from external large-scale corpus and then easily utilized for different data. To a certain degree, the knowledge mined from the corpus, possibly in very intricate ways, is coded in the vector space, the samples of which are easy to describe and ready for mathematical modeling. Despite the appeal, researchers will be interested in knowing how much gain an embedding can bring forth over the performance achievable by existing bag-of-words based approaches. Moreover, how can the gain be quantified? Such a preliminary evaluation will be carried out before building a sophisticated pipeline of analysis.

Almost every document analysis model used in practice is constructed assuming a certain basic representation—bag-of-words or word embeddings—for the sake of computational tractability. For example, after word embedding is done, high-level models in the embedded space, such as entity representations, similarity measures, data manifolds, hierarchical structures, language models, and neural architectures, are designed for various tasks. In order to invent or enhance analysis tools, we want to understand precisely the pros and cons of the high-level models and the underlying representations. Because the model and the representation are tightly coupled in an analytical system, it is not easy to pinpoint where the gain or loss found in practice comes from. Should the gain be credited to the mechanism of the model or to the use of word embeddings? As our experiments demonstrate, introducing certain assumptions will make individual methods effective only if certain constraints are met. We will address this issue under an unsupervised learning framework.

Our proposed clustering paradigm has several advantages. Instead of packing the information of a document into a fixed-length vector for subsequent analysis, we treat a document more thoroughly as a distributional entity. In our approach, the distance between two empirical

Correspondence should be sent to J. Ye (jxy198@psu.edu) and J. Li (jiali@psu.edu).
The work was done when Z. Wu was with Penn State.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1847–1856
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1169

nonparametric measures (or discrete distributions) over the word embedding space is defined as the Wasserstein metric (a.k.a. the Earth Mover's Distance or EMD) (Wan, 2007; Kusner et al., 2015). Comparing with a vector representation, an empirical distribution can represent with higher fidelity a cloud of points such as words in a document mapped to a certain space. In the extreme case, the empirical distribution can be set directly as the cloud of points. In contrast, a vector representation reduces data significantly, and its effectiveness relies on the assumption that the discarded information is irrelevant or nonessential to later analysis. This simplification itself can cause degradation in performance, obscuring the inherent power of the word embedding space.

Our approach is intuitive and robust. In addition to a high fidelity representation of the data, the Wasserstein distance takes into account the cross-term relationship between different words in a *principled* fashion. According to the definition, the distance between two documents A and B are the minimum cumulative cost that words from document A need to "travel" to match exactly the set of words for document B. Here, the travel cost of a path between two words is their (squared) Euclidean distance in the word embedding space. Therefore, how much benefit the Wasserstein distance brings also depends on how well the word embedding space captures the semantic difference between words.

While Wasserstein distance is well suited for document analysis, a major obstacle of approaches based on this distance is the computational intensity, especially for the original D2-clustering method (Li and Wang, 2008). The main technical hurdle is to compute efficiently the Wasserstein barycenter, which is itself a discrete distribution, for a given set of discrete distributions. Thanks to the recent advances in the algorithms for solving Wasserstein barycenters (Cuturi and Doucet, 2014; Ye and Li, 2014; Benamou et al., 2015; Ye et al., 2017), one can now perform document clustering by directly treating them as empirical measures over a word embedding space. Although the computational cost is still higher than the usual vector-based clustering methods, we believe that the new clustering approach has reached a level of efficiency to justify its usage given how important it is to obtain high-quality clustering of unstructured text data. For instance, clustering is a crucial step performed ahead of cross-document co-reference resolution (Singh et al., 2011), document summarization, retrospective events detection, and opinion mining (Zhai et al., 2011).

1.1 Contributions

Our work has two main contributions. First, we create a basic tool of document clustering, which is easy to use and scalable. The new method leverages the latest numerical toolbox developed for optimal transport. It achieves state-of-the-art clustering performance across heterogeneous text data—an advantage over other methods in the literature. Second, the method enables us to quantitatively inspect how well a word-embedding model can fit the data and how much gain it can produce over the bag-of-words models.

2 Related Work

In the original D2-clustering framework proposed by Li and Wang (2008), calculating Wasserstein barycenter involves solving a large-scale LP problem at each inner iteration, severely limiting the scalability and robustness of the framework. Such high magnitude of computations had prohibited it from deploying in many real-world applications until recently. To accelerate the computation of Wasserstein barycenter, and ultimately to improve D2-clustering, multiple numerical algorithmic efforts have been made in the recent few years (Cuturi and Doucet, 2014; Ye and Li, 2014; Benamou et al., 2015; Ye et al., 2017).

Although the effectiveness of Wasserstein distance has been well recognized in the computer vision and multimedia literature, the property of Wasserstein barycenter has not been well understood. To our knowledge, there still lacks systematic study of applying Wasserstein barycenter and D2-clustering in document analysis with word embeddings.

A closely related work by Kusner et al. (2015) connects the Wasserstein distance to the word embeddings for comparing documents. Our work differs from theirs in the methodology. We directly pursue a scalable clustering setting rather than construct a nearest neighbor graph based on calculated distances, because the calculation of the Wasserstein distances of all pairs is too expensive to be practical. Kusner et al. (2015) used a lower bound that was less costly to compute in order to prune unnecessary full distance calculation, but

the scalability of this modified approach is still limited, an issue to be discussed in Section 4.3. On the other hand, our approach adopts the framework similar to the K-means which is of complexity $O(n)$ per iteration and usually converges within just tens of iterations. The computation of D2-clustering, though in its original form was magnitudes heavier than typical document clustering methods, can now be efficiently carried out with parallelization and proper implementations (Ye et al., 2017).

3 The Method

This section introduces the distance, the D2-clustering technique, the fast computation framework, and how they are used in the proposed document clustering method.

3.1 Wasserstein Distance

Suppose we represent each document d_k consisting m_k unique words by a discrete measure or a discrete distribution, where $k = 1, \ldots, N$ with N being the sample size:

$$d_k = \sum_{i=1}^{m_k} w_i^{(k)} \delta_{x_i^{(k)}} \ . \tag{1}$$

Here δ_x denotes the Dirac measure with support x, and $w_i^{(k)} \geq 0$ is the "importance weight" for the i-th word in the k-th document, with $\sum_{i=1}^{m_k} w_i^{(k)} = 1$. And $x_i^{(k)} \in \mathbb{R}^d$, called a support point, is the semantic embedding vector of the i-th word. The 2nd-order Wasserstein distance between two documents d_1 and d_2 (and likewise for any document pairs) is defined by the following LP problem: $W^2(d_1, d_2) :=$

$$\begin{aligned} \min_{\Pi} \quad & \textstyle\sum_{i,j} \pi_{i,j} \|x_i^{(1)} - x_j^{(2)}\|_2^2 \\ \text{s.t.} \quad & \textstyle\sum_{j=1}^{m_2} \pi_{i,j} = w_i, \forall i, \quad \sum_{i=1}^{m_1} \pi_{i,j} = w_j, \forall j \\ & \pi_{i,j} \geq 0, \forall i, j \ , \end{aligned} \tag{2}$$

where $\Pi = \{\pi_{i,j}\}$ is a $m_1 \times m_2$ coupling matrix, and let $\{C_{i,j} := \|x_i^{(1)} - x_j^{(2)}\|_2^2\}$ be transportation costs between words. Wasserstein distance is a true metric (Villani, 2003) for measures, and its best exact algorithm has a complexity of $O(m^3 \log m)$ (Orlin, 1993), if $m_1 = m_2 = m$.

3.2 Discrete Distribution (D2-) Clustering

D2-clustering (Li and Wang, 2008) iterates between the assignment step and centroids updating step in a similar way as the Lloyd's K-means. Suppose we are to find K clusters. The assignment step finds each member distribution its nearest mean from K candidates. The mean of each cluster is again a discrete distribution with m support points, denoted by c_i, $i = 1, \ldots, K$. Each mean is iteratively updated to minimize its total within cluster variation. We can write the D2-clustering problem as follows: given sample data $\{d_k\}_{k=1}^N$, support size of means m, and desired number of clusters K, D2-clustering solves

$$\min_{c_1, \ldots, c_K} \sum_{k=1}^{N} \min_{1 \leq i \leq K} W^2(d_k, c_i) \ , \tag{3}$$

where $c_1, \ldots, c_K$ are Wasserstein barycenters. At the core of solving the above formulation is an optimization method that searches the Wasserstein barycenters of varying partitions. Therefore, we concentrate on the following problem. For each cluster, we reorganize the index of member distributions from $1, \ldots, n$. The Wasserstein barycenter (Agueh and Carlier, 2011; Cuturi and Doucet, 2014) is by definition the solution of

$$\min_{c} \sum_{k=1}^{n} W^2(d_k, c) \ , \tag{4}$$

where $c = \sum_{i=1}^{m} w_i \delta_{x_i}$. The above Wasserstein barycenter formulation involves two levels of optimization: the outer level finding the minimizer of total variations, and the inner level solving Wasserstein distances. We remark that in D2-clustering, we need to solve multiple Wasserstein barycenters rather than a single one. This constitutes the third level of optimization.

3.3 Modified Bregman ADMM for Computing Wasserstein Barycenter

The recent modified Bregman alternating direction method of multiplier (B-ADMM) algorithm (Ye et al., 2017), motivated by the work by Wang and Banerjee (2014), is a practical choice for computing Wasserstein barycenters. We briefly sketch their algorithmic procedure of this optimization method here for the sake of completeness. To solve for Wasserstein barycenter defined in Eq. (4), the key procedure of the modified Bregman ADMM involves iterative updates of four block of primal variables: the support points of c — $\{x_i\}_{i=1}^m$ (with transportation costs $\{C_{i,j}\}^{(k)}$ for $k = 1, \ldots, n$), the importance weights of c — $\{w_i\}_{i=1}^m$, and two sets of split matching variables — $\{\pi_{i,j}^{(k,1)}\}$ and $\{\pi_{i,j}^{(k,2)}\}$, for $k = 1, \ldots, n$, as well as Lagrangian variables $\{\lambda_{i,j}^{(k)}\}$ for $k = 1, \ldots, n$.

In the end, both $\{\pi_{i,j}^{(k,1)}\}$ and $\{\pi_{i,j}^{(k,2)}\}$ converge to the matching weight in Eq. (2) with respect to $d(c, d_k)$. The iterative algorithm proceeds as follows until c converges or a maximum number of iterations are reached: given constant $\tau \geq 10$, $\rho \propto \frac{\sum_{i,j,k} C_{i,j}^{(k)}}{\sum_{k=1}^{n} m_k m}$ and round-off tolerance $\epsilon = 10^{-10}$, those variables are updated in the following order.

Update $\{x_i\}_{i=1}^{m}$ **and** $\{C_{i,j}^{(k)}\}$ in every τ iterations:

$$x_i := \frac{1}{n w_i} \sum_{k=1}^{n} \sum_{j=1}^{m_k} \pi_{i,j}^{(k,1)} x_j^{(k)}, \forall i, \quad (5)$$

$$C_{i,j}^{(k)} := \|x_i - x_j^{(k)}\|_2^2, \quad \forall i, j \text{ and } k. \quad (6)$$

Update $\{\pi_{i,j}^{(k,1)}\}$ **and** $\{\pi_{i,j}^{(k,2)}\}$. For each i, j and k,

$$\pi_{i,j}^{(k,2)} := \pi_{i,j}^{(k,2)} \exp\left(\frac{-C_{i,j}^{(k)} - \lambda_{i,j}^{(k)}}{\rho}\right) + \epsilon\,, \quad (7)$$

$$\pi_{i,j}^{(k,1)} := w_j^{(k)} \pi_{i,j}^{(k,2)} \Big/ \left(\sum_{l=1}^{m} \pi_{l,j}^{(k,2)}\right), \quad (8)$$

$$\pi_{i,j}^{(k,1)} := \pi_{i,j}^{(k,1)} \exp\left(\lambda_{i,j}^{(k)}/\rho\right) + \epsilon\,. \quad (9)$$

Update $\{w_i\}_{i=1}^{m}$. For $i = 1, \ldots, m$,

$$w_i := \sum_{k=1}^{n} \frac{\sum_{j=1}^{m_k} \pi_{i,j}^{(k,1)}}{\sum_{i,j} \pi_{i,j}^{(k,1)}}, \quad (10)$$

$$w_i := w_i \Big/ \left(\sum_{i=1}^{m} w_i\right). \quad (11)$$

Update $\{\pi_{i,j}^{(k,2)}\}$ **and** $\{\lambda_{i,j}^{(k)}\}$. For each i, j and k,

$$\pi_{i,j}^{(k,2)} := w_i \pi_{i,j}^{(k,1)} \Big/ \left(\sum_{l=1}^{m_k} \pi_{i,l}^{(k,1)}\right), \quad (12)$$

$$\lambda_{i,j}^{(k)} := \lambda_{i,j}^{(k)} + \rho\left(\pi_{i,l}^{(k,1)} - \pi_{i,l}^{(k,2)}\right). \quad (13)$$

Eq. (5)-(13) can all be vectorized as very efficient numerical routines. In a data parallel implementation, only Eq. (5) and Eq. (10) (involving $\sum_{k=1}^{n}$) needs to be synchronized. The software package detailed in (Ye et al., 2017) was used to generate relevant experiments. We make available our codes and pre-processed datasets for reproducing all experiments of our approach.

4 Experimental Results

4.1 Datasets and Evaluation Metrics

We prepare six datasets to conduct a set of experiments. Two short-text datasets are created as follows. (D1) BBCNews abstract: We concatenate the title and the first sentence of news posts from BBCNews dataset[1] to create an abstract version. (D2) Wiki events: Each cluster/class contains a set of news abstracts on the same story such as "2014 Crimean Crisis" crawled from Wikipedia current events following (Wu et al., 2015); this dataset offers more challenges because it has more fine-grained classes and fewer documents (with shorter length) per class than the others. It also shows more realistic nature of applications such as news event clustering.

We also experiment with two long-text datasets and two domain-specific text datasets. (D3) Reuters-21578: We obtain the original Reuters-21578 text dataset and process as follows: remove documents with multiple categories, remove documents with empty body, remove duplicates, and select documents from the largest ten categories. Reuters dataset is a highly unbalanced dataset (the top category has more than 3,000 documents while the 10-th category has fewer than 100). This imbalance induces some extra randomness in comparing the results. (D4) 20Newsgroups "bydate" version: We obtain the raw "bydate" version and process them as follows: remove headers and footers, remove URLs and Email addresses, delete documents with less than ten words. 20Newsgroups have roughly comparable sizes of categories. (D5) BBCSports. (D6) Ohsumed and Ohsumed-full: Documents are medical abstracts from the MeSH categories of the year 1991. Specifically, there are 23 cardiovascular diseases categories.

Evaluating clustering results is known to be nontrivial. We use the following three sets of quantitative metrics to assess the quality of clusters by knowing the ground truth categorical labels of documents: (i) Homogeneity, Completeness, and V-measure (Rosenberg and Hirschberg, 2007); (ii) Adjusted Mutual Information (AMI) (Vinh et al., 2010); and (iii) Adjusted Rand Index (ARI) (Rand, 1971). For sensitivity analysis, we use the homogeneity score (Rosenberg and Hirschberg, 2007) as a projection dimension of other metrics, creating a 2D plot to visualize the metrics of a method along different homogeneity levels. Generally speaking, more clusters leads to higher homogeneity by chance.

[1] BBCNews and BBCSport are downloaded from http://mlg.ucd.ie/datasets/bbc.html

4.2 Methods in Comparison

We examine four categories of methods that assume a vector-space model for documents, and compare them to our D2-clustering framework. When needed, we use K-means++ to obtain clusters from dimension reduced vectors. To diminish the randomness brought by K-mean initialization, we ensemble the clustering results of 50 repeated runs (Strehl and Ghosh, 2003), and report the metrics for the ensembled one. The largest possible vocabulary used, excluding word embedding based approaches, is composed of words appearing in at least two documents. On each dataset, we select the same set of Ks, the number of clusters, for all methods. Typically, Ks are chosen around the number of ground truth categories in logarithmic scale.

We prepare two versions of the TF-IDF vectors as the unigram model. The ensembled K-means methods are used to obtain clusters. (1) *TF-IDF* vector (Sparck Jones, 1972). (2) *TF-IDF-N* vector is found by choosing the most frequent N words in a corpus, where $N \in \{500, 1000, 1500, 2000\}$. The difference between the two methods highlights the sensitivity issue brought by the size of chosen vocabulary.

We also compare our approach with the following seven additional baselines. They are (3) *Spectral Clustering (Laplacian)*, (4) *Latent Semantic Indexing (LSI)* (Deerwester et al., 1990), (5) *Locality Preserving Projection (LPP)* (He and Niyogi, 2004; Cai et al., 2005), (6) *Non-negative Matrix Factorization (NMF)* (Lee and Seung, 1999; Xu et al., 2003), (7) *Latent Dirichlet Allocation (LDA)* (Blei et al., 2003; Hoffman et al., 2010), (8) *Average of word vectors (AvgDoc)*, and (9) *Paragraph Vectors (PV)* (Le and Mikolov, 2014). Details on their experimental setups and hyper-parameter search strategies can be found in the Appendix.

4.3 Runtime

We report the runtime for our approach on two largest datasets. The experiments regarding other smaller datasets all terminate within minutes in a single machine, which we omit due to space limitation. Like K-means, the runtime by our approach depends on the number of actual iterations before a termination criterion is met. In the Newsgroups dataset, with $m = 100$ and $K = 45$, the time per iteration is 121 seconds on 48 processors. In Reuters dataset, with $m = 100$ and $K = 20$, the time per iteration is 190 seconds on 24 processors. Each run terminates in around tens of iterations typically, upon which the percentage of label changes is less than 0.1%.

Our approach adopts the Elkan's algorithm (2003) pruning unnecessary computations of Wasserstein distance in assignment steps of K-means. For the Newsgroups data (with $m = 100$ and $K = 45$), our approach terminates in 36 iterations, and totally computes $12,162,717$ ($\approx 3.5\% \times 18612^2$) distance pairs in assignment steps, saving 60% ($\approx 1 - \frac{12,162,717}{36 \times 45 \times 18612}$) distance pairs to calculate in the standard D2-clustering. In comparison, the clustering approaches based on K-nearest neighbor (KNN) graph with the prefetch-and-prune method of (Kusner et al., 2015) needs substantially more pairs to compute Wasserstein distance, meanwhile the speed-ups also suffer from the curse of dimensionality. Their detailed statistics are reported in Table 1. Based on the results, our approach is much more practical as a basic document clustering tool.

Method		EMD counts (%)
Our approach	$d = 400, K = 10$	**2.0**
Our approach	$d = 400, K = 40$	**3.5**
KNN	$d = 400, K = 1$	73.9
KNN	$d = 100, K = 1$	53.0
KNN	$d = 50, K = 1$	23.4

Table 1: Percentage of total 18612^2 Wasserstein distance pairs needed to compute on the full Newsgroup dataset. The KNN graph based on 1st order Wasserstein distance is computed from the prefetch-and-prune approach according to (Kusner et al., 2015).

4.4 Results

We now summarize our numerical results.

Regular text datasets. The first four datasets in Table 2 cover quite general and broad topics. We consider them to be regular and representative datasets encountered more frequently in applications. We report the clustering performances of the ten methods in Fig. 1, where three different metrics are plotted against the clustering homogeneity. The higher result at the same level of homogeneity is better, and the ability to achieve higher homogeneity is also welcomed. *Clearly, D2-clustering is the only method that shows ro-*

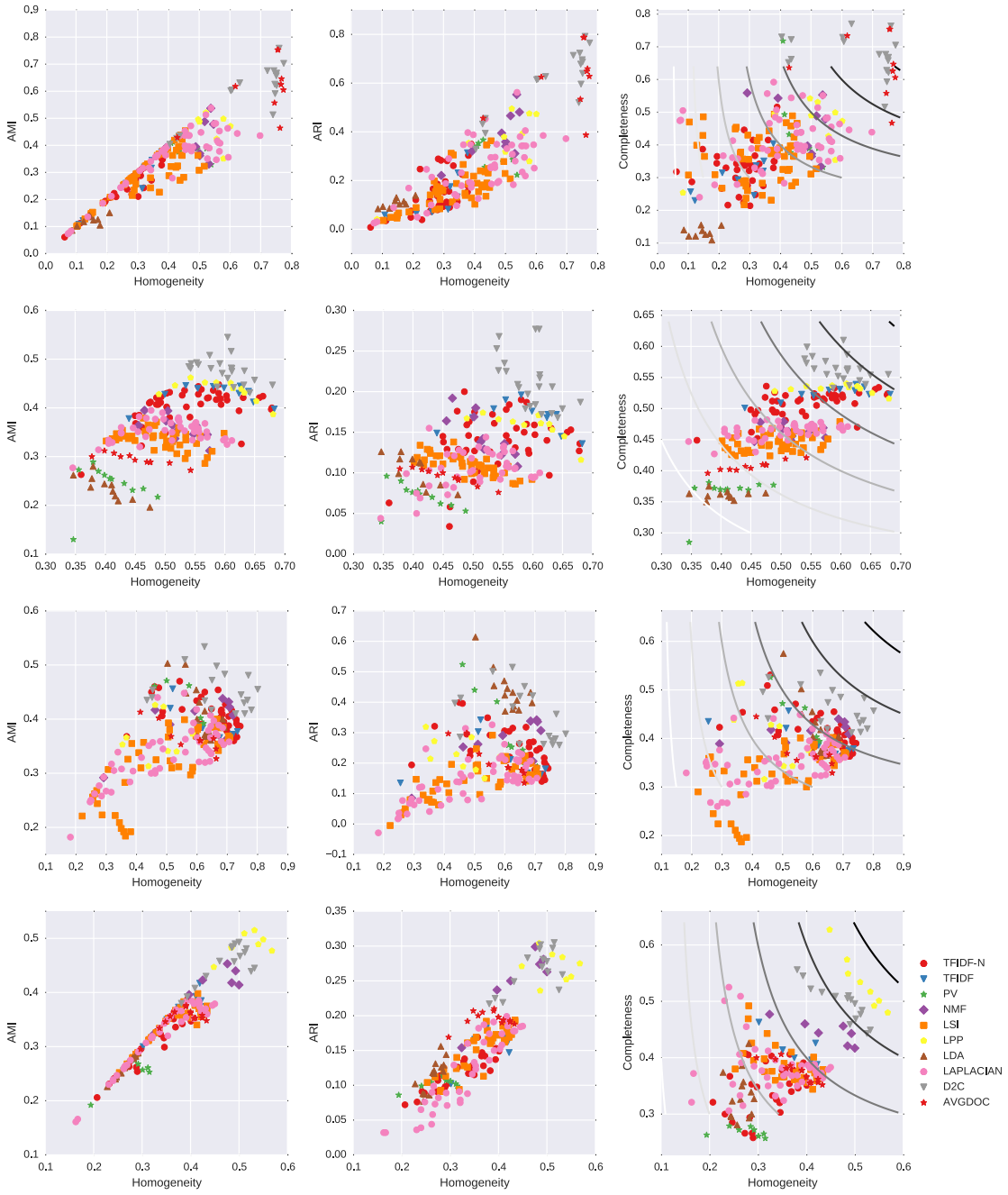


Figure 1: The quantitative cluster metrics used for performance evaluation of "BBC title and abstract", "Wiki events", "Reuters", and "Newsgroups" (row-wise, from top to down). Y-axis corresponds to AMI, ARI, and Completeness, respective (column-wise, from left to right). X-axis corresponds to Homogeneity for sensitivity analysis.

bustly superior performances among all ten methods. Specifically, it ranks first in three datasets, and second in the other one. In comparison, LDA performs competitively on the "Reuters" dataset, but is substantially unsuccessful on others. Meanwhile, LPP performs competitively on the "Wiki events" and "Newsgroups" datasets, but it underperforms on the other two. Laplacian, LSI, and Tfidf-N can achieve comparably performance if their reduced dimensions are fine tuned, which

Dataset	size	class	length	est. #voc.
BBCNews abstr.	2,225	5	26	7,452
Wiki events	1,983	54	22	5,313
Reuters	7,316	10	141	27,792
Newgroups	18,612	20	245	55,970
BBCSports	737	5	345	13,105
Ohsumed	4,340	23	-	-
Ohsumed-full*	34,386	23	184	43,895

Table 2: Description of corpus data that have been used in our experiments. *Ohsumed-full dataset is used for pre-training word embeddings only. Ohsumed is a downsampled evaluation set resulting from removing posts from Ohsumed-full that belong to multiple categories.

unfortunately is unrealistic in practice. NMF is a simple and effective method which always gives stable, though subpar, performance.

Short texts vs. long texts. D2-clustering performs much more impressively on short texts ("BBC abstract" and "Wiki events") than it does on long texts ("Reuters" and "Newsgroups"). This outcome is somewhat expected, because the bag-of-words method suffers from high sparsity for short texts, and word-embedding based methods in theory should have an edge here. As shown in Fig. 1, D2-clustering has indeed outperformed other non-embedding approaches by a large margin on short texts (improved by about 40% and 20% respectively). Nevertheless, we find lifting from word embedding to document clustering is not without a cost. Neither AvgDoc nor PV can perform as competitively as D2-clustering performs on both.

Domain-specific text datasets. We are also interested in how word embedding can help group domain-specific texts into clusters. In particular, does the semantic knowledge "embedded" in words provides enough clues to discriminate fine-grained concepts? We report the best AMI achieved by each method in Table 3. Our preliminary result indicates state-of-the-art word embeddings do not provide enough gain here to exceed the performance of existing methodologies. On the unchallenging one, the "BBCSport" dataset, basic bag-of-words approaches (Tfidf and Tfidf-N) already suffice to discriminate different sport categories; and on the difficult one, the "Ohsumed" dataset, D2-clustering only slightly improves over Tfidf and others, ranking behind LPP. Meanwhile, we feel the overall quality of clustering "Ohsumed" texts is quite far from useful in practice, no matter which method to use. More discussions will be provided next.

4.5 Sensitivity to Word Embeddings.

We validate the robustness of D2-clustering with different word embedding models, and we also show all their results in Fig. 2. As we mentioned, the effectiveness of Wasserstein document clustering depends on how relevant the utilized word embeddings are with the tasks. In those general document clustering tasks, however, word embedding models trained on general corpus perform robustly well with acceptably small variations. This outcome reveals our framework as generally effective and not dependent on a specific word embedding model. In addition, we also conduct experiments with word embeddings with smaller dimensions, at 50 and 100. Their results are not as good as those we have reported (therefore detailed numbers are not included due to space limitation).

Inadequate embeddings may not be disastrous. In addition to our standard running set, we also used D2-clustering with purely random word embeddings, meaning each word vector is independently sampled from spherical Gaussian at 300 dimension, to see how deficient it can be. Experimental results show that random word embeddings degrade the performance of D2-clustering, but it still performs much better than purely random clustering, and is even consistently better than LDA. Its performances across different datasets is highly correlated with the bag-of-words (Tfidf and Tfidf-N). By comparing a pre-trained word embedding model to a randomly generated one, we find that the extra gain is significant ($> 10\%$) in clustering four of the six datasets. Their detailed statistics are in Table 4 and Fig. 3.

5 Discussions

Performance advantage. There has been one immediate observation from these studies, D2-clustering always outperforms two of its degenerated cases, namely Tf-idf and AvgDoc, and three other popular methods: LDA, NMF, and PV, on all tasks. Therefore, for document clustering, users can expect to gain performance improvements by using our approach.

Clustering sensitivity. From the four 2D plots in Fig. 1, we notice that the results of Laplacian,

	regular dataset				domain-specific dataset		
	BBCNews abstract	Wik events	Reuters	Newsgroups	BBCSport	Ohsumed	Avg.
Tfidf-N	0.389	0.448	0.470	0.388	**0.883**	0.210	0.465
Tfidf	0.376	0.446	0.456	0.417	0.799	0.235	0.455
Laplacian	0.538	0.395	0.448	0.385	0.855	0.223	0.474
LSI	0.454	0.379	0.400	0.398	0.840	0.222	0.448
LPP	0.521	0.462	0.426	**0.515**	0.859	**0.284**	0.511
NMF	0.537	0.395	0.438	0.453	0.809	0.226	0.476
LDA	0.151	0.280	0.503	0.288	0.616	0.132	0.328
AvgDoc	**0.753**	0.312	0.413	0.376	0.504	0.172	0.422
PV	0.428	0.289	0.471	0.275	0.553	0.233	0.375
D2C (Our approach)	**0.759**	**0.545**	**0.534**	0.493	0.812	0.260	**0.567**

Table 3: Best AMIs (Vinh et al., 2010) of compared methods on different datasets and their averaging. The best results are marked in bold font for each dataset, the 2nd and 3rd are marked by blue and magenta colors respectively.

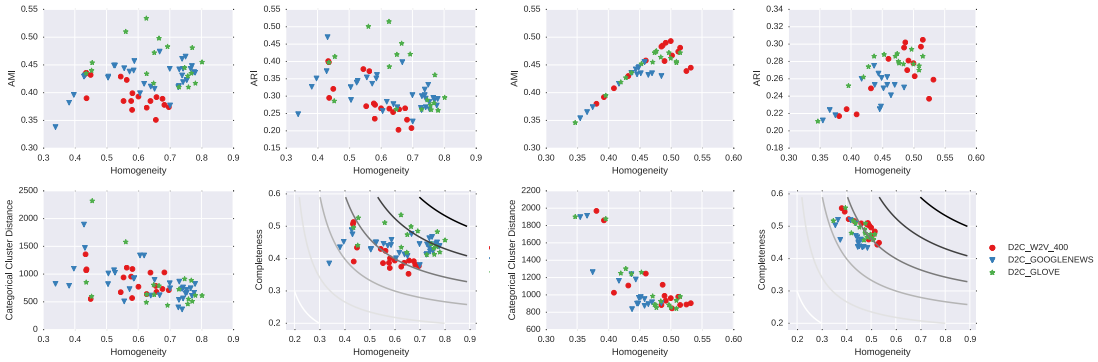


Figure 2: Sensitivity analysis: the clustering performances of D2C under different word embeddings. Left: Reuters, Right: Newsgroups. An extra evaluation index (CCD (Zhou et al., 2005)) is also used.

	ARI	AMI	V-measure
BBCNews abstract	.146	.187	.190
	$.792_{+442\%}$	$.759_{+306\%}$	$.762_{+301\%}$
Wiki events	.194	.369	.463
	$.277_{+43\%}$	$.545_{+48\%}$	$.611_{+32\%}$
Reuters	.498	.524	.588
	$.515_{+3\%}$	$.534_{+2\%}$	$.594_{+1\%}$
Newsgroups	.194	.358	.390
	$.305_{+57\%}$	$.493_{+38\%}$	$.499_{+28\%}$
BBCSport	.755	.740	.760
	$.801_{+6\%}$	$.812_{+10\%}$	$.817_{+8\%}$
Ohsumed	.080	.204	.292
	$.116_{+45\%}$	$.260_{+27\%}$	$.349_{+20\%}$

Table 4: Comparison between *random* word embeddings (upper row) and meaningful *pre-trained* word embeddings (lower row), based on their best ARI, AMI, and V-measures. The improvements by percentiles are also shown in the subscripts.

LSI and Tfidf-N are rather sensitive to their extra hyper-parameters. Once the vocabulary

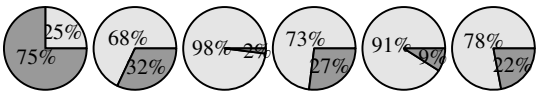


Figure 3: Pie charts of clustering gains in AMI calculated from our framework. Light region is by bag-of-words, and dark region is by pre-trained word embeddings. Six datasets (from left to right): BBCNews abstract, Wiki events, Reuters, Newsgroups, BBCSport, and Ohsumed.

set, weight scheme and embeddings of words are fixed, our framework involves only two additional hyper-parameters: the number of intended clusters, K, and the selected support size of centroid distributions, m. We have chosen more than one m in all related experiments ($m = \{64, 100\}$ for long documents, and $m = \{10, 20\}$ for short documents). Our empirical experiments show that the effect of m on different metrics is less

sensitive than the change of K. Results at different K are plotted for each method (Fig. 1). The gray dots denote results of multiple runs of D2-clustering. They are always contracted around the top-right region of the whole population, revealing the predictive and robustly supreme performance.

When bag-of-words suffices. Among the results of "BBCSport" dataset, Tfidf-N shows that by restricting the vocabulary set into a smaller one (which may be more relevant to the interest of tasks), it already can achieve highest clustering AMI without any other techniques. Other unsupervised regularization over data is likely unnecessary, or even degrades the performance slightly.

Toward better word embeddings. Our experiments on the Ohsumed dataset have been limited. The result shows that it could be highly desirable to incorporate certain domain knowledge to derive more effective vector embeddings of words and phrases to encode their domain-specific knowledge, such as jargons that have knowledge dependencies and hierarchies in educational data mining, and signal words that capture multi-dimensional aspects of emotions in sentiment analysis.

Finally, we report the best AMIs of all methods on all datasets in Table 3. By looking at each method and the average of best AMIs over six datasets, we find our proposed clustering framework often performs competitively and robustly, which is the only method reaching more than 90% of the best AMI on each dataset. Furthermore, this observation holds for varying lengths of documents and varying difficulty levels of clustering tasks.

6 Conclusions and Future Work

This paper introduces a nonparametric clustering framework for document analysis. Its computational tractability, robustness and supreme performance, as a fundamental tool, are empirically validated. Its ease of use enables data scientists to apply it for the pre-screening purpose of examining word embeddings in a specific task. Finally, the gains acquired from word embeddings are quantitatively measured from a nonparametric unsupervised perspective.

It would also be interesting to investigate several possible extensions to the current clustering work. One direction is to learn a proper ground distance for word embeddings such that the final document clustering performance can be improved with labeled data. The work by (Huang et al., 2016; Cuturi and Avis, 2014) have partly touched this goal with an emphasis on document proximities. A more appealing direction is to develop problem-driven methods to represent a document as a distributional entity, taking into consideration of phrases, sentence structures, and syntactical characteristics. We believe the framework of Wasserstein distance and D2-clustering creates room for further investigation on complex structures and knowledge carried by documents.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Nos. ECCS-1462230, DMS-1521092, and Research Grants Council of Hong Kong under Grant No. PolyU 152094/14E. The primary computational infrastructures used were supported by the Foundation under Grant Nos. ACI-0821527 (CyberStar) and ACI-1053575 (XSEDE).

References

Martial Agueh and Guillaume Carlier. 2011. Barycenters in the Wasserstein space. *SIAM J. Math. Analysis* 43(2):904–924.

Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems (NIPS)*. volume 14, pages 585–591.

Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. 2015. Iterative Bregman projections for regularized transportation problems. *SIAM J. Sci. Computing (SJSC)* 37(2):A1111–A1138.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *J. Machine Learning Research (JMLR)* 3:993–1022.

Deng Cai, Xiaofei He, and Jiawei Han. 2005. Document clustering using locality preserving indexing. *Trans. Knowledge and Data Engineering (TKDE)* 17(12):1624–1637.

Marco Cuturi and David Avis. 2014. Ground metric learning. *Journal of Machine Learning Research* 15(1):533–564.

Marco Cuturi and Arnaud Doucet. 2014. Fast computation of Wasserstein barycenters. In *Int. Conf. Machine Learning (ICML)*. pages 685–693.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *J. American Soc. Information Science* 41(6):391–407.

Charles Elkan. 2003. Using the triangle inequality to accelerate k-means. In *Int. Conf. Machine Learning (ICML)*. volume 3, pages 147–153.

Xiaofei He and Partha Niyogi. 2004. Locality preserving projections. In *Advances in Neural Information Processing Systems (NIPS)*. MIT, volume 16, page 153.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*. pages 856–864.

Gao Huang, Chuan Guo, Matt J Kusner, Yu Sun, Fei Sha, and Kilian Q Weinberger. 2016. Supervised word mover's distance. In *Advances in Neural Information Processing Systems (NIPS)*. pages 4862–4870.

Matt J Kusner, Yu Sun, Nicholas N I. Kolkin, and K Q. Weinberger. 2015. From word embeddings to document distances. In *Int. Conf. Machine Learning (ICML)*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Int. Conf. Machine Learning*. pages 1188–1196.

Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791.

Jia Li and James Z Wang. 2008. Real-time computerized annotation of pictures. *Trans. Pattern Analysis and Machine Intelligence (PAMI)* 30(6):985–1002.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*. pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *HLT-NAACL*. pages 746–751.

James B Orlin. 1993. A faster strongly polynomial minimum cost flow algorithm. *Operations research* 41(2):338–350.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. volume 14, pages 1532–1543.

William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *J. American Statistical Association* 66(336):846–850.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*. volume 7, pages 410–420.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2011. Large-scale cross-document coreference using distributed inference and hierarchical models. In *ACL-HLT*. Association for Computational Linguistics, pages 793–803.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation* 28(1):11–21.

Alexander Strehl and Joydeep Ghosh. 2003. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J. Machine Learning Research (JMLR)* 3:583–617.

Cédric Villani. 2003. *Topics in optimal transportation*. 58. American Mathematical Soc.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Machine Learning Research (JMLR)* 11:2837–2854.

Xiaojun Wan. 2007. A novel document similarity measure based on earth movers distance. *Information Sciences* 177(18):3718–3730.

Huahua Wang and Arindam Banerjee. 2014. Bregman alternating direction method of multipliers. In *Advances in Neural Information Processing Systems (NIPS)*. pages 2816–2824.

Zhaohui Wu, Chen Liang, and C Lee Giles. 2015. Storybase: Towards building a knowledge base for news events. In *ACL-IJCNLP 2015*. pages 133–138.

Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *ACM SIGIR Conf. on Research and Development in Informaion Retrieval*. ACM, pages 267–273.

Jianbo Ye and Jia Li. 2014. Scaling up discrete distribution clustering using admm. In *Int. Conf. Image Processing (ICIP)*. IEEE, pages 5267–5271.

Jianbo Ye, Panruo Wu, James Z. Wang, and Jia Li. 2017. Fast discrete distribution clustering using Wasserstein barycenter with sparse support. *IEEE Trans. on Signal Processing (TSP)* 65(9):2317–2332.

Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2011. Clustering product features for opinion mining. In *Int. Conf. on Web Search and Data Mining (WSDM)*. ACM, pages 347–354.

Ding Zhou, Jia Li, and Hongyuan Zha. 2005. A new mallows distance based metric for comparing clusterings. In *Int. Conf. Machine Learning (ICML)*. ACM, pages 1028–1035.

Towards a Seamless Integration of Word Senses into Downstream NLP Applications

Mohammad Taher Pilehvar[2], Jose Camacho-Collados[1], Roberto Navigli[1] and Nigel Collier[2]

[1]Department of Computer Science, Sapienza University of Rome
[2]Department of Theoretical and Applied Linguistics, University of Cambridge
[1]{collados,navigli}@di.uniroma1.it
[2]{mp792,nhc30}@cam.ac.uk

Abstract

Lexical ambiguity can impede NLP systems from accurate understanding of semantics. Despite its potential benefits, the integration of sense-level information into NLP systems has remained understudied. By incorporating a novel disambiguation algorithm into a state-of-the-art classification model, we create a pipeline to integrate sense-level information into downstream NLP applications. We show that a simple disambiguation of the input text can lead to consistent performance improvement on multiple topic categorization and polarity detection datasets, particularly when the fine granularity of the underlying sense inventory is reduced and the document is sufficiently large. Our results also point to the need for sense representation research to focus more on *in vivo* evaluations which target the performance in downstream NLP applications rather than artificial benchmarks.

1 Introduction

As a general trend, most current Natural Language Processing (NLP) systems function at the word level, i.e. individual words constitute the most fine-grained meaning-bearing elements of their input. The word level functionality can affect the performance of these systems in two ways: (1) it can hamper their efficiency in handling words that are not encountered frequently during training, such as multiwords, inflections and derivations, and (2) it can restrict their semantic understanding to the level of words, with all their ambiguities, and thereby prevent accurate capture of the intended meanings.

The first issue has recently been alleviated by techniques that aim to boost the generalisation power of NLP systems by resorting to sub-word or character-level information (Ballesteros et al., 2015; Kim et al., 2016). The second limitation, however, has not yet been studied sufficiently. A reasonable way to handle word ambiguity, and hence to tackle the second issue, is to *semantify* the input text: transform it from its surface-level semantics to the deeper level of word senses, i.e. their intended meanings. We take a step in this direction by designing a pipeline that enables seamless integration of word senses into downstream NLP applications, while benefiting from knowledge extracted from semantic networks. To this end, we propose a quick graph-based Word Sense Disambiguation (WSD) algorithm which allows high confidence disambiguation of words without much computation overload on the system. We evaluate the pipeline in two downstream NLP applications: polarity detection and topic categorization. Specifically, we use a classification model based on Convolutional Neural Networks which has been shown to be very effective in various text classification tasks (Kalchbrenner et al., 2014; Kim, 2014; Johnson and Zhang, 2015; Tang et al., 2015; Xiao and Cho, 2016). We show that a simple disambiguation of input can lead to performance improvement of a state-of-the-art text classification system on multiple datasets, particularly for long inputs and when the granularity of the sense inventory is reduced. Our pipeline is quite flexible and modular, as it permits the integration of different WSD and sense representation techniques.

2 Motivation

With the help of an example news article from the BBC, shown in Figure 1, we highlight some of the potential deficiencies of word-based models.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1857–1869
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1170

Lewis Hamilton is heading to his fourth **F1** drivers' title after **German GP** win

The **German Grand Prix** was the last **race** before **Formula 1** heads off for its four-week summer break, so it was fitting that it consolidated the two overriding trends that have emerged so far this year.

[...]

Hockenheim was **Hamilton**'s sixth win in seven races, a remarkable run that has seen a 62-point swing between himself and **Mercedes** team-mate **Nico Rosberg**, turning a 43-point deficit into a 19-point lead.

Figure 1: Excerpt of a news article from the BBC.

Algorithm 1 Disambiguation algorithm

Input: Input text T and semantic network N
Output: Set of disambiguated senses $\hat{S}$

Graph representation of T: $(S, E) \leftarrow \text{getGraph}(T, N)$
$\hat{S} \leftarrow \emptyset$
for each iteration $i \in \{1, ..., len(T)\}$
  $\hat{s} = \text{argmax}_{s \in S} |\{(s, s') \in E : s' \in S\}|$
  $\text{maxDeg} = |\{(\hat{s}, s') \in E : s' \in S\}|$
  if $\text{maxDeg} < \theta |S| \ / \ 100$ **then**
    break
  else
    $\hat{S} \leftarrow \hat{S} \cup \{\hat{s}\}$
    $E \leftarrow E \setminus \{(s, s') : s \vee s' \in \text{getLex}(\hat{s})\}$
return Disambiguation output $\hat{S}$

Ambiguity. Language is inherently ambiguous. For instance, *Mercedes*, *race*, *Hamilton* and *Formula* can refer to several different entities or meanings. Current neural models have managed to successfully represent complex semantic associations by effectively analyzing large amounts of data. However, the word-level functionality of these systems is still a barrier to the depth of their natural language understanding. Our proposal is particularly tailored towards addressing this issue.

Multiword expressions (MWE). MWE are lexical units made up of two or more words which are idiosyncratic in nature (Sag et al., 2002), e.g, *Lewis Hamilton*, *Nico Rosberg* and *Formula 1*. Most existing word-based models ignore the interdependency between MWE's subunits and treat them as individual units. Handling MWE has been a long-standing problem in NLP and has recently received a considerable amount of interest (Tsvetkov and Wintner, 2014; Salehi et al., 2015). Our pipeline facilitates this goal.

Co-reference. Co-reference resolution of concepts and entities is not explicitly tackled by our approach. However, thanks to the fact that words that refer to the same meaning in context, e.g., *Formula 1-F1* or *German Grand Prix-German GP-Hockenheim*, are all disambiguated to the same concept, the co-reference issue is also partly addressed by our pipeline.

3 Disambiguation Algorithm

Our proposal relies on a seamless integration of word senses in word-based systems. The goal is to semantify the text prior to its being fed into the system by transforming its individual units from word surface form to the deeper level of word senses. The semantification step is mainly tailored towards resolving ambiguities, but it brings about other advantages mentioned in the previous section. The aim is to provide the system with an input of reduced ambiguity which can facilitate its decision making.

To this end, we developed a simple graph-based joint disambiguation and entity linking algorithm which can take any arbitrary semantic network as input. The gist of our disambiguation technique lies in its speed and scalability. Conventional knowledge-based disambiguation systems (Hoffart et al., 2012; Agirre et al., 2014; Moro et al., 2014; Ling et al., 2015; Pilehvar and Navigli, 2014) often rely on computationally expensive graph algorithms, which limits their application to on-the-fly processing of large number of text documents, as is the case in our experiments. Moreover, unlike supervised WSD and entity linking techniques (Zhong and Ng, 2010; Cheng and Roth, 2013; Melamud et al., 2016; Limsopatham and Collier, 2016), our algorithm relies only on semantic networks and does not require any sense-annotated data, which is limited to English and almost non-existent for other languages.

Algorithm 1 shows our procedure for disambiguating an input document T. First, we retrieve from our semantic network the list of candidate senses[1] for each content word, as well as semantic relationships among them. As a result, we obtain a graph representation (S, E) of the input text, where S is the set of candidate senses and E is the set of edges among different senses in S. The graph is, in fact, a small sub-graph of the input semantic network, N. Our algorithm then selects the best candidates iteratively. In each iteration, the

[1] As defined in the underlying sense inventory, up to trigrams. We used Stanford CoreNLP (Manning et al., 2014) for tokenization, Part-of-Speech (PoS) tagging and lemmatization.

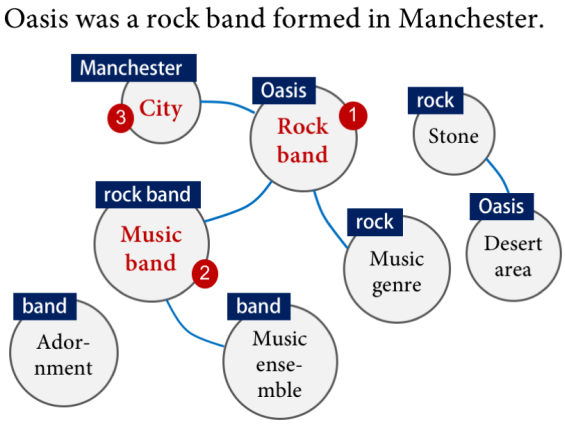


Figure 2: Simplified graph-based representation of a sample sentence.

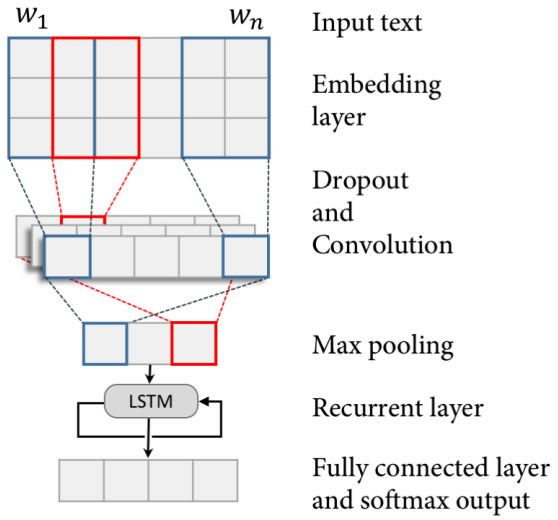


Figure 3: Text classification model architecture.

candidate sense that has the highest graph degree maxDeg is chosen as the winning sense:

$$\text{maxDeg} = \max_{s \in S} |\{(s, s') \in E : s' \in S\}| \quad (1)$$

After each iteration, when a candidate sense $\hat{s}$ is selected, all the possible candidate senses of the corresponding word (i.e. $\text{getLex}(\hat{s})$) are removed from E (line 10 in the algorithm).

Figure 2 shows a simplified version of the graph for a sample sentence. The algorithm would disambiguate the content words in this sentence as follows. It first associates *Oasis* with its *rock band* sense, since its corresponding node has the highest degree, i.e. 3. On the basis of this, the *desert* sense of *Oasis* and its link to the *stone* sense of *rock* are removed from the graph. In the second iteration, *rock band* is disambiguated as *music band* given that its degree is 2.[2] Finally, *Manchester* is associated with its *city* sense (with a degree of 1).

In order to enable disambiguating at different confidence levels, we introduce a threshold θ which determines the stopping criterion of the algorithm. Iteration continues until the following condition is fulfilled: $\text{maxDeg} < \theta|S| \,/\, 100$. This ensures that the system will only disambiguate those words for which it has a high confidence and backs off to the word form otherwise, avoiding the introduction of unwanted noise in the data for uncertain cases or for word senses that are not defined in the inventory.

[2]For bigrams and trigrams whose individual words might also be disambiguated (such as *rock* and *band* in *rock band*), the longest unit has the highest priority (i.e. *rock band*).

4 Classification Model

In our experiments, we use a standard neural network based classification approach which is similar to the Convolution Neural Network classifier of Kim (2014) and the pioneering model of Collobert et al. (2011). Figure 3 depicts the architecture of the model. The network receives the concatenated vector representations of the input words, $\mathbf{v}_{1:n} = \mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \cdots \oplus \mathbf{v}_n$, and applies (convolves) filters F on windows of h words, $m_i = f(\text{F}.\mathbf{v}_{i:i+h-1} + b)$, where b is a bias term and $f()$ is a non-linear function, for which we use ReLU (Nair and Hinton, 2010). The convolution transforms the input text to a feature map $m = [m_1, m_2, \ldots, m_{n-h+1}]$. A max pooling operation then selects the most salient feature $\hat{m} = \max\{m\}$ for each filter.

In the network of Kim (2014), the pooled features are directly passed to a fully connected softmax layer whose outputs are class probabilities. However, we add a recurrent layer before softmax in order to enable better capturing of long-distance dependencies. It has been shown by Xiao and Cho (2016) that a recurrent layer can replace multiple layers of convolution and be beneficial, particularly when the length of input text grows. Specifically, we use a Long Short-Term Memory (Hochreiter and Schmidhuber, 1997, LSTM) as our recurrent layer which was originally proposed to avoid the vanishing gradient problem and has proven its abilities in capturing distant dependencies. The LSTM unit computes three gate vectors

(forget, input, and output) as follows:

$$\begin{aligned}\mathbf{f}_t &= \sigma(\mathbf{W}_f\ g_t + \mathbf{U}_f\ h_{t-1} + \mathbf{b}_f),\\ \mathbf{i}_t &= \sigma(\mathbf{W}_i\ g_t + \mathbf{U}_i\ h_{t-1} + \mathbf{b}_i),\\ \mathbf{o}_t &= \sigma(\mathbf{W}_o\ g_t + \mathbf{U}_o\ h_{t-1} + \mathbf{b}_o),\end{aligned} \quad (2)$$

where $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are model parameters and g and h are input and output sequences, respectively. The cell state vector $\mathbf{c}_t$ is then computed as $\mathbf{c}_t = \mathbf{f}_t\ \mathbf{c}_{t-1} + \mathbf{i}_t\ \tanh(\tilde{\mathbf{c}}_t)$ where $\tilde{c}_t = \mathbf{W}_c\ g_t + \mathbf{U}_c\ h_{t-1}$. Finally, the output sequence is computed as $h_t = \mathbf{o}_t\ \tanh(\mathbf{c}_t)$. As for regularization, we used dropout (Hinton et al., 2012) after the embedding layer.

We perform experiments with two configurations of the embedding layer: (1) *Random*, initialized randomly and updated during training, and (2) *Pre-trained*, initialized by pre-trained representations and updated during training. In the following section we describe the pre-trained word and sense representation used for the initialization of the second configuration.

4.1 Pre-trained Word and Sense Embeddings

One of the main advantages of neural models is that they usually represent the input words as dense vectors. This can significantly boost a system's generalisation power and results in improved performance (Zou et al., 2013; Bordes et al., 2014; Kim, 2014; Weiss et al., 2015, *inter-alia*). This feature also enables us to directly plug in pre-trained sense representations and check them in a downstream application.

In our experiments we generate a set of sense embeddings by extending DeConf, a recent technique with state-of-the-art performance on multiple semantic similarity benchmarks (Pilehvar and Collier, 2016). We leave the evaluation of other representations to future work. DeConf gets a pre-trained set of word embeddings and computes sense embeddings in the same semantic space. To this end, the approach exploits the semantic network of WordNet (Miller, 1995), using the Personalized PageRank (Haveliwala, 2002) algorithm, and obtains a set of *sense biasing words* $\mathcal{B}_s$ for a word sense s. The sense representation of s is then obtained using the following formula:

$$\hat{\mathbf{v}}(s) = \frac{1}{|\mathcal{B}_s|}\sum_{i=1}^{|\mathcal{B}_s|} \mathrm{e}^{\frac{-i}{\delta}}\mathbf{v}(w_i), \quad (3)$$

where δ is a decay parameter and $\mathbf{v}(w_i)$ is the embedding of w_i, i.e. the i^{th} word in the sense biasing list of s, i.e. $\mathcal{B}_s$. We follow Pilehvar and Collier (2016) and set $\delta = 5$. Finally, the vector for sense s is calculated as the average of $\hat{\mathbf{v}}(s)$ and the embedding of its corresponding word.

Owing to its reliance on WordNet's semantic network, DeConf is limited to generating only those word senses that are covered by this lexical resource. We propose to use Wikipedia in order to expand the vocabulary of the computed word senses. Wikipedia provides a high coverage of named entities and domain-specific terms in many languages, while at the same time also benefiting from a continuous update by collaborators. Moreover, it can easily be viewed as a sense inventory where individual articles are word senses arranged through hyperlinks and redirections.

Camacho-Collados et al. (2016b) proposed NASARI[3], a technique to compute the most salient words for each Wikipedia page. These salient words were computed by exploiting the structure and content of Wikipedia and proved effective in tasks such as Word Sense Disambiguation (Tripodi and Pelillo, 2017; Camacho-Collados et al., 2016a), knowledge-base construction (Lieto et al., 2016), domain-adapted hypernym discovery (Espinosa-Anke et al., 2016; Camacho-Collados and Navigli, 2017) or object recognition (Young et al., 2016). We view these lists as *biasing words* for individual Wikipedia pages, and then leverage the exponential decay function (Equation 3) to compute new sense embeddings in the same semantic space. In order to represent both WordNet and Wikipedia sense representations in the same space, we rely on the WordNet-Wikipedia mapping provided by BabelNet[4] (Navigli and Ponzetto, 2012). For the WordNet synsets which are mapped to Wikipedia pages in BabelNet, we average the corresponding Wikipedia-based and WordNet-based sense embeddings.

4.2 Pre-trained Supersense Embeddings

It has been argued that WordNet sense distinctions are too fine-grained for many NLP applications (Hovy et al., 2013). The issue can be tackled by grouping together similar senses of the same word, either using automatic clustering techniques (Navigli, 2006; Agirre and Lopez, 2003; Snow et al., 2007) or with the help of WordNet's lexicographer

[3]We downloaded the salient words for Wikipedia pages (NASARI English lexical vectors, version 3.0) from http://lcl.uniroma1.it/nasari/

[4]We used the Java API from http://babelnet.org

files[5]. Various applications have been shown to improve upon moving from senses to supersenses (Rüd et al., 2011; Severyn et al., 2013; Flekova and Gurevych, 2016). In WordNet's lexicographer files there are a total of 44 sense clusters, referred to as supersenses, for categories such as *event*, *animal*, and *quantity*. In our experiments we use these supersenses in order to reduce granularity of our WordNet and Wikipedia senses. To generate supersense embeddings, we simply average the embeddings of senses in the corresponding cluster.

5 Evaluation

We evaluated our model on two classification tasks: topic categorization (Section 5.2) and polarity detection (Section 5.3). In the following section we present the common experimental setup.

5.1 Experimental setup

Classification model. Throughout all the experiments we used the classification model described in Section 4. The general architecture of the model was the same for both tasks, with slight variations in hyperparameters given the different natures of the tasks, following the values suggested by Kim (2014) and Xiao and Cho (2016) for the two tasks. Hyperparameters were fixed across all configurations in the corresponding tasks. The embedding layer was fixed to 300 dimensions, irrespective of the configuration, i.e. Random and Pre-trained. For both tasks the evaluation was carried out by 10-fold cross-validation unless standard training-testing splits were available. The disambiguation threshold θ (cf. Section 3) was tuned on the training portion of the corresponding data, over seven values in [0,3] in steps of 0.5.[6] We used Keras (Chollet, 2015) and Theano (Team, 2016) for our model implementations.

Semantic network. The integration of senses was carried out as described in Section 3. For disambiguating with both WordNet and Wikipedia senses we relied on the joint semantic network of Wikipedia hyperlinks and WordNet via the mapping provided by BabelNet.[7]

Pre-trained word and sense embeddings. Throughout all the experiments we used Word2vec (Mikolov et al., 2013) embeddings, trained on the Google News corpus.[8] We truncated this set to its 250K most frequent words. We also used WordNet 3.0 (Fellbaum, 1998) and the Wikipedia dump of November 2014 to compute the sense embeddings (see Section 4.1). As a result, we obtained a set of 757,262 sense embeddings in the same space as the pre-trained Word2vec word embeddings. We used DeConf (Pilehvar and Collier, 2016) as our pre-trained WordNet sense embeddings. All vectors had a fixed dimensionality of 300.

Supersenses. In addition to WordNet senses, we experimented with supersenses (see Section 4.2) to check how reducing granularity would affect system performance. For obtaining supersenses in a given text we relied on our disambiguation pipeline and simply clustered together senses belonging to the same WordNet supersense.

Evaluation measures. We report the results in terms of standard accuracy and F1 measures.[9]

5.2 Topic Categorization

The task of topic categorization consists of assigning a label (i.e. topic) to a given document from a pre-defined set of labels.

5.2.1 Datasets

For this task we used two newswire and one medical topic categorization datasets. Table 1 summarizes the statistics of each dataset.[10] The **BBC news** dataset[11] (Greene and Cunningham, 2006) comprises news articles taken from BBC, divided into five topics: business, entertainment, politics, sport and tech. **Newsgroups** (Lang, 1995) is a collection of 11,314 documents for training and 7532 for testing[12] divided into six topics: computing, sport and motor vehicles, science, politics, reli-

[5] https://wordnet.princeton.edu/man/lexnames.5WN.html

[6] We observed that values higher than 3 led to very few disambiguations. While the best results were generally achieved in the [1.5,2.5] range, performance differences across threshold values were not statistically significant in most cases.

[7] For simplicity we refer to this joint sense inventory as Wikipedia, but note that WordNet senses are also covered.

[8] https://code.google.com/archive/p/word2vec/

[9] Since all models in our experiments provide full coverage, accuracy and F1 denote micro- and macro-averaged F1, respectively (Yang, 1999).

[10] The coverage of the datasets was computed using the 250K top words in the Google News Word2vec embeddings.

[11] http://mlg.ucd.ie/datasets/bbc.html

[12] We used the train-test partition available at http://qwone.com/~jason/20Newsgroups/

Dataset	Domain	No. of classes	No. of docs	Avg. doc. size	Size of vocab.	Coverage	Evaluation
BBC	News	5	2,225	439.5	35,628	87.4%	10 cross valid.
Newsgroups	News	6	18,846	394.0	225,046	83.4%	Train-Test
Ohsumed	Medical	23	23,166	201.2	65,323	79.3%	Train-Test

Table 1: Statistics of the topic categorization datasets.

Initialization	Input type		BBC News		Newsgroups		Ohsumed	
			Acc	F1	Acc	F1	Acc	F1
Random	**Word**		93.0	92.8	87.7	85.6	30.1	20.7
	Sense	**WordNet**	**93.5**	**93.3**	**88.1**	**86.9**	27.2†	18.3
		Wikipedia	92.7	92.5	86.7	84.9	29.7	**20.9**
	Supersense	**WordNet**	**93.6**	**93.4**	**90.1***	**89.0**	**31.8***	**22.0**
		Wikipedia	**94.6***	**94.4**	**88.5**	**85.8**	**31.1**	**21.3**
Pre-trained	**Word**		97.6	97.5	91.1	90.6	29.4	20.1
	Sense	**WordNet**	97.3	97.1	90.2	88.6	**30.2**	**20.4**
		Wikipedia	96.3	96.2	89.6†	88.9	**32.4**	**22.3**
	Supersense	**WordNet**	96.8	96.7	89.6	88.9	**29.5**	19.9
		Wikipedia	96.9	96.9	88.6	87.4	**30.6***	**20.3**

Table 2: Classification performance at the word, sense, and supersense levels with random and pre-trained embedding initialization. We show in bold those settings that improve the word-based model.

gion and sales.[13] Finally, **Ohsumed**[14] is a collection of medical abstracts from MEDLINE, an online medical information database, categorized according to 23 cardiovascular diseases. For our experiments we used the partition split of 10,433 documents for training and 12,733 for testing.[15]

5.2.2 Results

Table 2 shows the results of our classification model and its variants on the three datasets.[16] When the embedding layer is initialized randomly, the model integrated with word senses consistently improves over the word-based model, particularly when the fine-granularity of the underlying sense inventory is reduced using supersenses (with statistically significant gains on the three datasets). This highlights the fact that a simple disambiguation of the input can bring about performance gain for a state-of-the-art classification system. Also, the better performance of supersenses suggests that the sense distinctions of WordNet are too fine-grained for the topic categorization task. However, when pre-trained representations are used to initialize the embedding layer, no improvement is observed over the word-based model. This can be attributed to the quality of the representations, as the model utilizing them was unable to benefit from the advantage offered by sense distinctions. Our results suggest that research in sense representation should put special emphasis on real-world evaluations on benchmarks for downstream applications, rather than on artificial tasks such as word similarity. In fact, research has previously shown that word similarity might not constitute a reliable proxy to measure the performance of word embeddings in downstream applications (Tsvetkov et al., 2015; Chiu et al., 2016).

Among the three datasets, Ohsumed proves to be the most challenging one, mainly for its larger number of classes (i.e. 23) and its domain-specific nature (i.e. medicine). Interestingly, unlike for the other two datasets, the introduction of pre-trained word embeddings to the system results in reduced performance on Ohsumed. This suggests that general domain embeddings might not be beneficial

[13] The dataset has 20 fine-grained categories clustered into six general topics. We used the coarse-grained labels for their clearer distinction and consistency with BBC topics.

[14] ftp://medir.ohsu.edu/pub/ohsumed

[15] http://disi.unitn.it/moschitti/corpora.htm

[16] Symbols * and † indicate the sense-based model with the smallest margin to the word-based model whose accuracy is statistically significant at 0.95 confidence level according to unpaired t-test (* for positive and † for negative change).

in specialized domains, which corroborates previous findings by Yadav et al. (2017) on a different task, i.e. entity extraction. This performance drop may also be due to diachronic issues (Ohsumed dates back to the 1980s) and low coverage: the pre-trained Word2vec embeddings cover 79.3% of the words in Ohsumed (see Table 1), in contrast to the higher coverage on the newswire datasets, i.e. Newsgroups (83.4%) and BBC (87.4%). However, also note that the best overall performance is attained when our pre-trained Wikipedia sense embeddings are used. This highlights the effectiveness of Wikipedia in handling domain-specific entities, thanks to its broad sense inventory.

5.3 Polarity Detection

Polarity detection is the most popular evaluation framework for sentiment analysis (Dong et al., 2015). The task is essentially a binary classification which determines if the sentiment of a given sentence or document is negative or positive.

5.3.1 Datasets

For the polarity detection task we used five standard evaluation datasets. Table 1 summarizes statistics. **PL04** (Pang and Lee, 2004) is a polarity detection dataset composed of full movie reviews. **PL05**[18] (Pang and Lee, 2005), instead, is composed of short snippets from movie reviews. **RTC** contains critic reviews from Rotten Tomatoes[19], divided into 436,000 training and 2,000 test instances. **IMDB** (Maas et al., 2011) includes 50,000 movie reviews, split evenly between training and test. Finally, we used the **Stanford** Sentiment dataset (Socher et al., 2013), which associates each review with a value that denotes its sentiment. To be consistent with the binary classification of the other datasets, we removed the neutral phrases according to the dataset's scale (between 0.4 and 0.6) and considered the reviews whose values were below 0.4 as negative and above 0.6 as positive. This resulted in a binary polarity dataset of 119,783 phrases. Unlike the previous four datasets, this dataset does not contain an even distribution of positive and negative labels.

5.3.2 Results

Table 4 lists accuracy performance of our classification model and all its variants on five polarity detection datasets. Results are generally better than those of Kim (2014), showing that the addition of the recurrent layer to the model (cf. Section 4) was beneficial. However, interestingly, no consistent performance gain is observed in the polarity detection task, when the model is provided with disambiguated input, particularly for datasets with relatively short reviews. We attribute this to the nature of the task. Firstly, given that words rarely happen to be ambiguous with respect to their sentiment, the semantic sense distinctions provided by the disambiguation stage do not assist the classifier in better decision making, and instead introduce data sparsity. Secondly, since the datasets mostly contain short texts, e.g., sentences or snippets, the disambiguation algorithm does not have sufficient context to make high-confidence judgements, resulting in fewer disambiguations or less reliable ones. In the following section we perform a more in-depth analysis of the impact of document size on the performance of our sense-based models.

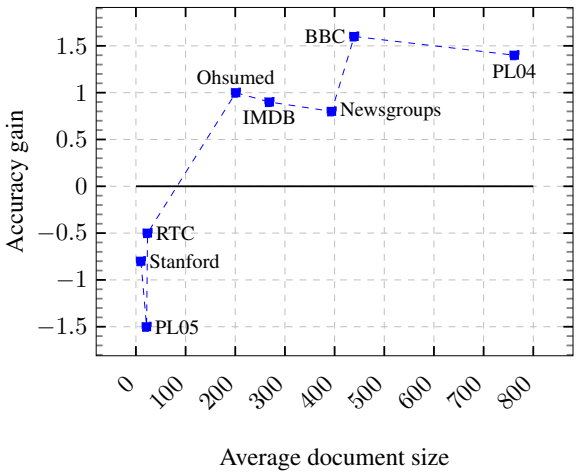


Figure 4: Relation between average document size and performance improvement using Wikipedia supersenses with random initialization.

5.4 Analysis

Document size. A detailed analysis revealed a relation between document size (the number of tokens) and performance gain of our sense-level model. We show in Figure 4 how these two vary for our most consistent configuration, i.e. Wikipedia supersenses, with random initialization. Interestingly, as a general trend, the performance gain increases with average document size, irre-

[18] Both PL04 and PL05 were downloaded from http://www.cs.cornell.edu/people/pabo/movie-review-data/

[19] http://www.rottentomatoes.com

[19] Stanford is the only unbalanced dataset, but F1 results were almost identical to accuracy.

Dataset	Type	No. of docs	Avg. doc. size	Vocabulary size	Coverage	Evaluation
RTC	Snippets	438,000	23.4	128,056	81.3%	Train-Test
IMDB	Reviews	50,000	268.8	140,172	82.5%	Train-Test
PL05	Snippets	10,662	21.5	19,825	81.3%	10 cross valid.
PL04	Reviews	2,000	762.1	45,077	82.4%	10 cross valid.
Stanford	Phrases	119,783	10.0	19,400	81.6%	10 cross valid.

Table 3: Statistics of the polarity detection datasets.

Initialization	Input type		RTC	IMDB	PL05	PL04	Stanford
Random	**Word**		83.6	87.7	77.3	67.9	91.8
	Sense	**WordNet**	83.2	87.4	76.6	67.4	91.3
		Wikipedia	83.1	**88.0**	75.9†	67.1	91.0
	Supersense	**WordNet**	**84.4**	**88.0**	75.9	66.2	91.4†
		Wikipedia	83.1	**88.4***	75.8	**69.3***	91.0
Pre-trained	**Word**		85.5	88.3	80.2	72.5	93.1
	Sense	**WordNet**	83.4	**88.3**	79.2	69.7†	92.6
		Wikipedia	83.8	87.0†	79.2	**73.1**	92.3
	Supersense	**WordNet**	85.2	**88.8**	79.5	**73.8**	92.7†
		Wikipedia	84.2	87.9	78.3†	**72.6**	92.2

Table 4: Accuracy performance on five polarity detection datasets. Given that polarity datasets are balanced[17], we do not report F1 which would have been identical to accuracy.

spective of the classification task. We attribute this to two main factors:

1. **Sparsity**: Splitting a word into multiple word senses can have the negative side effect that the corresponding training data for that word is distributed among multiple independent senses. This reduces the training instances per word sense, which might affect the classifier's performance, particularly when senses are semantically related (in comparison to fine-grained senses, supersenses address this issue to some extent).

2. **Disambiguation quality**: As also mentioned previously, our disambiguation algorithm requires the input text to be sufficiently large so as to create a graph with an adequate number of coherent connections to function effectively. In fact, for topic categorization, in which the documents are relatively long, our algorithm manages to disambiguate a larger proportion of words in documents with high confidence. The lower performance of graph-based disambiguation algorithms on short texts is a known issue (Moro et al., 2014; Raganato et al., 2017), the tackling of which remains an area of exploration.

Senses granularity. Our results showed that reducing fine-granularity of sense distinctions can be beneficial to both tasks, irrespective of the underlying sense inventory, i.e. WordNet or Wikipedia, which corroborates previous findings (Hovy et al., 2013; Flekova and Gurevych, 2016). This suggests that text classification does not require fine-grained semantic distinctions. In this work we used a simple technique based on WordNet's lexicographer files for coarsening senses in this sense inventory as well as in Wikipedia. We leave the exploration of this promising area as well as the evaluation of other granularity reduction techniques for WordNet (Snow et al., 2007; Bhagwani et al., 2013) and Wikipedia (Dandala et al., 2013) sense inventories to future work.

6 Related Work

The past few years have witnessed a growing research interest in semantic representation, mainly as a consequence of the word embedding tsunami

(Mikolov et al., 2013; Pennington et al., 2014). Soon after their introduction, word embeddings were integrated into different NLP applications, thanks to the migration of the field to deep learning and the fact that most deep learning models view words as dense vectors. The waves of the word embedding tsunami have also lapped on the shores of sense representation. Several techniques have been proposed that either extend word embedding models to cluster contexts and induce senses, usually referred to as unsupervised sense representations (Schütze, 1998; Reisinger and Mooney, 2010; Huang et al., 2012; Neelakantan et al., 2014; Guo et al., 2014; Tian et al., 2014; Šuster et al., 2016; Ettinger et al., 2016; Qiu et al., 2016) or exploit external sense inventories and lexical resources for generating sense representations for individual meanings of words (Chen et al., 2014; Johansson and Pina, 2015; Jauhar et al., 2015; Iacobacci et al., 2015; Rothe and Schütze, 2015; Camacho-Collados et al., 2016b; Mancini et al., 2016; Pilehvar and Collier, 2016).

However, the integration of sense representations into deep learning models has not been so straightforward, and research in this field has often opted for alternative evaluation benchmarks such as WSD, or artificial tasks, such as word similarity. Consequently, the problem of integrating sense representations into downstream NLP applications has remained understudied, despite the potential benefits it can have. Li and Jurafsky (2015) proposed a "multi-sense embedding" pipeline to check the benefit that can be gained by replacing word embeddings with sense embeddings in multiple tasks. With the help of two simple disambiguation algorithms, unsupervised sense embeddings were integrated into various downstream applications, with varying degrees of success. Given the interdependency of sense representation and disambiguation in this model, it is very difficult to introduce alternative algorithms into its pipeline, either to benefit from the state of the art, or to carry out an evaluation. Instead, our pipeline provides the advantage of being modular: thanks to its use of disambiguation in the pre-processing stage and use of sense representations that are linked to external sense inventories, different WSD techniques and sense representations can be easily plugged in and checked. Along the same lines, Flekova and Gurevych (2016) proposed a technique for learning supersense representations, using automatically-annotated corpora. Coupled with a supersense tagger, the representations were fed into a neural network classifier as additional features to the word-based input. Through a set of experiments, Flekova and Gurevych (2016) showed that the supersense enrichment can be beneficial to a range of binary classification tasks. Our proposal is different in that it focuses directly on the benefits that can be gained by semantifying the input, i.e. reducing lexical ambiguity in the input text, rather than assisting the model with additional sources of knowledge.

7 Conclusion and Future Work

We proposed a pipeline for the integration of sense level knowledge into a state-of-the-art text classifier. We showed that a simple disambiguation of the input can lead to consistent performance gain, particularly for longer documents and when the granularity of the underlying sense inventory is reduced. Our pipeline is modular and can be used as an *in vivo* evaluation framework for WSD and sense representation techniques. We release our code and data (including pre-trained sense and supersense embeddings) at https://pilehvar.github.io/sensecnn/ to allow further checking of the choice of hyperparameters and to allow further analysis and comparison. We hope that our work will foster future research on the integration of sense-level knowledge into downstream applications. As future work, we plan to investigate the extension of the approach to other languages and applications. Also, given the promising results observed for supersenses, we plan to investigate task-specific coarsening of sense inventories, particularly Wikipedia, or the use of SentiWordNet (Baccianella et al., 2010), which could be more suitable for polarity detection.

Acknowledgments

The authors gratefully acknowledge the support of the MRC grant No. MR/M025160/1 for PheneBank and ERC Consolidator Grant MOUSSE No. 726487. Jose Camacho-Collados is supported by a Google Doctoral Fellowship in Natural Language Processing. Nigel Collier is supported by EPSRC Grant No. EP/M005089/1. We thank Jim McManus for his suggestions on the manuscript and the anonymous reviewers for their helpful comments.

References

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics* 40(1):57–84.

Eneko Agirre and Oier Lopez. 2003. Clustering WordNet word senses. In *Proceedings of Recent Advances in Natural Language Processing*. Borovets, Bulgaria, pages 121–130.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*. volume 10, pages 2200–2204.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of EMNLP*.

Sumit Bhagwani, Shrutiranjan Satapathy, and Harish Karnick. 2013. Merging word senses. In *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing*. Seattle, Washington, USA, pages 11–19.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *EMNLP*.

José Camacho-Collados, Claudio Delli Bovi, Alessandro Raganato, and Roberto Navigli. 2016a. A Large-Scale Multilingual Disambiguation of Glosses. In *Proceedings of LREC*. Portoroz, Slovenia, pages 1701–1708.

Jose Camacho-Collados and Roberto Navigli. 2017. BabelDomains: Large-Scale Domain Labeling of Lexical Resources. In *Proceedings of EACL (2)*. Valencia, Spain.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016b. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence* 240:36–64.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*. Doha, Qatar, pages 1025–1035.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proceedings of EMNLP*. Seattle, Washington, pages 1787–1796.

Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the Workshop on Evaluating Vector Space Representations for NLP, ACL*.

Franois Chollet. 2015. Keras. https://github.com/fchollet/keras.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* 12:2493–2537.

Bharath Dandala, Chris Hokamp, Rada Mihalcea, and Razvan C. Bunescu. 2013. Sense clustering using Wikipedia. In *Proceedings of Recent Advances in Natural Language Processing*. Hissar, Bulgaria, pages 164–171.

Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. 2015. A statistical parsing framework for sentiment classification. *Computational Linguistics* 41(2):293–336.

Luis Espinosa-Anke, Jose Camacho-Collados, Claudio Delli Bovi, and Horacio Saggion. 2016. Supervised distributional hypernym discovery via domain adaptation. In *Proceedings of EMNLP*. pages 424–435.

Allyson Ettinger, Philip Resnik, and Marine Carpuat. 2016. Retrofitting sense-specific word vectors using parallel text. In *Proceedings of NAACL-HLT*. San Diego, California, pages 1378–1383.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Lucie Flekova and Iryna Gurevych. 2016. Supersense embeddings: A unified model for supersense interpretation, prediction, and utilization. In *Proceedings of ACL*.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd International conference on Machine learning*. ACM, pages 377–384.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *COLING*. pages 497–507.

Taher H. Haveliwala. 2002. Topic-sensitive PageRank. In *Proceedings of the 11th International Conference on World Wide Web*. Hawaii, USA, pages 517–526.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580.

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

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of CIKM*. pages 545–554.

Eduard H. Hovy, Roberto Navigli, and Simone Paolo Ponzetto. 2013. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence* 194:2–27.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*. Jeju Island, Korea, pages 873–882.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of ACL*. Beijing, China, pages 95–105.

Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of NAACL*. Denver, Colorado, pages 683–693.

Richard Johansson and Luis Nieto Pina. 2015. Embedding a semantic network in a word space. In *Proceedings of NAACL*. Denver, Colorado, pages 1428–1433.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of NAACL*. Denver, Colorado, pages 103–112.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*. Baltimore, USA, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*. Doha, Qatar, pages 1746–1751.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI*. Phoenix, Arizona, pages 2741–2749.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*. Tahoe City, California, pages 331–339.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of EMNLP*. Lisbon, Portugal, pages 683–693.

Antonio Lieto, Enrico Mensa, and Daniele P Radicioni. 2016. A resource-driven approach for anchoring linguistic resources to conceptual spaces. In *AI* IA 2016 Advances in Artificial Intelligence*, Springer, pages 435–449.

Nut Limsopatham and Nigel Collier. 2016. Normalising medical concepts in social media texts by learning semantic representation. In *Proceedings of ACL*. Berlin, Germany, pages 1014–1023.

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics* 3:315–328.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL-HLT*. Portland, Oregon, USA, pages 142–150.

Massimiliano Mancini, José Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2016. Embedding words and senses together via joint knowledge-enhanced training. *CoRR* abs/1612.02703. http://arxiv.org/abs/1612.02703.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany, pages 51–61.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

George A Miller. 1995. WordNet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics (TACL)* 2:231–244.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*. pages 807–814.

Roberto Navigli. 2006. Meaningful clustering of senses helps boost Word Sense Disambiguation performance. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*. Sydney, Australia, pages 105–112.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193:217–250.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*. Doha, Qatar, pages 1059–1069.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*. Barcelona, Spain, pages 51–61.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*. Ann Arbor, Michigan, pages 115–124.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*. pages 1532–1543.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. In *Proceedings of EMNLP*. Austin, TX, pages 1680–1690.

Mohammad Taher Pilehvar and Roberto Navigli. 2014. A large-scale pseudoword-based evaluation framework for state-of-the-art Word Sense Disambiguation. *Computational Linguistics* 40(4).

Lin Qiu, Kewei Tu, and Yong Yu. 2016. Context-dependent sense embedding. In *Proceedings of EMNLP*. Austin, Texas, pages 183–191.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of EACL*. Valencia, Spain, pages 99–110.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proceedings of ACL*. pages 109–117.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of ACL*. Beijing, China, pages 1793–1803.

Stefan Rüd, Massimiliano Ciaramita, Jens Müller, and Hinrich Schütze. 2011. Piggyback: Using search engines for robust cross-domain named entity recognition. In *Proceedings of ACL-HLT*. Portland, Oregon, USA, pages 965–975.

Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Mexico City, Mexico, pages 1–15.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *NAACL-HTL*. Denver, Colorado, pages 977–983.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational linguistics* 24(1):97–123.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning semantic textual similarity with structural representations. In *Proceedings of ACL (2)*. Sofia, Bulgaria, pages 714–718.

Rion Snow, Sushant Prakash, Daniel Jurafsky, and Andrew Y. Ng. 2007. Learning to merge word senses. In *Proceedings of EMNLP*. Prague, Czech Republic, pages 1005–1014.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Parsing with compositional vector grammars. In *Proceedings of EMNLP*. Sofia, Bulgaria, pages 455–465.

Simon Šuster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. In *Proceedings of NAACL-HLT*. San Diego, California, pages 1346–1356.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Porceedings of EMNLP*. Lisbon, Portugal, pages 1422–1432.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*. pages 151–160.

Rocco Tripodi and Marcello Pelillo. 2017. A game-theoretic approach to word sense disambiguation. *Computational Linguistics* 43(1):31–70.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of EMNLP (2)*. Lisbon, Portugal, pages 2049–2054.

Yulia Tsvetkov and Shuly Wintner. 2014. Identification of multiword expressions by combining multiple linguistic information sources. *Computational Linguistics* 40(2):449–468.

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

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *CoRR* abs/1602.00367.

Shweta Yadav, Asif Ekbal, Sriparna Saha, and Pushpak Bhattacharyya. 2017. Entity extraction in biomedical corpora: An approach to evaluate word embedding features with pso based feature selection. In

Proceedings of EACL. Valencia, Spain, pages 1159–1170.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Information retrieval* 1(1-2):69–90.

Jay Young, Valerio Basile, Lars Kunze, Elena Cabrio, and Nick Hawes. 2016. Towards lifelong object learning by integrating situated robot perception and semantic web mining. In *Proceedings of the European Conference on Artificial Intelligence conference*. The Hague, Netherland, pages 1458–1466.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A wide-coverage Word Sense Disambiguation system for free text. In *Proceedings of the ACL System Demonstrations*. Uppsala, Sweden, pages 78–83.

Will Y. Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of EMNLP*. Seattle, USA, pages 1393–1398.

Reading Wikipedia to Answer Open-Domain Questions

Danqi Chen*
Computer Science
Stanford University
Stanford, CA 94305, USA
danqi@cs.stanford.edu

Adam Fisch, Jason Weston & Antoine Bordes
Facebook AI Research
770 Broadway
New York, NY 10003, USA
{afisch,jase,abordes}@fb.com

Abstract

This paper proposes to tackle open-domain question answering using Wikipedia as the unique knowledge source: the answer to any factoid question is a text span in a Wikipedia article. This task of *machine reading at scale* combines the challenges of document retrieval (finding the relevant articles) with that of machine comprehension of text (identifying the answer spans from those articles). Our approach combines a search component based on bigram hashing and TF-IDF matching with a multi-layer recurrent neural network model trained to detect answers in Wikipedia paragraphs. Our experiments on multiple existing QA datasets indicate that (1) both modules are highly competitive with respect to existing counterparts and (2) multitask learning using distant supervision on their combination is an effective complete system on this challenging task.

1 Introduction

This paper considers the problem of answering factoid questions in an open-domain setting using Wikipedia as the unique knowledge source, such as one does when looking for answers in an encyclopedia. Wikipedia is a constantly evolving source of detailed information that could facilitate intelligent machines — if they are able to leverage its power. Unlike knowledge bases (KBs) such as Freebase (Bollacker et al., 2008) or DBPedia (Auer et al., 2007), which are easier for computers to process but too sparsely populated for open-domain question answering (Miller et al., 2016), Wikipedia contains up-to-date knowledge that humans are interested in. It is designed, however, for humans – not machines – to read.

Using Wikipedia articles as the knowledge source causes the task of question answering (QA) to combine the challenges of both large-scale open-domain QA and of machine comprehension of text. In order to answer any question, one must first retrieve the few relevant articles among more than 5 million items, and then scan them carefully to identify the answer. We term this setting, *machine reading at scale* (MRS). Our work treats Wikipedia as a collection of articles and does not rely on its internal graph structure. As a result, our approach is generic and could be switched to other collections of documents, books, or even daily updated newspapers.

Large-scale QA systems like IBM's DeepQA (Ferrucci et al., 2010) rely on multiple sources to answer: besides Wikipedia, it is also paired with KBs, dictionaries, and even news articles, books, etc. As a result, such systems heavily rely on information redundancy among the sources to answer correctly. Having a single knowledge source forces the model to be very precise while searching for an answer as the evidence might appear only once. This challenge thus encourages research in the ability of a machine to read, a key motivation for the machine comprehension subfield and the creation of datasets such as SQuAD (Rajpurkar et al., 2016), CNN/Daily Mail (Hermann et al., 2015) and CBT (Hill et al., 2016).

However, those machine comprehension resources typically assume that a short piece of relevant text is already identified and given to the model, which is not realistic for building an open-domain QA system. In sharp contrast, methods that use KBs or information retrieval over documents have to employ search as an integral part of

* Most of this work was done while DC was with Facebook AI Research.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1870–1879
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1171

the solution. Instead MRS is focused on simultaneously maintaining the challenge of machine comprehension, which requires the deep understanding of text, while keeping the realistic constraint of searching over a large open resource.

In this paper, we show how multiple existing QA datasets can be used to evaluate MRS by requiring an open-domain system to perform well on all of them at once. We develop DrQA, a strong system for question answering from Wikipedia composed of: (1) Document Retriever, a module using bigram hashing and TF-IDF matching designed to, given a question, efficiently return a subset of relevant articles and (2) Document Reader, a multi-layer recurrent neural network machine comprehension model trained to detect answer spans in those few returned documents. Figure 1 gives an illustration of DrQA.

Our experiments show that Document Retriever outperforms the built-in Wikipedia search engine and that Document Reader reaches state-of-the-art results on the very competitive SQuAD benchmark (Rajpurkar et al., 2016). Finally, our full system is evaluated using multiple benchmarks. In particular, we show that performance is improved across all datasets through the use of multitask learning and distant supervision compared to single task training.

2 Related Work

Open-domain QA was originally defined as finding answers in collections of unstructured documents, following the setting of the annual TREC competitions[1]. With the development of KBs, many recent innovations have occurred in the context of QA from KBs with the creation of resources like WebQuestions (Berant et al., 2013) and SimpleQuestions (Bordes et al., 2015) based on the Freebase KB (Bollacker et al., 2008), or on automatically extracted KBs, e.g., OpenIE triples and NELL (Fader et al., 2014). However, KBs have inherent limitations (incompleteness, fixed schemas) that motivated researchers to return to the original setting of answering from raw text.

A second motivation to cast a fresh look at this problem is that of machine comprehension of text, i.e., answering questions after reading a short text or story. That subfield has made considerable progress recently thanks to new deep learning architectures like attention-based and memory-augmented neural networks (Bahdanau et al., 2015; Weston et al., 2015; Graves et al., 2014) and release of new training and evaluation datasets like QuizBowl (Iyyer et al., 2014), CNN/Daily Mail based on news articles (Hermann et al., 2015), CBT based on children books (Hill et al., 2016), or SQuAD (Rajpurkar et al., 2016) and WikiReading (Hewlett et al., 2016), both based on Wikipedia. An objective of this paper is to test how such new methods can perform in an open-domain QA framework.

QA using Wikipedia as a resource has been explored previously. Ryu et al. (2014) perform open-domain QA using a Wikipedia-based knowledge model. They combine article content with multiple other answer matching modules based on different types of semi-structured knowledge such as infoboxes, article structure, category structure, and definitions. Similarly, Ahn et al. (2004) also combine Wikipedia as a text resource with other resources, in this case with information retrieval over other documents. Buscaldi and Rosso (2006) also mine knowledge from Wikipedia for QA. Instead of using it as a resource for seeking answers to questions, they focus on validating answers returned by their QA system, and use Wikipedia categories for determining a set of patterns that should fit with the expected answer. In our work, we consider the comprehension of text only, and use Wikipedia text documents as the sole resource in order to emphasize the task of machine reading at scale, as described in the introduction.

There are a number of highly developed full pipeline QA approaches using either the Web, as does QuASE (Sun et al., 2015), or Wikipedia as a resource, as do Microsoft's AskMSR (Brill et al., 2002), IBM's DeepQA (Ferrucci et al., 2010) and YodaQA (Baudiš, 2015; Baudiš and Šedivỳ, 2015) — the latter of which is open source and hence reproducible for comparison purposes. AskMSR is a search-engine based QA system that relies on "data redundancy rather than sophisticated linguistic analyses of either questions or candidate answers", i.e., it does not focus on machine comprehension, as we do. DeepQA is a very sophisticated system that relies on both unstructured information including text documents as well as structured data such as KBs, databases and ontologies to generate candidate answers or vote over evidence. YodaQA is an open source system modeled after DeepQA, similarly combining websites,

[1] http://trec.nist.gov/data/qamain.html

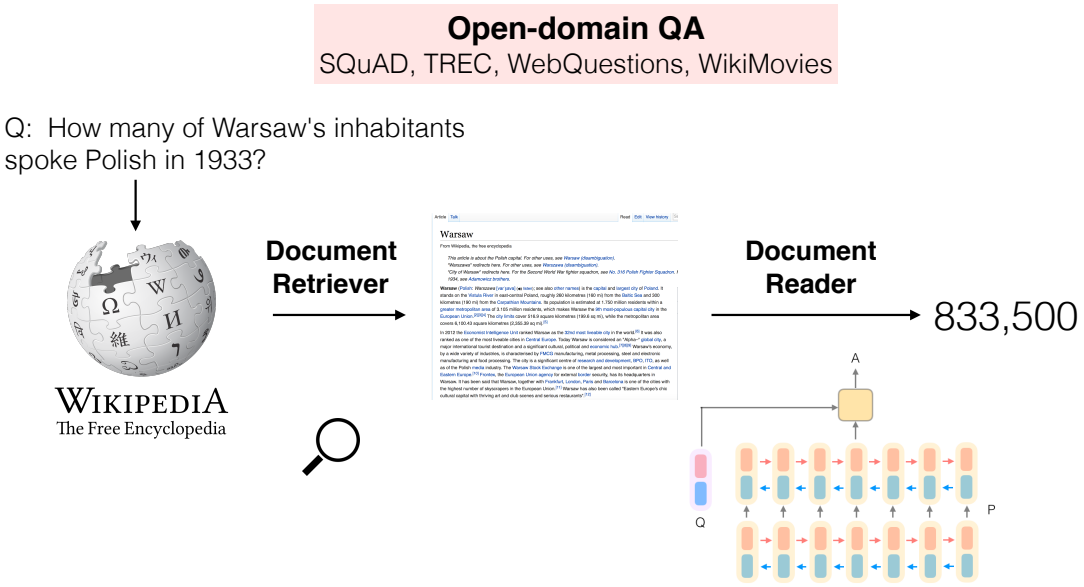


Figure 1: An overview of our question answering system DrQA.

information extraction, databases and Wikipedia in particular. Our comprehension task is made more challenging by only using a single resource. Comparing against these methods provides a useful datapoint for an "upper bound" benchmark on performance.

Multitask learning (Caruana, 1998) and task transfer have a rich history in machine learning (e.g., using ImageNet in the computer vision community (Huh et al., 2016)), as well as in NLP in particular (Collobert and Weston, 2008). Several works have attempted to combine multiple QA training datasets via multitask learning to (i) achieve improvement across the datasets via task transfer; and (ii) provide a single general system capable of asking different kinds of questions due to the inevitably different data distributions across the source datasets. Fader et al. (2014) used WebQuestions, TREC and WikiAnswers with four KBs as knowledge sources and reported improvement on the latter two datasets through multitask learning. Bordes et al. (2015) combined WebQuestions and SimpleQuestions using distant supervision with Freebase as the KB to give slight improvements on both datasets, although poor performance was reported when training on only one dataset and testing on the other, showing that task transfer is indeed a challenging subject; see also (Kadlec et al., 2016) for a similar conclusion. Our work follows similar themes, but in the setting of having to retrieve and then read text documents, rather than using a KB, with positive results.

3 Our System: DrQA

In the following we describe our system DrQA for MRS which consists of two components: (1) the Document Retriever module for finding relevant articles and (2) a machine comprehension model, Document Reader, for extracting answers from a single document or a small collection of documents.

3.1 Document Retriever

Following classical QA systems, we use an efficient (non-machine learning) document retrieval system to first narrow our search space and focus on reading only articles that are likely to be relevant. A simple inverted index lookup followed by term vector model scoring performs quite well on this task for many question types, compared to the built-in ElasticSearch based Wikipedia Search API (Gormley and Tong, 2015). Articles and questions are compared as TF-IDF weighted bag-of-word vectors. We further improve our system by taking local word order into account with n-gram features. Our best performing system uses bigram counts while preserving speed and memory efficiency by using the hashing of (Weinberger et al., 2009) to map the bigrams to 2^{24} bins with an unsigned *murmur3* hash.

We use Document Retriever as the first part of our full model, by setting it to return 5 Wikipedia

articles given any question. Those articles are then processed by Document Reader.

3.2 Document Reader

Our Document Reader model is inspired by the recent success of neural network models on machine comprehension tasks, in a similar spirit to the *AttentiveReader* described in (Hermann et al., 2015; Chen et al., 2016).

Given a question q consisting of l tokens $\{q_1, \ldots, q_l\}$ and a document or a small set of documents of n paragraphs where a single paragraph p consists of m tokens $\{p_1, \ldots, p_m\}$, we develop an RNN model that we apply to each paragraph in turn and then finally aggregate the predicted answers. Our method works as follows:

Paragraph encoding We first represent all tokens p_i in a paragraph p as a sequence of feature vectors $\tilde{\mathbf{p}}_i \in \mathbb{R}^d$ and pass them as the input to a recurrent neural network and thus obtain:

$$\{\mathbf{p}_1, \ldots, \mathbf{p}_m\} = \text{RNN}(\{\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_m\}),$$

where $\mathbf{p}_i$ is expected to encode useful context information around token p_i. Specifically, we choose to use a multi-layer bidirectional long short-term memory network (LSTM), and take $\mathbf{p}_i$ as the concatenation of each layer's hidden units in the end.

The feature vector $\tilde{\mathbf{p}}_i$ is comprised of the following parts:

- *Word embeddings*: $f_{emb}(p_i) = \mathbf{E}(p_i)$. We use the 300-dimensional Glove word embeddings trained from 840B Web crawl data (Pennington et al., 2014). We keep most of the pre-trained word embeddings fixed and only fine-tune the 1000 most frequent question words because the representations of some key words such as *what*, *how*, *which*, *many* could be crucial for QA systems.

- *Exact match*: $f_{exact\_match}(p_i) = \mathbb{I}(p_i \in q)$. We use three simple binary features, indicating whether p_i can be exactly matched to one question word in q, either in its original, lowercase or lemma form. These simple features turn out to be extremely helpful, as we will show in Section 5.

- *Token features*: $f_{token}(p_i) = (\text{POS}(p_i), \text{NER}(p_i), \text{TF}(p_i))$. We also add a few manual features which reflect some properties of token p_i in its context, which include its part-of-speech (POS) and named entity recognition (NER) tags and its (normalized) term frequency (TF).

- *Aligned question embedding*: Following (Lee et al., 2016) and other recent works, the last part we incorporate is an aligned question embedding $f_{align}(p_i) = \sum_j a_{i,j}\mathbf{E}(q_j)$, where the attention score $a_{i,j}$ captures the similarity between p_i and each question words q_j. Specifically, $a_{i,j}$ is computed by the dot products between nonlinear mappings of word embeddings:

$$a_{i,j} = \frac{\exp\left(\alpha(\mathbf{E}(p_i)) \cdot \alpha(\mathbf{E}(q_j))\right)}{\sum_{j'} \exp\left(\alpha(\mathbf{E}(p_i)) \cdot \alpha(\mathbf{E}(q_{j'}))\right)},$$

and $\alpha(\cdot)$ is a single dense layer with ReLU nonlinearity. Compared to the *exact match* features, these features add soft alignments between similar but non-identical words (e.g., *car* and *vehicle*).

Question encoding The question encoding is simpler, as we only apply another recurrent neural network on top of the word embeddings of q_i and combine the resulting hidden units into one single vector: $\{\mathbf{q}_1, \ldots, \mathbf{q}_l\} \rightarrow \mathbf{q}$. We compute $\mathbf{q} = \sum_j b_j \mathbf{q}_j$ where b_j encodes the importance of each question word:

$$b_j = \frac{\exp(\mathbf{w} \cdot \mathbf{q}_j)}{\sum_{j'} \exp(\mathbf{w} \cdot \mathbf{q}_{j'})},$$

and $\mathbf{w}$ is a weight vector to learn.

Prediction At the paragraph level, the goal is to predict the span of tokens that is most likely the correct answer. We take the the paragraph vectors $\{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$ and the question vector $\mathbf{q}$ as input, and simply train two classifiers independently for predicting the two ends of the span. Concretely, we use a bilinear term to capture the similarity between $\mathbf{p}_i$ and $\mathbf{q}$ and compute the probabilities of each token being start and end as:

$$\begin{aligned} P_{start}(i) &\propto \exp\left(\mathbf{p}_i \mathbf{W}_s \mathbf{q}\right) \\ P_{end}(i) &\propto \exp\left(\mathbf{p}_i \mathbf{W}_e \mathbf{q}\right) \end{aligned}$$

During prediction, we choose the best span from token i to token i' such that $i \leq i' \leq i + 15$ and $P_{start}(i) \times P_{end}(i')$ is maximized. To make scores

compatible across paragraphs in one or several retrieved documents, we use the unnormalized exponential and take argmax over all considered paragraph spans for our final prediction.

4 Data

Our work relies on three types of data: (1) Wikipedia that serves as our knowledge source for finding answers, (2) the SQuAD dataset which is our main resource to train Document Reader and (3) three more QA datasets (CuratedTREC, WebQuestions and WikiMovies) that in addition to SQuAD, are used to test the open-domain QA abilities of our full system, and to evaluate the ability of our model to learn from multitask learning and distant supervision. Statistics of the datasets are given in Table 2.

4.1 Wikipedia (Knowledge Source)

We use the 2016-12-21 dump[2] of English Wikipedia for all of our full-scale experiments as the knowledge source used to answer questions. For each page, only the plain text is extracted and all structured data sections such as lists and figures are stripped.[3] After discarding internal disambiguation, list, index, and outline pages, we retain 5,075,182 articles consisting of 9,008,962 unique uncased token types.

4.2 SQuAD

The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) is a dataset for machine comprehension based on Wikipedia. The dataset contains 87k examples for training and 10k for development, with a large hidden test set which can only be accessed by the SQuAD creators. Each example is composed of a paragraph extracted from a Wikipedia article and an associated human-generated question. The answer is always a span from this paragraph and a model is given credit if its predicted answer matches it. Two evaluation metrics are used: exact string match (EM) and F1 score, which measures the weighted average of precision and recall at the token level.

In the following, we use SQuAD for training and evaluating our Document Reader for the standard machine comprehension task given the relevant paragraph as defined in (Rajpurkar et al., 2016). For the task of evaluating open-domain question answering over Wikipedia, we use the SQuAD development set QA pairs only, and we ask systems to uncover the correct answer spans *without* having access to the associated paragraphs. That is, a model is required to answer a question given the whole of Wikipedia as a resource; it is *not* given the relevant paragraph as in the standard SQuAD setting.

4.3 Open-domain QA Evaluation Resources

SQuAD is one of the largest general purpose QA datasets currently available. SQuAD questions have been collected via a process involving showing a paragraph to each human annotator and asking them to write a question. As a result, their distribution is quite specific. We hence propose to train and evaluate our system on other datasets developed for open-domain QA that have been constructed in different ways (not necessarily in the context of answering from Wikipedia).

CuratedTREC This dataset is based on the benchmarks from the TREC QA tasks that have been curated by Baudiš and Šedivỳ (2015). We use the large version, which contains a total of 2,180 questions extracted from the datasets from TREC 1999, 2000, 2001 and 2002.[4]

WebQuestions Introduced in (Berant et al., 2013), this dataset is built to answer questions from the Freebase KB. It was created by crawling questions through the Google Suggest API, and then obtaining answers using Amazon Mechanical Turk. We convert each answer to text by using entity names so that the dataset does not reference Freebase IDs and is purely made of plain text question-answer pairs.

WikiMovies This dataset, introduced in (Miller et al., 2016), contains 96k question-answer pairs in the domain of movies. Originally created from the OMDb and MovieLens databases, the examples are built such that they can also be answered by using a subset of Wikipedia as the knowledge source (the title and the first section of articles from the movie domain).

[2] https://dumps.wikimedia.org/enwiki/latest

[3] We use the WikiExtractor script: https://github.com/attardi/wikiextractor.

[4] This dataset is available at https://github.com/brmson/dataset-factoid-curated.

Dataset	Example	Article / Paragraph
SQuAD	**Q**: How many provinces did the Ottoman empire contain in the 17th century? **A**: 32	**Article**: Ottoman Empire **Paragraph**: ... At the beginning of the 17th century the empire contained 32 provinces and numerous vassal states. Some of these were later absorbed into the Ottoman Empire, while others were granted various types of autonomy during the course of centuries.
CuratedTREC	**Q**: What U.S. state's motto is "Live free or Die"? **A**: New Hampshire	**Article**: Live Free or Die **Paragraph**: "Live Free or Die" is the official motto of the U.S. state of New Hampshire, adopted by the state in 1945. It is possibly the best-known of all state mottos, partly because it conveys an assertive independence historically found in American political philosophy and partly because of its contrast to the milder sentiments found in other state mottos.
WebQuestions	**Q**: What part of the atom did Chadwick discover?† **A**: neutron	**Article**: Atom **Paragraph**: ... The atomic mass of these isotopes varied by integer amounts, called the whole number rule. The explanation for these different isotopes awaited the discovery of the neutron, an uncharged particle with a mass similar to the proton, by the physicist James Chadwick in 1932. ...
WikiMovies	**Q**: Who wrote the film Gigli? **A**: Martin Brest	**Article**: Gigli **Paragraph**: Gigli is a 2003 American romantic comedy film written and directed by Martin Brest and starring Ben Affleck, Jennifer Lopez, Justin Bartha, Al Pacino, Christopher Walken, and Lainie Kazan.

Table 1: Example training data from each QA dataset. In each case we show an associated paragraph where distant supervision (DS) correctly identified the answer within it, which is highlighted.

Dataset	Train		Test
	Plain	DS	
SQuAD	87,599	71,231	10,570†
CuratedTREC	1,486*	3,464	694
WebQuestions	3,778*	4,602	2,032
WikiMovies	96,185*	36,301	9,952

Table 2: Number of questions for each dataset used in this paper. DS: distantly supervised training data. *: These training sets are not used as is because no paragraph is associated with each question. †: Corresponds to SQuAD development set.

Dataset	Wiki	Doc. Retriever	
	Search	plain	+bigrams
SQuAD	62.7	76.1	**77.8**
CuratedTREC	81.0	85.2	**86.0**
WebQuestions	73.7	**75.5**	74.4
WikiMovies	61.7	54.4	**70.3**

Table 3: Document retrieval results. % of questions for which the answer segment appears in one of the top 5 pages returned by the method.

4.4 Distantly Supervised Data

All the QA datasets presented above contain training portions, but CuratedTREC, WebQuestions and WikiMovies only contain question-answer pairs, and not an associated document or paragraph as in SQuAD, and hence cannot be used for training Document Reader directly. Following previous work on distant supervision (DS) for relation extraction (Mintz et al., 2009), we use a procedure to automatically associate paragraphs to such training examples, and then add these examples to our training set.

We use the following process for each question-answer pair to build our training set. First, we run Document Retriever on the question to retrieve the top 5 Wikipedia articles. All paragraphs from those articles without an exact match of the known answer are directly discarded. All paragraphs shorter than 25 or longer than 1500 characters are also filtered out. If any named entities are detected in the question, we remove any paragraph that does not contain them at all. For every remaining paragraph in each retrieved page, we score all positions that match an answer using unigram and bigram overlap between the question and a 20 token window, keeping up to the top 5 paragraphs with the highest overlaps. If there is no paragraph with non-zero overlap, the example is discarded; otherwise we add each found pair to our DS training dataset. Some examples are shown in Table 1 and data statistics are given in Table 2.

Note that we can also generate additional DS data for SQuAD by trying to find mentions of the answers not just in the paragraph provided, but also from other pages or the same page that the given paragraph was in. We observe that around half of the DS examples come from pages outside of the articles used in SQuAD.

5 Experiments

This section first presents evaluations of our Document Retriever and Document Reader modules separately, and then describes tests of their combination, DrQA, for open-domain QA on the full Wikipedia.

5.1 Finding Relevant Articles

We first examine the performance of our Document Retriever module on all the QA datasets. Table 3 compares the performance of the two approaches described in Section 3.1 with that of the Wikipedia Search Engine[5] for the task of finding articles that contain the answer given a question. Specifically, we compute the ratio of questions for which the text span of any of their associated answers appear in at least one the top 5 relevant pages returned by each system. Results on all datasets indicate that our simple approach outperforms Wikipedia Search, especially with bigram hashing. We also compare doing retrieval with Okapi BM25 or by using cosine distance in the word embeddings space (by encoding questions and articles as bag-of-embeddings), both of which we find performed worse.

5.2 Reader Evaluation on SQuAD

Next we evaluate our Document Reader component on the standard SQuAD evaluation (Rajpurkar et al., 2016).

Implementation details We use 3-layer bidirectional LSTMs with $h = 128$ hidden units for both paragraph and question encoding. We apply the Stanford CoreNLP toolkit (Manning et al., 2014) for tokenization and also generating lemma, part-of-speech, and named entity tags.

Lastly, all the training examples are sorted by the length of paragraph and divided into mini-batches of 32 examples each. We use *Adamax* for optimization as described in (Kingma and Ba, 2014). Dropout with $p = 0.3$ is applied to word embeddings and all the hidden units of LSTMs.

Result and analysis Table 4 presents our evaluation results on both development and test sets. SQuAD has been a very competitive machine comprehension benchmark since its creation and we only list the best-performing systems in the table. Our system (single model) can achieve 70.0% exact match and 79.0% F1 scores on the test set, which surpasses all the published results and can match the top performance on the SQuAD leaderboard at the time of writing. Additionally, we think that our model is conceptually simpler than most of the existing systems. We conducted an ablation analysis on the feature vector of paragraph tokens. As shown in Table 5 all the features contribute to the performance of our final system. Without the aligned question embedding feature (only word embedding and a few manual features), our system is still able to achieve F1 over 77%. More interestingly, if we remove both $f_{aligned}$ and f_{exact_match}, the performance drops dramatically, so we conclude that both features play a similar but complementary role in the feature representation related to the paraphrased nature of a question vs. the context around an answer.

5.3 Full Wikipedia Question Answering

Finally, we assess the performance of our full system DrQA for answering open-domain questions using the four datasets introduced in Section 4. We compare three versions of DrQA which evaluate the impact of using distant supervision and multitask learning across the training sources provided to Document Reader (Document Retriever remains the same for each case):

- SQuAD: A single Document Reader model is trained on the SQuAD training set only and used on all evaluation sets.
- Fine-tune (DS): A Document Reader model is pre-trained on SQuAD and then fine-tuned for each dataset independently using its distant supervision (DS) training set.
- Multitask (DS): A single Document Reader model is jointly trained on the SQuAD training set and *all* the DS sources.

For the full Wikipedia setting we use a streamlined model that does not use the CoreNLP parsed f_{token} features or lemmas for f_{exact_match}. We

[5] We use the Wikipedia Search API `https://www.mediawiki.org/wiki/API:Search`.

Method	Dev		Test	
	EM	F1	EM	F1
Dynamic Coattention Networks (Xiong et al., 2016)	65.4	75.6	66.2	75.9
Multi-Perspective Matching (Wang et al., 2016)†	66.1	75.8	65.5	75.1
BiDAF (Seo et al., 2016)	67.7	77.3	68.0	77.3
R-net†	n/a	n/a	71.3	79.7
DrQA (Our model, Document Reader Only)	**69.5**	**78.8**	70.0	79.0

Table 4: Evaluation results on the SQuAD dataset (single model only). †: Test results reflect the SQuAD leaderboard (https://stanford-qa.com) as of Feb 6, 2017.

Features	F1
Full	78.8
No f_{token}	78.0 (-0.8)
No f_{exact_match}	77.3 (-1.5)
No $f_{aligned}$	77.3 (-1.5)
No $f_{aligned}$ and f_{exact_match}	59.4 (-19.4)

Table 5: Feature ablation analysis of the paragraph representations of our Document Reader. Results are reported on the SQuAD development set.

find that while these help for more exact paragraph reading in SQuAD, they don't improve results in the full setting. Additionally, WebQuestions and WikiMovies provide a list of candidate answers (e.g., 1.6 million Freebase entity strings for WebQuestions) and we restrict the answer span must be in this list during prediction.

Results Table 6 presents the results. Despite the difficulty of the task compared to machine comprehension (where you are given the right paragraph) and unconstrained QA (using redundant resources), DrQA still provides reasonable performance across all four datasets.

We are interested in a single, full system that can answer any question using Wikipedia. The single model trained only on SQuAD is outperformed on all four of the datasets by the multitask model that uses distant supervision. However performance when training on SQuAD alone is not far behind, indicating that task transfer is occurring. The majority of the improvement from SQuAD to Multitask (DS) however is likely not from task transfer as fine-tuning on each dataset alone using DS also gives improvements, showing that is is the introduction of extra data in the same domain that helps. Nevertheless, the best single model that we can find is our overall goal, and that is the Multitask (DS) system.

We compare to an unconstrained QA system using redundant resources (not just Wikipedia), YodaQA (Baudiš, 2015), giving results which were previously reported on CuratedTREC and WebQuestions. Despite the increased difficulty of our task, it is reassuring that our performance is not too far behind on CuratedTREC (31.3 vs. 25.4). The gap is slightly bigger on WebQuestions, likely because this dataset was created from the specific structure of Freebase which YodaQA uses directly.

DrQA's performance on SQuAD compared to its Document Reader component on machine comprehension in Table 4 shows a large drop (from 69.5 to 27.1) as we now are given Wikipedia to read, not a single paragraph. Given the correct document (but not the paragraph) we can achieve 49.4, indicating many false positives come from highly topical sentences. This is despite the fact that the Document Retriever works relatively well (77.8% of the time retrieving the answer, see Table 3). It is worth noting that a large part of the drop comes from the nature of the SQuAD questions. They were written with a specific paragraph in mind, thus their language can be ambiguous when the context is removed. Additional resources other than SQuAD, specifically designed for MRS, might be needed to go further.

6 Conclusion

We studied the task of machine reading at scale, by using Wikipedia as the unique knowledge source for open-domain QA. Our results indicate that MRS is a key challenging task for researchers to focus on. Machine comprehension systems alone cannot solve the overall task. Our method integrates search, distant supervision, and multitask learning to provide an effective complete system. Evaluating the individual components as well as the full system across multiple benchmarks showed the efficacy of our approach.

Dataset	YodaQA	DrQA		
		SQuAD	+Fine-tune (DS)	+Multitask (DS)
SQuAD *(All Wikipedia)*	n/a	27.1	28.4	29.8
CuratedTREC	31.3	19.7	25.7	25.4
WebQuestions	39.8	11.8	19.5	20.7
WikiMovies	n/a	24.5	34.3	36.5

Table 6: Full Wikipedia results. Top-1 exact-match accuracy (in %, using SQuAD eval script). +Fine-tune (DS): Document Reader models trained on SQuAD and fine-tuned on each DS training set independently. +Multitask (DS): Document Reader single model trained on SQuAD and all the distant supervision (DS) training sets jointly. YodaQA results are extracted from https://github.com/brmson/yodaqa/wiki/Benchmarks and use additional resources such as Freebase and DBpedia, see Section 2.

Future work should aim to improve over our DrQA system. Two obvious angles of attack are: (i) incorporate the fact that Document Reader aggregates over multiple paragraphs and documents directly in the training, as it currently trains on paragraphs independently; and (ii) perform end-to-end training across the Document Retriever and Document Reader pipeline, rather than independent systems.

Acknowledgments

The authors thank Pranav Rajpurkar for testing Document Reader on the test set of SQuAD.

References

David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Mller, Maarten de Rijke, and Stefan Schlobach. 2004. Using wikipedia at the trec qa track. In *Proceedings of TREC 2004*.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, Springer, pages 722–735.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Petr Baudiš. 2015. YodaQA: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*. pages 1156–1165.

Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the YodaQA system. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, pages 222–228.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1533–1544.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075* .

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the AskMSR question-answering system. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 257–264.

Davide Buscaldi and Paolo Rosso. 2006. Mining knowledge from Wikipedia for the question answering task. In *International Conference on Language Resources and Evaluation (LREC)*. pages 727–730.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer, pages 95–133.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *International Conference on Machine Learning (ICML)*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *ACM SIGKDD international conference on Knowledge discovery and data mining*. pages 1156–1165.

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

Clinton Gormley and Zachary Tong. 2015. *Elasticsearch: The Definitive Guide*. " O'Reilly Media, Inc.".

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* .

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A novel large-scale language understanding task over wikipedia. In *Association for Computational Linguistics (ACL)*. pages 1535–1545.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The Goldilocks Principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations (ICLR)*.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. 2016. What makes ImageNet good for transfer learning? *arXiv preprint arXiv:1608.08614* .

Mohit Iyyer, Jordan L Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 633–644.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2016. From particular to general: A preliminary case study of transfer learning in reading comprehension. *Machine Intelligence Workshop, NIPS* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Association for Computational Linguistics (ACL)*. pages 55–60.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1400–1409.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL/IJCNLP)*. pages 1003–1011.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Pum-Mo Ryu, Myung-Gil Jang, and Hyun-Ki Kim. 2014. Open domain question answering using Wikipedia-based knowledge model. *Information Processing & Management* 50(5):683–692.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, pages 1045–1055.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *International Conference on Machine Learning (ICML)*. pages 1113–1120.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *International Conference on Learning Representations (ICLR)*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Learning to Skim Text

Adams Wei Yu*
Carnegie Mellon University
weiyu@cs.cmu.edu

Hongrae Lee
Google
hrlee@google.com

Quoc V. Le
Google
qvl@google.com

Abstract

Recurrent Neural Networks are showing much promise in many sub-areas of natural language processing, ranging from document classification to machine translation to automatic question answering. Despite their promise, many recurrent models have to read the whole text word by word, making it slow to handle long documents. For example, it is difficult to use a recurrent network to read a book and answer questions about it. In this paper, we present an approach of reading text while skipping irrelevant information if needed. The underlying model is a recurrent network that learns how far to jump after reading a few words of the input text. We employ a standard policy gradient method to train the model to make discrete jumping decisions. In our benchmarks on four different tasks, including number prediction, sentiment analysis, news article classification and automatic Q&A, our proposed model, a modified LSTM with jumping, is up to 6 times faster than the standard sequential LSTM, while maintaining the same or even better accuracy.

1 Introduction

The last few years have seen much success of applying neural networks to many important applications in natural language processing, e.g., part-of-speech tagging, chunking, named entity recognition (Collobert et al., 2011), sentiment analysis (Socher et al., 2011, 2013), document classification (Kim, 2014; Le and Mikolov, 2014; Zhang et al., 2015; Dai and Le, 2015), machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2014; Sennrich et al., 2015; Wu et al., 2016), conversational/dialogue modeling (Sordoni et al., 2015; Vinyals and Le, 2015; Shang et al., 2015), document summarization (Rush et al., 2015; Nallapati et al., 2016), parsing (Andor et al., 2016) and automatic question answering (Q&A) (Weston et al., 2015; Hermann et al., 2015; Wang and Jiang, 2016; Wang et al., 2016; Trischler et al., 2016; Lee et al., 2016; Seo et al., 2016; Xiong et al., 2016). An important characteristic of all these models is that they read all the text available to them. While it is essential for certain applications, such as machine translation, this characteristic also makes it slow to apply these models to scenarios that have long input text, such as document classification or automatic Q&A. However, the fact that texts are usually written with redundancy inspires us to think about the possibility of reading selectively.

In this paper, we consider the problem of understanding documents with partial reading, and propose a modification to the basic neural architectures that allows them to read input text with skipping. The main benefit of this approach is faster inference because it skips irrelevant information. An unexpected benefit of this approach is that it also helps the models generalize better.

In our approach, the model is a recurrent network, which learns to predict the number of jumping steps after it reads one or several input tokens. Such a discrete model is therefore not fully differentiable, but it can be trained by a standard policy gradient algorithm, where the reward can be the accuracy or its proxy during training.

In our experiments, we use the basic LSTM recurrent networks (Hochreiter and Schmidhuber, 1997) as the base model and benchmark the proposed algorithm on a range of document classification or reading comprehension tasks, using various datasets such as Rotten Tomatoes (Pang

*Most of work was done when AWY was with Google.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1880–1890
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1172

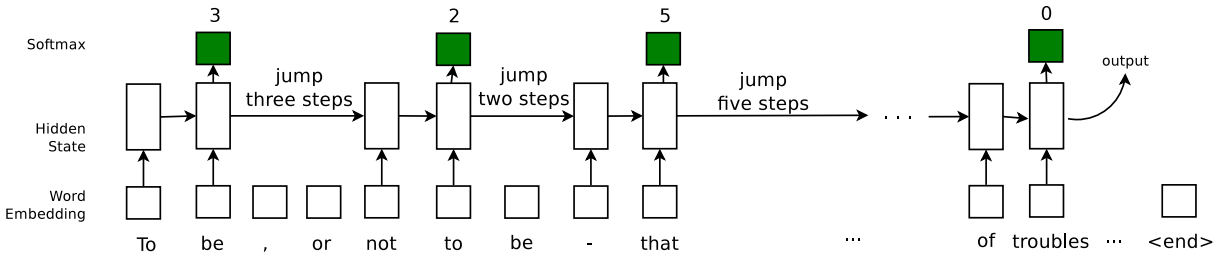


Figure 1: A synthetic example of the proposed model to process a text document. In this example, the maximum size of jump K is 5, the number of tokens read before a jump R is 2 and the number of jumps allowed N is 10. The green softmax are for jumping predictions. The processing stops if a) the jumping softmax predicts a 0 or b) the jump times exceeds N or c) the network processed the last token. We only show the case a) in this figure.

and Lee, 2005), IMDB (Maas et al., 2011), AG News (Zhang et al., 2015) and Children's Book Test (Hill et al., 2015). We find that the proposed approach of selective reading speeds up the base model by two to six times. Surprisingly, we also observe our model beats the standard LSTM in terms of accuracy.

In summary, the main contribution of our work is to design an architecture that learns to skim text and show that it is both faster and more accurate in practical applications of text processing. Our model is simple and flexible enough that we anticipate it would be able to incorporate to recurrent nets with more sophisticated structures to achieve even better performance in the future.

2 Methodology

In this section, we introduce the proposed model named LSTM-Jump. We first describe its main structure, followed by the difficulty of estimating part of the model parameters because of non-differentiability. To address this issue, we appeal to a reinforcement learning formulation and adopt a policy gradient method.

2.1 Model Overview

The main architecture of the proposed model is shown in Figure 1, which is based on an LSTM recurrent neural network. Before training, the number of jumps allowed N, the number of tokens read between every two jumps R and the maximum size of jumping K are chosen ahead of time. While K is a fixed parameter of the model, N and R are hyperparameters that can vary between training and testing. Also, throughout the paper, we would use $d_{1:p}$ to denote a sequence $d_1, d_2, ..., d_p$.

In the following, we describe in detail how the model operates when processing text. Given a training example $x_{1:T}$, the recurrent network will read the embedding of the first R tokens $x_{1:R}$ and output the hidden state. Then this state is used to compute the jumping softmax that determines a distribution over the jumping steps between 1 and K. The model then samples from this distribution a jumping step, which is used to decide the next token to be read into the model. Let κ be the sampled value, then the next starting token is $x_{R+\kappa}$. Such process continues until either

a) the jump softmax samples a 0; or
b) the number of jumps exceeds N; or
c) the model reaches the last token x_T.

After stopping, as the output, the latest hidden state is further used for predicting desired targets. How to leverage the hidden state depends on the specifics of the task at hand. For example, for classification problems in Section 3.1, 3.2 and 3.3, it is directly applied to produce a softmax for classification, while in automatic Q&A problem of Section 3.4, it is used to compute the correlation with the candidate answers in order to select the best one. Figure 1 gives an example with $K = 5$, $R = 2$ and $N = 10$ terminating on condition a).

2.2 Training with REINFORCE

Our goal for training is to estimate the parameters of LSTM and possibly word embedding, which are denoted as θ_m, together with the jumping action parameters θ_a. Once obtained, they can be used for inference.

The estimation of θ_m is straightforward in the tasks that can be reduced as classification problems (which is essentially what our experiments cover), as the cross entropy objective $J_1(\theta_m)$ is

differentiable over θ_m that we can directly apply backpropagation to minimize.

However, the nature of discrete jumping decisions made at every step makes it difficult to estimate θ_a, as cross entropy is no longer differentiable over θ_a. Therefore, we formulate it as a reinforcement learning problem and apply policy gradient method to train the model. Specifically, we need to maximize a reward function over θ_a which can be constructed as follows.

Let $j_{1:N}$ be the jumping action sequence during the training with an example $x_{1:T}$. Suppose h_i is a hidden state of the LSTM right before the i-th jump j_i,[1] then it is a function of $j_{1:i-1}$ and thus can be denoted as $h_i(j_{1:i-1})$. Now the jump is attained by sampling from the multinomial distribution $p(j_i|h_i(j_{1:i-1}); \theta_a)$, which is determined by the jump softmax. We can receive a reward R after processing $x_{1:T}$ under the current jumping strategy.[2] The reward should be positive if the output is favorable or non-positive otherwise. In our experiments, we choose

$$R = \begin{cases} 1 & \text{if prediction correct;} \\ -1 & \text{otherwise.} \end{cases}$$

Then the objective function of θ_a we want to maximize is the expected reward under the distribution defined by the current jumping policy, i.e.,

$$J_2(\theta_a) = \mathbb{E}_{p(j_{1:N};\theta_a)}[R]. \quad (1)$$

where $p(j_{1:N}; \theta_a) = \prod_i p(j_{1:i}|h_i(j_{1:i-1}); \theta_a)$.

Optimizing this objective numerically requires computing its gradient, whose exact value is intractable to obtain as the expectation is over high dimensional interaction sequences. By running S examples, an approximated gradient can be computed by the following REINFORCE algorithm (Williams, 1992):

$$\begin{aligned} \nabla_{\theta_a} J_2(\theta_a) &= \sum_{i=1}^{N} \mathbb{E}_{p(j_{1:N};\theta_a)}[\nabla_{\theta_a} \log p(j_{1:i}|h_i; \theta_a) R] \\ &\approx \frac{1}{S} \sum_{s=1}^{S} \sum_{i=1}^{N} [\nabla_{\theta_a} \log p(j^s_{1:i}|h^s_i; \theta_a) R^s] \end{aligned}$$

where the superscript s denotes a quantity belonging to the s-th example. Now the term $\nabla_{\theta_a} \log p(j_{1:i}|h_i; \theta_a)$ can be computed by standard backpropagation.

Although the above estimation of $\nabla_{\theta_a} J_2(\theta_a)$ is unbiased, it may have very high variance. One widely used remedy to reduce the variance is to subtract a *baseline* value b^s_i from the reward R^s, such that the approximated gradient becomes

$$\nabla_{\theta_a} J_2(\theta_a) \approx \frac{1}{S} \sum_{s=1}^{S} \sum_{i=1}^{N} [\nabla_{\theta_a} \log p(j^s_{1:i}|h^s_i; \theta)(R^s - b^s_i)]$$

It is shown (Williams, 1992; Zaremba and Sutskever, 2015) that any number b^s_i will yield an unbiased estimation. Here, we adopt the strategy of Mnih et al. (2014) that $b^s_i = w_b h^s_i + c_b$ and the parameter $\theta_b = \{w_b, c_b\}$ is learned by minimizing $(R^s - b^s_i)^2$. Now the final objective to minimize is

$$J(\theta_m, \theta_a, \theta_b) = J_1(\theta_m) - J_2(\theta_a) + \sum_{s=1}^{S} \sum_{i=1}^{N} (R^s - b^s_i)^2,$$

which is fully differentiable and can be solved by standard backpropagation.

2.3 Inference

During inference, we can either use sampling or greedy evaluation by selecting the most probable jumping step suggested by the jump softmax and follow that path. In the our experiments, we will adopt the sampling scheme.

3 Experimental Results

In this section, we present our empirical studies to understand the efficiency of the proposed model in reading text. The tasks under experimentation are: synthetic number prediction, sentiment analysis, news topic classification and automatic question answering. Those, except the first one, are representative tasks in text reading involving different sizes of datasets and various levels of text processing, from character to word and to sentence. Table 1 summarizes the statistics of the dataset in our experiments.

To exclude the potential impact of advanced models, we restrict our comparison between the vanilla LSTM (Hochreiter and Schmidhuber, 1997) and our model, which is referred to as LSTM-Jump. In a nutshell, we show that, while achieving the same or even better testing accuracy, our model is up to 6 times and 66 times faster than the baseline LSTM model in real and synthetic

[1] The i-th jumping step is usually *not* x_i.

[2] In the general case, one may receive (discounted) intermediate rewards after each jump. But in our case, we only consider final reward. It is equivalent to a special case that all intermediate rewards are identical and without discount.

Task	Dataset	Level	Vocab	AvgLen	#train	#valid	#test	#class
Number Prediction	synthetic	word	100	100 words	1M	10K	10K	100
Sentiment Analysis	Rotten Tomatoes	word	18,764	22 words	8,835	1,079	1,030	2
Sentiment Analysis	IMDB	word	112,540	241 words	21,143	3,857	25,000	2
News Classification	AG	character	70	200 characters	101,851	18,149	7,600	4
Q/A	Children Book Test-NE	sentence	53,063	20 sentences	108,719	2,000	2,500	10
Q/A	Children Book Test-CN	sentence	53,185	20 sentences	120,769	2,000	2,500	10

Table 1: Task and dataset statistics.

datasets, respectively, as we are able to selectively skip a large fraction of text.

In fact, the proposed model can be readily extended to other recurrent neural networks with sophisticated mechanisms such as attention and/or hierarchical structure to achieve higher accuracy than those presented below. However, this is orthogonal to the main focus of this work and would be left as an interesting future work.

General Experiment Settings We use the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 in all experiments. We also apply gradient clipping to all the trainable variables with the threshold of 1.0. The dropout rate between the LSTM layers is 0.2 and the embedding dropout rate is 0.1. We repeat the notations N, K, R defined previously in Table 2, so readers can easily refer to when looking at Tables 4,5,6 and 7. While K is fixed during both training and testing, we would fix R and N at training but vary their values during test to see the impact of parameter changes. Note that N is essentially a constraint which can be relaxed. Yet we prefer to enforce this constraint here to let the model learn to read fewer tokens. Finally, the reported test time is measured by running one pass of the whole test set instance by instance, and the speedup is over the base LSTM model. The code is written with TensorFlow.[3]

Notation	Meaning
N	number of jumps allowed
K	maximum size of jumping
R	number of tokens read before a jump

Table 2: Notations referred to in experiments.

3.1 Number Prediction with a Synthetic Dataset

We first test whether LSTM-Jump is indeed able to learn how to jump if a *very clear* jumping signal is given in the text. The input of the task is a sequence of L positive integers $x_{0:T-1}$ and the output is simply x_{x_0}. That is, the output is chosen from the input sequence, with index determined by x_0 . Here are two examples to illustrate the idea:

$$\text{input1} : \underline{4}, 5, 1, 7, \underline{6}, 2. \quad \text{output1} : 6$$

$$\text{input2} : \underline{2}, 4, \underline{9}, 4, 5, 6. \quad \text{output2} : 9$$

One can see that x_0 is essentially the oracle jumping signal, i.e. the indicator of how many steps the reading should jump to get the exact output and obviously, the remaining number of the sequence are useless. After reading the first token, a "smart" network should be able to learn from the training examples to jump to the output position, skipping the rest.

We generate 1 million training and 10,000 validation examples with the rule above, each with sequence length $T = 100$. We also impose $1 \leq x_0 < T$ to ensure the index is valid. We find that directly training the LSTM-Jump with full sequence is unlikely to converge, therefore, we adopt a curriculum training scheme. More specifically, we generate sequences with lengths $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ and train the model starting from the shortest. Whenever the training accuracy reaches a threshold, we shift to longer sequences. We also train an LSTM with the same curriculum training scheme. The training stops when the validation accuracy is larger than 98%. We choose such stopping criterion simply because it is the highest that both models can achieve.[4] All the networks are single layered, with hidden size 512, embedding size 32 and batch size 100. During testing, we generate sequences of lengths 10, 100 and 1000 with the same rule, each having 10,000 examples. As the training size is large enough, we do not have to worry about overfitting so dropout is not applied. In fact, we find that the training, validation and testing accuracies are almost the same.

[3] https://www.tensorflow.org/

[4] In fact, our model can get higher but we stick to 98% for ease of comparison.

Seq length	LSTM-Jump	LSTM	Speedup
Test accuracy			
10	**98%**	96%	n/a
100	**98%**	96%	n/a
1000	**90%**	80%	n/a
Test time (Avg tokens read)			
10	**13.5s (2.1)**	18.9s (10)	1.40x
100	**13.9s (2.2)**	120.4s (100)	8.66x
1000	**18.9s (3.0)**	1250s (1000)	**66.14x**

Table 3: Testing accuracy and time of synthetic number prediction problem. The jumping level is number.

The results of LSTM and our method, LSTM-Jump, are shown in Table 3. The first observation is that LSTM-Jump is faster than LSTM; the longer the sequence is, the more significant speedup LSTM-Jump can gain. This is because the well-trained LSTM-Jump is aware of the jumping signal at the first token and hence can directly jump to the output position to make prediction, while LSTM is agnostic to the signal and has to read the whole sequence. As a result, the reading speed of LSTM-Jump is hardly affected by the length of sequence, but that of LSTM is linear with respect to length. Besides, LSTM-Jump also outperforms LSTM in terms of test accuracy under all cases. This is not surprising either, as LSTM has to read a large amount of tokens that are potentially not helpful and could interfere with the prediction. In summary, the results indicate LSTM-Jump is able to learn to jump if the signal is clear.

3.2 Word Level Sentiment Analysis with Rotten Tomatoes and IMDB datasets

As LSTM-Jump has shown great speedups in the synthetic dataset, we would like to understand whether it could carry this benefit to real-world data, where "jumping" signal is not explicit. So in this section, we conduct sentiment analysis on two movie review datasets, both containing equal numbers of positive and negative reviews.

The first dataset is Rotten Tomatoes, which contains 10,662 documents. Since there is not a standard split, we randomly select around 80% for training, 10% for validation, and 10% for testing. The average and maximum lengths of the reviews are 22 and 56 words respectively, and we pad each of them to 60. We choose the pre-trained word2vec embeddings[5] (Mikolov et al., 2013) as our fixed word embedding that we do not update this matrix during training. Both LSTM-Jump and LSTM contain 2 layers, 256 hidden units and the batch size is 100. As the amount of training data is small, we slightly augment the data by sampling a continuous 50-word sequence in each padded reviews as one training sample. During training, we enforce LSTM-Jump to read 8 tokens before a jump ($R = 8$), and the maximum skipping tokens per jump is 10 ($K = 10$), while the number of jumps allowed is 3 ($N = 3$).

The testing result is reported in Table 4. In a nutshell, LSTM-Jump is always faster than LSTM under different combinations of R and N. At the same time, the accuracy is on par with that of LSTM. In particular, the combination of $(R, N) = (7, 4)$ even achieves slightly better accuracy than LSTM while having a 1.5x speedup.

Model	(R, N)	Accuracy	Time	Speedup
LSTM-Jump	(9, 2)	0.783	**6.3s**	**1.98x**
	(8, 3)	0.789	7.3s	1.71x
	(7, 4)	**0.793**	8.1s	1.54x
LSTM	n/a	0.791	12.5s	1x

Table 4: Testing time and accuracy on the Rotten Tomatoes review classification dataset. The maximum size of jumping K is set to 10 for all the settings. The jumping level is word.

The second dataset is IMDB (Maas et al., 2011),[6] which contains 25,000 training and 25,000 testing movie reviews, where the average length of text is 240 words, much longer than that of Rotten Tomatoes. We randomly set aside about 15% of training data as validation set. Both LSTM-Jump and LSTM has one layer and 128 hidden units, and the batch size is 50. Again, we use pretrained word2vec embeddings as initialization but they are updated during training. We either pad a short sequence to 400 words or randomly select a 400-word segment from a long sequence as a training example. During training, we set $R = 20, K = 40$ and $N = 5$.

As Table 5 shows, the result exhibits a similar trend as found in Rotten Tomatoes that LSTM-Jump is uniformly faster than LSTM under many settings. The various (R, N) combinations again demonstrate the trade-off between efficiency and accuracy. If one cares more about accuracy, then allowing LSTM-Jump to read and jump more

[5] https://code.google.com/archive/p/word2vec/

[6] http://ai.Stanford.edu/amaas/data/sentiment/index.html

Model	(R, N)	Accuracy	Time	Speedup
LSTM-Jump	(80, 8)	**0.894**	769s	1.62x
	(80, 3)	0.892	764s	1.63x
	(70, 3)	0.889	673s	1.85x
	(50, 2)	0.887	585s	2.12x
	(100, 1)	0.880	**489s**	**2.54x**
LSTM	n/a	0.891	1243s	1x

Table 5: Testing time and accuracy on the IMDB sentiment analysis dataset. The maximum size of jumping K is set to 40 for all the settings. The jumping level is word.

times is a good choice. Otherwise, shrinking either one would bring a significant speedup though at the price of losing some accuracy. Nevertheless, the configuration with the highest accuracy still enjoys a 1.6x speedup compared to LSTM. With a slight loss of accuracy, LSTM-Jump can be 2.5x faster .

3.3 Character Level News Article Classification with AG dataset

We now present results on testing the character level jumping with a news article classification problem. The dataset contains four classes of topics (World, Sports, Business, Sci/Tech) from the AG's news corpus,[7] a collection of more than 1 million news articles. The data we use is the subset constructed by Zhang et al. (2015) for classification with character-level convolutional networks. There are 30,000 training and 1,900 testing examples for each class respectively, where 15% of training data is set aside as validation. The non-space alphabet under use are:

```
abcdefghijklmnopqrstuvwxyz0123456
789-,;.!?:/\|_@#$%&*~`'+-=<>()[]{}
```

Since the vocabulary size is small, we choose 16 as the embedding size. The initialized entries of the embedding matrix are drawn from a uniform distribution in $[-0.25, 0.25]$, which are progressively updated during training. Both LSTM-Jump and LSTM have 1 layer and 64 hidden units and the batch sizes are 20 and 100 respectively. The training sequence is again of length 400 that it is either padded from a short sequence or sampled from a long one. During training, we set $R = 30, K = 40$ and $N = 5$.

The result is summarized in Table 6. It is interesting to see that even with skipping, LSTM-Jump is not always faster than LSTM. This is mainly due to the fact that the embedding size and hidden layer are both much smaller than those used previously, and accordingly the processing of a token is much faster. In that case, other computation overhead such as calculating and sampling from the jump softmax might become a dominating factor of efficiency. By this cross-task comparison, we can see that the larger the hidden unit size of recurrent neural network and the embedding are, the more speedup LSTM-Jump can gain, which is also confirmed by the task below.

Model	(R, N)	Accuracy	Time	Speedup
LSTM-Jump	(50, 5)	0.854	102s	0.80x
	(40, 6)	0.874	98.1s	0.83x
	(40, 5)	0.889	83.0s	0.98x
	(30, 5)	0.885	**63.6s**	**1.28x**
	(30, 6)	**0.893**	74.2s	1.10x
LSTM	n/a	0.881	81.7s	1x

Table 6: Testing time and accuracy on the AG news classification dataset. The maximum size of jumping K is set to 40 for all the settings. The jumping level is character.

3.4 Sentence Level Automatic Question Answering with Children's Book Test dataset

The last task is automatic question answering, in which we aim to test the sentence level skimming of LSTM-Jump. We benchmark on the data set Children's Book Test (CBT) (Hill et al., 2015).[8] In each document, there are 20 contiguous sentences (context) extracted from a children's book followed by a query sentence. A word of the query is deleted and the task is to select the best fit for this position from 10 candidates. Originally, there are four types of tasks according to the part of speech of the missing word, from which, we choose the most difficult two, i.e., the name entity (NE) and common noun (CN) as our focus, since simple language models can already achieve human-level performance for the other two types .

The models, LSTM or LSTM-Jump, firstly read the whole query, then the context sentences and finally output the predicted word. While LSTM reads everything, our jumping model would decide how many context sentences should skip after reading one sentence. Whenever a model finishes reading, the context and query are encoded in its

[7] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

[8] http://www.thespermwhale.com/jaseweston/babi/CBTest.tgz

hidden state h_{O}, and the best answer from the candidate words has the same index that maximizes the following:

$$\mathrm{softmax}(CWh_{\mathrm{O}}) \in \mathbb{R}^{10},$$

where $C \in \mathbb{R}^{10\times d}$ is the word embedding matrix of the 10 candidates and $W \in \mathbb{R}^{d\times \text{hidden\_size}}$ is a trainable weight variable. Using such bilinear form to select answer basically follows the idea of Chen et al. (2016), as it is shown to have good performance. The task is now distilled to a classification problem of 10 classes.

We either truncate or pad each context sentence, such that they all have length 20. The same preprocessing is applied to the query sentences except that the length is set as 30. For both models, the number of layers is 2, the number of hidden units is 256 and the batch size is 32. Pretrained word2vec embeddings are again used and they are not adjusted during training. The maximum number of context sentences LSTM-Jump can skip per time is $K = 5$ while the number of total jumping is limited to $N = 5$. We let the model jump after reading every sentence, so $R = 1$ (20 words).

The result is reported in Table 7. The performance of LSTM-Jump is superior to LSTM in terms of both accuracy and efficiency under all settings in our experiments. In particular, the fastest LSTM-Jump configuration achieves a remarkable 6x speedup over LSTM, while also having respectively 1.4% and 4.4% higher accuracy in Children's Book Test - Named Entity and Children's Book Test - Common Noun.

Model	(R, N)	Accuracy	Time	Speedup
Children's Book Test - Named Entity				
LSTM-Jump	(1, 5)	**0.468**	40.9s	3.04x
	(1, 3)	0.464	30.3s	4.11x
	(1, 1)	0.452	**19.9s**	**6.26x**
LSTM	n/a	0.438	124.5s	1x
Children's Book Test - Common Noun				
LSTM-Jump	(1, 5)	0.493	39.3s	3.09x
	(1, 3)	0.487	29.7s	4.09x
	(1, 1)	**0.497**	**19.8s**	**6.14x**
LSTM	n/a	0.453	121.5s	1x

Table 7: Testing time and accuracy on the Children's Book Test dataset. The maximum size of jumping K is set to 5 for all the settings. The jumping level is sentence.

The dominant performance of LSTM-Jump over LSTM might be interpreted as follows. After reading the query, both LSTM and LSTM-Jump know what the question is. However, LSTM still has to process the remaining 20 sentences and thus at the very end of the last sentence, the long dependency between the question and output might become weak that the prediction is hampered. On the contrary, the question can guide LSTM-Jump on how to read selectively and stop early when the answer is clear. Therefore, when it comes to the output stage, the "memory" is both fresh and uncluttered that a more accurate answer is likely to be picked.

In the following, we show two examples of how the model reads the context given a query (bold face sentences are those read by our model in the increasing order). XXXXX is the missing word we want to fill. Note that due to truncation, a few sentences might look uncompleted.

Example 1 In the first example, the exact answer appears in the context multiple times, which makes the task relatively easy, as long as the reader has captured their occurrences.

(a) *Query:* 'XXXXX!

(b) *Context:*

1. **said Big Klaus, and he ran off at once to Little Klaus.**
2. 'Where did you get so much money from?'
3. **'Oh, that was from my horse-skin.**
4. I sold it yesterday evening.'
5. 'That 's certainly a good price!'
6. said Big Klaus; and running home in great haste, he took an axe, knocked all his four
7. **'Skins!**
8. **skins!**
9. Who will buy skins?'
10. he cried through the streets.
11. All the shoemakers and tanners came running to ask him what he wanted for them.'
12. **A bushel of money for each,' said Big Klaus.**
13. 'Are you mad?'
14. **they all exclaimed.**
15. 'Do you think we have money by the bushel?'
16. 'Skins!
17. skins!
18. Who will buy skins?'
19. he cried again, and to all who asked him what they cost, he answered,' A bushel
20. 'He is making game of us,' they said; and the shoemakers seized their yard measures and

(c) *Candidates:* Klaus | Skins | game | haste | head | home | horses | money | price| streets

(d) *Answer:* Skins

The reading behavior might be interpreted as follows. The model tries to search for clues, and after reading sentence 8, it realizes that the most plausible answer is "Klaus" or "Skins", as they both appear twice. "Skins" is more likely to be the answer as it is followed by a "!". The model searches further to see if "Klaus!" is mentioned somewhere, but it only finds "Klaus" without "!" for the third time. After the last attempt at sentence 14, it is confident about the answer and stops to output with "Skins".

Example 2 In this example, the answer is illustrated by a word "nuisance" that does not show up in the context at all. Hence, to answer the query, the model has to understand the meaning of both the query and context and locate the synonym of "nuisance", which is not merely verbatim and thus much harder than the previous example. Nevertheless, our model is still able to make a right choice while reading much fewer sentences.

(a) *Query:* Yes, I call XXXXX a nuisance.

(b) *Context:*

1. **But to you and me it would have looked just as it did to Cousin Myra – a very discontented**
2. "I'm awfully glad to see you, Cousin Myra, "explained Frank carefully, "and your
3. **But Christmas is just a bore – a regular bore."**
4. **That was what Uncle Edgar called things that didn't interest him, so that Frank felt pretty sure of**
5. **Nevertheless, he wondered uncomfortably what made Cousin Myra smile so queerly.**
6. **"Why, how dreadful!"**
7. she said brightly.
8. "I thought all boys and girls looked upon Christmas as the very best time in the year."
9. "We don't, "said Frank gloomily.
10. **"It's just the same old thing year in and year out.**
11. We know just exactly what is going to happen.
12. We even know pretty well what presents we are going to get.
13. And Christmas Day itself is always the same.
14. We'll get up in the morning , and our stockings will be full of things, and half of
15. Then there 's dinner.
16. It 's always so poky.
17. And all the uncles and aunts come to dinner – just the same old crowd, every year, and
18. Aunt Desda always says, 'Why, Frankie, how you have grown!'
19. She knows I hate to be called Frankie.
20. And after dinner they'll sit round and talk the rest of the day, and that's all.

(c) *Candidates:* Christmas | boys | day | dinner | half | interest | rest | stockings | things | uncles

(d) *Answer:* Christmas

The reading behavior can be interpreted as follows. After reading the query, our model realizes that the answer should be something like a nuisance. Then it starts to process the text. Once it hits sentence 3, it may begin to consider "Christmas" as the answer, since "bore" is a synonym of "nuisance". Yet the model is not 100% sure, so it continues to read, very conservatively – it does not jump for the next three sentences. After that, the model gains more confidence on the answer "Christmas" and it makes a large jump to see if there is something that can turn over the current hypothesis. It turns out that the last-read sentence is still talking about Christmas with a negative voice. Therefore, the model stops to take "Christmas" as the output.

4 Related Work

Closely related to our work is the idea of learning visual attention with neural networks (Mnih et al., 2014; Ba et al., 2014; Sermanet et al., 2014), where a recurrent model is used to combine visual evidence at multiple fixations processed by a convolutional neural network. Similar to our approach, the model is trained end-to-end using the REINFORCE algorithm (Williams, 1992). However, a major difference between those work and ours is that we have to sample from discrete jumping distribution, while they can sample from continuous distribution such as Gaussian. The difference is mainly due to the inborn characteristics of text and image. In fact, as pointed out by Mnih et al. (2014), it was difficult to learn policies over more than 25 possible discrete locations.

This idea has recently been explored in the context of natural language processing applications, where the main goal is to filter irrelevant content using a small network (Choi et al., 2016). Perhaps the most closely related to our work is the concurrent work on learning to reason with reinforcement

learning (Shen et al., 2016). The key difference between our work and Shen et al. (2016) is that they focus on early stopping after multiple pass of data to ensure accuracy whereas our method focuses on selective reading with single pass to enable fast processing.

The concept of "hard" attention has also been used successfully in the context of making neural network predictions more interpretable (Lei et al., 2016). The key difference between our work and Lei et al. (2016)'s method is that our method optimizes for faster inference, and is more dynamic in its jumping. Likewise is the difference between our approach and the "soft" attention approach by (Bahdanau et al., 2014).

Our method belongs to adaptive computation of neural networks, whose idea is recently explored by (Graves, 2016; Jernite et al., 2016), where different amount of computations are allocated dynamically per time step. The main difference between our method and Graves; Jernite et al.'s methods is that our method can set the amount of computation to be exactly zero for many steps, thereby achieving faster scanning over texts. Even though our method requires policy gradient methods to train, which is a disadvantage compared to (Graves, 2016; Jernite et al., 2016), we do not find training with policy gradient methods problematic in our experiments.

At the high-level, our model can be viewed as a simplified trainable Turing machine, where the controller can move on the input tape. It is therefore related to the prior work on Neural Turing Machines (Graves et al., 2014) and especially its RL version (Zaremba and Sutskever, 2015). Compared to (Zaremba and Sutskever, 2015), the output tape in our method is more simple and reward signals in our problems are less sparse, which explains why our model is easy to train. It is worth noting that Zaremba and Sutskever report difficulty in using policy gradients to train their model.

Our method, by skipping irrelevant content, shortens the length of recurrent networks, thereby addressing the vanishing or exploding gradients in them (Hochreiter et al., 2001). The baseline method itself, Long Short Term Memory (Hochreiter and Schmidhuber, 1997), belongs to the same category of methods. In this category, there are several recent methods that try to achieve the same goal, such as having recurrent networks that operate in different frequency (Koutnik et al., 2014) or is organized in a hierarchical fashion (Chan et al., 2015; Chung et al., 2016).

Lastly, we should point out that we are among the recent efforts that deploy reinforcement learning to the field of natural language processing, some of which have achieved encouraging results in the realm of such as neural symbolic machine (Liang et al., 2017), machine reasoning (Shen et al., 2016) and sequence generation (Ranzato et al., 2015).

5 Conclusions

In this paper, we focus on learning how to skim text for fast reading. In particular, we propose a "jumping" model that after reading every few tokens, it decides how many tokens should be skipped by sampling from a softmax. Such jumping behavior is modeled as a discrete decision making process, which can be trained by reinforcement learning algorithm such as REINFORCE. In four different tasks with six datasets (one synthetic and five real), we test the efficiency of the proposed method on various levels of text jumping, from character to word and then to sentence. The results indicate our model is several times faster than, while the accuracy is on par with the baseline LSTM model.

Acknowledgments

The authors would like to thank the Google Brain Team, especially Zhifeng Chen and Yuan Yu for helpful discussion about the implementation of this model on Tensorflow. The first author also wants to thank Chen Liang, Hanxiao Liu, Yingtao Tian, Fish Tung, Chiyuan Zhang and Yu Zhang for their help during the project. Finally, the authors appreciate the invaluable feedback from anonymous reviewers.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042* .

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2014. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473* .

William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. 2015. Listen, attend and spell. *arXiv preprint arXiv:1508.01211* .

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Eunsol Choi, Daniel Hewlett, Alexandre Lacoste, Illia Polosukhin, Jakob Uszkoreit, and Jonathan Berant. 2016. Hierarchical question answering for long documents. *arXiv preprint arXiv:1611.01839* .

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704* .

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*. pages 3079–3087.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983* .

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* .

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv:1511.02301* .

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*, IEEE press.

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

Yacine Jernite, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Variable computation in recurrent neural networks. *arXiv preprint arXiv:1611.06188* .

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. 2014. A clockwork rnn. In *International Conference on Machine Learning*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning (ICML)*.

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155* .

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017: Long Papers*.

Andrew L Maas, Raymond E Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 142–150.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in neural information processing systems*. pages 2204–2212.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Conference on Computational Natural Language Learning (CoNLL)*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 115–124.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR* abs/1511.06732. http://arxiv.org/abs/1511.06732.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Pierre Sermanet, Andrea Frome, and Esteban Real. 2014. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054* .

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. *arXiv preprint arXiv:1609.05284* .

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. 2016. A parallel-hierarchical model for machine comprehension on sparse data. *arXiv preprint arXiv:1603.08884* .

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* .

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* .

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* .

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines-revised. *arXiv preprint arXiv:1505.00521* .

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. pages 649–657.

An Algebra for Feature Extraction

Vivek Srikumar
School of Computing
University of Utah
svivek@cs.utah.edu

Abstract

Though feature extraction is a necessary first step in statistical NLP, it is often seen as a mere preprocessing step. Yet, it can dominate computation time, both during training, and especially at deployment. In this paper, we formalize feature extraction from an algebraic perspective. Our formalization allows us to define a message passing algorithm that can restructure feature templates to be more computationally efficient. We show via experiments on text chunking and relation extraction that this restructuring does indeed speed up feature extraction in practice by reducing redundant computation.

1 Introduction

Often, the first step in building statistical NLP models involves feature extraction. It is well understood that the right choice of features can substantially improve classifier performance. However, from the computational point of view, the process of feature extraction is typically treated, at best as the preprocessing step of caching featurized inputs over entire datasets, and at worst, as 'somebody else's problem'. While such approaches work for training, when trained models are deployed, the computational cost of feature extraction cannot be ignored.

In this paper, we present the first (to our knowledge) algebraic characterization of the process of feature extraction. We formalize feature extractors as arbitrary functions that map objects (words, sentences, etc) to a vector space and show that this set forms a commutative semiring with respect to feature addition and feature conjunction.

An immediate consequence of the semiring characterization is a computational one. Every semiring admits the *Generalized Distributive Law (GDL) Algorithm* (Aji and McEliece, 2000) that exploits the distributive property to provide computational speedups. Perhaps the most common manifestation of this algorithm in NLP is in the form of inference algorithms for factor graphs and Bayesian networks like the max-product, max-sum and sum-product algorithms (*e.g.* Goodman, 1999; Kschischang et al., 2001). When applied to feature extractors, the GDL algorithm can refactor a feature extractor into a faster one by reducing redundant computation. In this paper, we propose a junction tree construction to allow such refactoring. Since the refactoring is done at the feature template level, the actual computational savings grow as classifiers encounter more examples.

We demonstrate the practical utility of our approach by factorizing existing feature sets for text chunking and relation extraction. We show that, by reducing the number of operations performed, we can obtain significant savings in the time taken to extract features.

To summarize, the main contribution of this paper is the recognition that feature extractors form a commutative semiring over addition and conjunction. We demonstrate a practical consequence of this characterization in the form of a mechanism for automatically refactoring any feature extractor into a faster one. Finally, we show the empirical usefulness of our approach on relation extraction and text chunking tasks.

2 Problem Definition

Before formal definitions, let us first see a running example.

2.1 Motivating Example

Consider the frequently used unigram, bigram and trigram features. Each of these is a template that specifies a feature representation for a word. In

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1891–1900
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1173

fact, the `bigram` and `trigram` templates themselves are compositional by definition. A `bigram` is simply the conjunction of a word `w` and previous word, which we will denote as $\texttt{w}_{-1}$; *i.e.*, $\texttt{bigram} = \texttt{w}_{-1}\&\texttt{w}$. Similarity, a `trigram` is the conjunction of $\texttt{w}_{-2}$ and `bigram`.

These templates are a function that operate on inputs. Given a sentence, say *John ate alone*, and a target word, say *alone*, they will produce indicators for the strings $\texttt{w}$=*alone*, $\texttt{w}_{-1}$=*ate*&$\texttt{w}$=*alone* and $\texttt{w}_{-2}$=*John*&$\texttt{w}_{-1}$=*ate*&$\texttt{w}$=*alone* respectively. Equivalently, each template maps an input to a vector. Here, the three vectors will be basis vectors associated with the feature strings.

Observe that the function that extracts the target word (*i.e.*, `w`) has to be executed in all three feature templates. Similarly, $\texttt{w}_{-1}$ has to be extracted to compute both the bigrams and the trigrams. *Can we optimize feature computation by automatically detecting such repetitions?*

2.2 Definitions and Preliminaries

Let X be a set of inputs to a classification problem at hand; *e.g.*, X could be words, sentences, etc. Let $\mathcal{V}$ be a possibly infinite dimensional vector space that represents the feature space. Feature extractors are functions that map the input space X to the feature space $\mathcal{V}$ to produce feature vectors for inputs. Let $\mathcal{F}$ represent the set of feature functions, defined as the set $\{\texttt{f} : X \to \mathcal{V}\}$. We will use the typewriter font to denote feature functions like `w` and `bigram`.

To round up the definitions, we will name two special feature extractors in $\mathcal{F}$. The feature extractor $\mathbb{0}$ maps all inputs to the zero vector. The feature extractor $\mathbb{1}$ maps all inputs to a bias feature vector. Without loss of generality, we will designate the basis vector $i_0 \in \mathcal{V}$ as the bias feature vector.

In this paper, we are concerned about two generally well understood operators on feature functions – addition and conjunction. However, let us see formal definitions for completeness.

Feature Addition. Given two feature extractors $\texttt{f}_1, \texttt{f}_2 \in \mathcal{F}$, feature addition (denoted by $+$) produces a feature extractor $\texttt{f}_1 + \texttt{f}_2$ that adds up the images of $\texttt{f}_1$ and $\texttt{f}_2$. That is, for any example $x \in X$, we have

$$(\texttt{f}_1 + \texttt{f}_2)(x) = \texttt{f}_1(x) + \texttt{f}_2(x) \qquad (1)$$

For example, the feature extractor $\texttt{w} + \texttt{w}_{-1}$ will map the word *alone* to a vector that is one for the basis elements $\texttt{w}$=*alone* and $\texttt{w}_{-1}$=*went*. This vector is the sum of the indicator vectors produced by the two operands `w` and $\texttt{w}_{-1}$.

Feature Conjunction. Given two feature extractors $\texttt{f}_1, \texttt{f}_2 \in \mathcal{F}$, their conjunction (denoted by $\&$) can be interpreted as an extension of Boolean conjunction. Indicator features like `bigram` are predicates for certain observations. Conjoining indicator features for two predicates is equivalent to an indicator feature for the Boolean conjunction of the predicates. More generally, with feature extractors that produce real valued vectors, the conjunction will produce their tensor product. The equivalence of feature conjunctions to tensor products has been explored and exploited in recent literature for various NLP tasks (Lei et al., 2014; Srikumar and Manning, 2014; Gormley et al., 2015; Lei et al., 2015).

We can further generalize this with an additional observation that is crucial for the rest of this paper. We argue that the conjunction operator produces *symmetric tensor products* rather than general tensor products. To see why, consider the `bigram` example. Though we defined the `bigram` feature as the conjunction of $\texttt{w}_{-1}$ and `w`, their ordering is irrelevant from classification perspective – the eventual goal is to associate weights with this combination of features. This observation allows us to formally define the conjunction operator as:

$$(\texttt{f}_1\&\texttt{f}_2)(x) = \text{vec}\left(\texttt{f}_1(x) \odot \texttt{f}_2(x)\right) \qquad (2)$$

Here, $\text{vec}(\cdot)$ stands for vectorize, which simply converts the resulting tensor into a vector and $\odot$ denotes the symmetric tensor product, introduced by Ryan (1980, Proposition 1.1). A symmetric tensor product is defined to be the average of the tensor products of all possible permutations of the operands, and thus, unlike a simple tensor product, is invariant to permutation of is operands. Informally, if we think of a tensor as a mapping from an ordered sequence of keys to real numbers, then, symmetric tensor product can be thought of as a mapping from a *set* of keys to numbers.

3 An Algebra for Feature Extraction

In this section, we will see that the set of feature extractors $\mathcal{F}$ form a commutative semiring with respect to addition and conjunction. First, let us revisit the definition of a commutative semiring.

Definition 1. *A* commutative semiring *is an algebraic structure consisting of a set K and two bi-*

nary operations $\oplus$ and $\otimes$ (addition and multiplication respectively) such that:

S1. $(K, \oplus)$ *is a commutative monoid:* $\oplus$ *is associative and commutative, and the set* K *contains a unique additive identity* 0 *such that* $\forall x \in K$*, we have* $0 \oplus x = x \oplus 0 = x$.

S2. $(K, \otimes)$ *is a commutative monoid:* $\otimes$ *is associative and commutative, and the set* K *contains a unique multiplicative identity* 1 *such that* $\forall x \in K$*, we have* $1 \otimes x = x \otimes 1 = x$.

S3. Multiplication distributes over addition on both sides. That is, for any $x, y, z \in K$*, we have* $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$ *and* $(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$.

S4. The additive identity is an annihilating element with respect to multiplication. That is, for any $x \in K$*, we have* $x \otimes 0 = 0 = 0 \otimes x$.

We refer the reader to Golan (2013) for a broad-ranging survey of semiring theory. We can now state and prove the main result of this paper.

Theorem 1. *Let* X *be any set and let* $\mathcal{F}$ *denote the set of feature extractors defined on the set. Then,* $(\mathcal{F}, +, \&)$ *is a commutative semiring.*

Proof. We will show that the properties of a commutative semiring hold for $(\mathcal{F}, +, \&)$ using the definitions of the operators from §2.2. Let $\mathtt{f}_1, \mathtt{f}_2$ and $\mathtt{f} \in \mathcal{F}$ be feature extractors.

S1. For any example $x \in X$, we have $(\mathtt{f}_1 + \mathtt{f}_2)(x) = \mathtt{f}_1(x) + \mathtt{f}_2(y)$. The right hand side denotes vector addition, which is associative and commutative. The $\mathbb{0}$ feature extractor is the additive identify because it produces the zero vector for any input. Thus, $(\mathcal{F}, +)$ is a commutative monoid.

S2. To show that the conjunction operator is associative over feature extractors, it suffices to observe that the tensor product (and hence the symmetric tensor product) is associative. Furthermore, the symmetric tensor product is commutative by definition, because it is invariant to permutation of its operands.
Finally, the bias feature extractor, $\mathbb{1}$, that maps all inputs to the bias vector i_0, is the multiplicative identity. To see this, consider the conjunction $\mathtt{f}\&\mathbb{1}$, applied to an input x:

$$\begin{aligned}(\mathtt{f}\&\mathbb{1})(x) &= \text{vec}(\mathtt{f}(x) \odot \mathbb{1}(x)) \\ &= \text{vec}(\mathtt{f}(x) \odot i_0)\end{aligned}$$

The product term within the $\text{vec}(\cdot)$ in the final expression is a symmetric tensor, defined by basis vectors that are sets of the form $\{i_0, i_0\}, \{i_1, i_0\}, \cdots$. Each basis $\{i_j, i_0\}$ is associated with a feature value $\mathtt{f}(x)_j$. Thus, the vectorized form of this tensor will contain the *same elements* as $\mathtt{f}(x)$, perhaps mapped to different bases. The mapping from $\mathtt{f}(x)$ to the final vector is independent of the input x because the bias feature extractor is independent of x. Without loss of generality, we can fix this mapping to be the identity mapping, thereby rendering the final vectorized form equal to $\mathtt{f}(x)$. That is, $\mathtt{f}\&\mathbb{1} = \mathtt{f}$.
Thus, $(\mathcal{F}, \&)$ is a commutative monoid.

S3. Since tensor products distribute over addition, we get the distributive property.

S4. By definition, conjoining with the $\mathbb{0}$ feature extractor annihilates all feature functions because $\mathbb{0}$ maps all inputs to the zero vector.

■

4 From Algebra to an Algorithm

The fact that feature extractors form a commutative semiring has a computational consequence. The *generalized distributive law (GDL) algorithm* (Aji and McEliece, 2000) exploits the properties of a commutative semiring to potentially reduce the computational effort for marginalizing sums of products. The GDL algorithm manifests itself as the Viterbi, Baum-Welch, Floyd-Warshall and belief propagation algorithms, and the Fast Fourier and Hadamard transforms. Each corresponds to a different commutative semiring and a specific associated marginalization problem.

Here, we briefly describe the general marginalization problem from Aji and McEliece (2000) to introduce notation and also highlight the analogies to inference in factor graphs. Let $x_1, x_2, \cdots, x_n$ denote a collection of variables that can take values from finite sets $A_1, A_2, \cdots, A_n$ respectively. Let boldface $\mathbf{x}$ denote the entire set of variables. These variables are akin to inference variables in factor graphs that may be assigned values or marginalized away.

Let $(K, \oplus, \otimes)$ denote a commutative semiring. Suppose α_i is a function that maps a subset of the variables $\{x_{i_1}, x_{i_2}, \cdots\}$ to the set K. The subset of variables that constitute the domain of α_i is called the *local domain* of the corresponding *local function*. Local domains and local functions are analogous to factors and factor potentials in a factor graph. With a collection of local domains, each associated with a function α_i, the "marginalize the

product" problem is that of computing:

$$\sum_{\mathbf{x}} \prod_{i} \alpha_i \left(x_{i_1}, x_{i_2}, \cdots\right) \tag{3}$$

Here, the sum and product use the semiring operators. The summation is over all possible valid assignments of the variables $\mathbf{x}$ over the cross product of the sets $A_1, A_2, \cdots, A_n$. This problem generalizes the familiar max-product or sum-product settings. Indeed, the GDL algorithm is a generalization of the message passing (Pearl, 2014) for efficiently computing marginals.

To make feature extraction efficient using the GDL algorithm, in the next section, we will define a marginalization problem in terms of the semiring operators by specifying the variables involved, the local domains and local functions. Instead of describing the algorithm in the general setting, we will instantiate it on the semiring at hand.

5 Marginalizing Feature Extractors

First, let us see why we can expect any benefit from the GDL algorithm by revisiting our running example (unigrams, bigrams and trigrams), written below using the semiring operations:

$$\mathtt{f} = \mathtt{w} + (\mathtt{w}_{-1} \& \mathtt{w}) + (\mathtt{w}_{-2} \& \mathtt{w}_{-1} \& \mathtt{w}) \tag{4}$$

When applied to a token, $\mathtt{f}$ performs two additions and three conjunctions. However, by applying the distributive property, we can refactor it as follows to reduce the number of operations:

$$\mathtt{f}' = (\mathbb{1} + (\mathbb{1} + \mathtt{w}_{-2}) \,\&\mathtt{w}_{-1}) \,\&\mathtt{w} \tag{5}$$

The refactored version $\mathtt{f}'$ – equivalent to the original one – only performs two additions and two conjunctions, offering a computational saving of one operation. This refactoring is done at the level of feature templates (*i.e.*, feature extractors); the actual savings are realized when the feature vectors are computed by applying this feature function to an input. Thus, the simplification, though seemingly modest at the template level, can lead to a substantial speed improvements when the features vectors are actually manifested from data.

The GDL algorithm instantiated with the feature extractor semiring, automates such factorization at a symbolic level. In the rest of this section, first (§5.1), we will write our problem as a marginalization problem, as in Equation (3). Then (§5.2), we will construct a junction tree to apply the message passing algorithm.

5.1 Canonicalizing Feature Extractors

To frame feature simplification as marginalization, we need to first write any feature extractor as a *canonical* sum of products that is amenable for factorization (*i.e.*, as in (3)). To do so, in this section, we will define: (a) the variables involved, (b) the local domains (*i.e.*, subsets of variables contributing to each product term), and, (c) a local function for each local domain (*i.e.*, the α_i's).

Variables. First, we write a feature extractor as a sum of products. Our running example (4) is already one. If we had an expression like $\mathtt{f}_1 \& (\mathtt{f}_2 + \mathtt{f}_3)$, we can expand it into $\mathtt{f}_1 \& \mathtt{f}_2 + \mathtt{f}_1 \& \mathtt{f}_3$. From the sum of products, we identify the *base feature extractors* (*i.e.*, ones not composed of other feature extractors) and define a variable x_i for each. In our example, we have $\mathtt{w}$, $\mathtt{w}_{-1}$ and $\mathtt{w}_{-2}$.

Next, recall from §4 that each variable x_i can take values from a finite set A_i. If a base feature extractor $\mathtt{f}_i$ corresponds to the variable x_i, then, we define x_i's domain to be the set $A_i = \{\mathbb{1}, \mathtt{f}_i\}$. That is, each variable can either be the bias feature extractor or the feature extractor associated with it. Our example gives three variables x_1, x_2, x_3 with domains $A_1 = \{\mathbb{1}, \mathtt{w}\}$, $A_2 = \{\mathbb{1}, \mathtt{w}_{-1}\}$, $A_3 = \{\mathbb{1}, \mathtt{w}_{-2}\}$ respectively.

Local domains. Local domains are subsets of the variables defined above. They are the domains of functions that constitute products in the canonical form of a feature extractor. We define the following local domains, each illustrated with the corresponding instantiation in our running example:

1. A singleton set for each variable: $\{x_1\}$, $\{x_2\}$, and $\{x_3\}$.
2. One local domain consisting of all the variables: The set $\{x_1, x_2, x_3\}$.
3. One local domain consisting of no variables: The empty set $\{\}$.
4. One local domain for each subset of base feature extractors that participate in at least two conjunctions in the sum-of-products (*i.e.*, the ones that can be factored away): Only $\{x_1, x_2\}$ in our example, because only $\mathtt{w}$ and $\mathtt{w}_{-1}$ participate in two conjunctions in (4).

Local functions. Each local domain is associated with a function that maps variable assignments to feature extractors. These functions (called local kernels by Aji and McEliece (2000)) are like potential functions in a factor graph. We define two kinds of local functions, driven by the goal of de-

signing a marginalization problem that pushes towards simpler feature functions.

1. We associate the *identity function* with all singleton local domains, and the constant function that returns the bias $\mathbb{1}$ with the empty domain $\{\}$.
2. With all other local domains, we associate an *indicator function*, denoted by z. For a local domain, z is an indicator for those assignments of the variables involved, whose conjunctions are present in any product term in sum-of-products. In our running example, the function $z(x_1, x_2)$ is the indicator for (x_1, x_2) belonging to the set $\{(\mathtt{w}, \mathbb{1}), (\mathtt{w}, \mathtt{w}_{-1})\}$, represented by the table:

x_1	x_2	$z(x_1, x_2)$
$\mathbb{1}$	$\mathbb{1}$	$\mathbb{0}$
$\mathbb{1}$	$\mathtt{w}_{-1}$	$\mathbb{0}$
$\mathtt{w}$	$\mathbb{1}$	$\mathbb{1}$
$\mathtt{w}$	$\mathtt{w}_{-1}$	$\mathbb{1}$

The indicator returns the semiring's multiplicative and additive identities. The value of z above for inputs $(\mathtt{w}, \mathbb{1})$ is $\mathbb{1}$ because the first term in (4) that defines the feature extractor contains $\mathtt{w}$, but not $\mathtt{w}_{-1}$. On the other hand, the input $(\mathbb{1}, \mathbb{1})$ is mapped to $\mathbb{0}$ because every product term contains either $\mathtt{w}$ or $\mathtt{w}_{-1}$. For the local domain $\{x_1, x_2, x_3\}$, the local function is the indicator for the set $\{(\mathtt{w}, \mathbb{1}, \mathbb{1}), (\mathtt{w}, \mathtt{w}_{-1}, \mathbb{1}), (\mathtt{w}, \mathtt{w}_{-1}, \mathtt{w}_{-2})\}$, corresponding to each product term.

In summary, for the running example we have:

Local domain	Local function
$\{x_1\}$	x_1
$\{x_2\}$	x_2
$\{x_3\}$	x_3
$\{x_1, x_2, x_3\}$	$z(x_1, x_2, x_3)$
$\{\}$	$\mathbb{1}$
$\{x_1, x_2\}$	$z(x_1, x_2)$

The procedure described here aims to convert *any* feature function into a canonical form that can be factorized using the GDL algorithm. Indeed, using local domains and functions specified above, any feature extractor can we written as a canonical sum of products as in (3). For example, using the table above, our running example is identical to

$$\sum_{x_1, x_2, x_3} z(x_1, x_2, x_3) \& z(x_1, x_2) \& x_1 \& x_2 \& x_3 \quad (6)$$

Here, the summation is over the cross product of the A_i's. The choice of the z functions ensures that only those conjunctions that were in the original feature extractor remain.

This section shows one approach for canonicalization; the local domains and functions are a design choice that may be optimized in future work. We should also point out that, while this process is notationally tedious, its actual computational cost is negligible, especially given that it is to be performed only once at the template level.

5.2 Simplifying feature extractors

As mentioned in §4, a commutative semiring can allow us to employ the GDL algorithm to efficiently compute a sum of products. Starting from a canonical sum-of-products expression such as the one in (6), this process is similar to variable elimination for Bayesian networks. The junction tree algorithm is a general scheme to avoid redundant computation in such networks (Cowell, 2006). To formalize this, we will first build a junction tree and then define the messages sent from the leaves to the root. The final message at the root will give us the simplified feature function.

Constructing a Junction Tree. First, we will construct a junction tree using the local domains from § 5.1. In any junction tree, the edges should satisfy the *running intersection property*: *i.e.*, if a variable x_i is in two nodes in the tree, then it should be in every node in the path connecting them. To build a junction tree, we will first create a graph whose nodes are the local domains. The edges of this graph connect pairs of nodes if the variables in one are a subset of the other. For simplicity, we will assume that our nodes are arranged in a lattice as shown in Figure 1, with edges connecting nodes in subsequent levels. For example, there is no edge connecting nodes `B` and `C`.

Every spanning tree of this lattice is a junction tree. Which one should we consider? Let us examine the properties that we need. First, the root of the tree should correspond to the empty local domain $\{\}$ because messages arriving at this node will accumulate all products. Second, as we will see, feature extractors farther from the root will appear in inner terms in the factorized form. That is, frequent or more expensive feature extractors should be incentivized to appear higher in the tree.

To capture these preferences, we frame the task of constructing the junction tree as a maximum spanning tree problem over the graph, with edge weights incorporating the preferences. One natural weighting function is the computational expense of the base feature extractors associated with that edge. For example, the weight associated with the edge connecting nodes `E` and `D` in the fig-

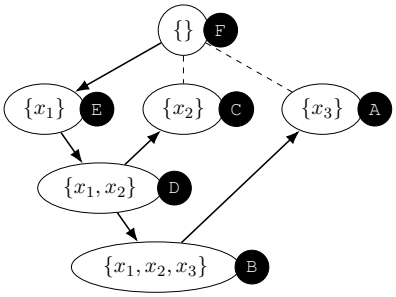


Figure 1: The junction tree for our running example. The process of constructing the junction tree is described in the text. Here, we show both the tree and the graph from which it is constructed; dashed lines show edges are not in the tree. Filled circles denote the names of the nodes. The local domain $\{x_1\}$ is connected to the empty local domain because the feature `w` corresponding to it is most frequent.

ure can be the average cost of the `w` and $\texttt{w}_{-1}$ feature extractors. If computational costs are unavailable, we can use the number times a feature extractor appears in the expression to be simplified. Under this criterion, in our example, edges connecting `E` to its neighbors will be weighted highest.

Once we have a spanning tree, we make the edges directed so that the empty set is the root. Figure 1 shows the junction tree obtained for our running example.

Message Passing for Feature Simplification. Given the junction tree, we can use a standard message passing scheme for factorization. The goal is to collect information at each node in the tree from its children all the way to the root.

Suppose v_i, v_j denote two nodes in the tree. Since nodes are associated with sets of variables, their intersections $v_i \cap v_j$ and differences $v_i \setminus v_j$ are defined. For example, in the example, $\texttt{A} \cap \texttt{B} = \{x_3\}$ and $\texttt{B} \setminus \texttt{D} = \{x_3\}$. We will denote children of a node v_i in the junction tree by $C(v_i)$.

The message from any node v_i to its parent v_j is a function that maps the variables $v_i \cap v_j$ to a feature extractor by marginalizing out all variables that are in v_i but not in v_j. Formally, we define the message μ_{ij} from a node v_i to a node v_j as:

$$\mu_{ij}\left(v_i \cap v_j\right) = \sum_{v_j \setminus v_i} \alpha_i\left(v_i\right) \prod_{v_k \in C(v_i)} \mu_{ki}\left(v_k \cap v_i\right). \quad (7)$$

Here, α_i is the local function at node v_i. To complete the formal definition of the algorithm, we note that by performing post-order traversal of the junction tree, we will accumulate all messages at the root of the tree, that corresponds to the empty set of variables. The incoming message at this node represents the factorized feature extractor.

Algorithm 1 briefly summarizes the entire simplification process. The proof of correctness of the algorithm follows from the fact that the range of all the local functions is a commutative semiring, namely the feature extractor semiring. We refer the reader to (Aji and McEliece, 2000, Appendix A) for details.

Algorithm 1 The Generalized Distributive Law Algorithm for simplifying a feature extractor `f`. See the text for details.

1: Convert `f` into a canonical sum of products representation (§ 5.1).
2: Construct a junction tree whose nodes are local domains.
3: **for** edge (v_j, v_i) in the post-order traversal of the tree **do**
4: Receive a message μ_{ij} at v_j using (7).
5: **end for**
6: **return** the incoming message at the root

Example run of message propagation. As an illustration, let us apply it to our running example.

1. The first message is from `A` to `B`. Since `A` has no children and its local function is the identity function, we have $\mu_{AB}(x) = x$. Similarly, we have $\mu_{CD}(x) = x$.
2. The message from `B` to `D` has to marginalize out the variable x_3. That is, we have $\mu_{BD}(x_1, x_2) = \sum_{x_3} z(x_1, x_2, x_3)\mu_{AB}(x_3)$. The summation is over the domain of x_3, namely $\{\mathbb{1}, \texttt{w}_{-2}\}$. By substituting for z and μ_{AB}, and simplifying, we get the message:

x_1	x_2	$\mu_{BD}(x_1, x_2)$
$\mathbb{1}$	$\mathbb{1}$	$\mathbb{0}$
$\mathbb{1}$	$\texttt{w}_{-1}$	$\mathbb{0}$
$\texttt{w}$	$\mathbb{1}$	$\mathbb{1}$
$\texttt{w}$	$\texttt{w}_{-1}$	$\mathbb{1} + \texttt{w}_{-2}$

3. The message from `D` to `E` marginalizes out the variable x_2 to give us $\mu_{DE}(x_1) = \sum_{x_2} z(x_1, x_2)\mu_{CD}(x_2)\mu_{BD}(x_1, x_2)$. Here, the summation is over the domain of x_2, namely $\{\mathbb{1}, \texttt{w}_{-1}\}$. We can simplify the message as:

x_1	$\mu_{DE}(x_1)$
$\mathbb{1}$	$\mathbb{0}$
$\texttt{w}$	$\mathbb{1} + (\mathbb{1} + \texttt{w}_{-2})\,\&\texttt{w}_{-1}$

4. Finally, the message from `E` to the root `F` marginalizes out the variable x_1 by summing over its domain $\{\mathbb{1}, \texttt{w}\}$ to give us the message $(\mathbb{1} + (\mathbb{1} + \texttt{w}_{-2})\,\&\texttt{w}_{-1})\,\&\texttt{w}$.

The message received at the root is the factorized feature extractor. Note that the final form is identical to (5) at the beginning of §5.

Discussion. An optimal refactoring algorithm would produce a feature extractor that is both correct and fastest. The algorithm above has the former guarantee. While it does reduce the number of operations performed, the closeness of the refac-

tored feature function to the fastest one depends on the heuristic used to weight edges for identifying the junction tree. Changing the heuristic can change the junction tree, thus changing the final factorized function. We found via experiments that using the number of times a feature extractor occurs in the sum-of-products to weight edges is promising. A formal study of optimality of factorization is an avenue of future research.

6 Experiments

We show the practical usefulness of feature function refactoring using text chunking and relation extraction. In both cases, the question we seek to evaluate empirically is: *Does the feature function refactoring algorithm improve feature extraction time?* We should point out that our goal is not to measure accuracy of prediction, but the efficiency of feature extraction. Indeed, we are guaranteed that refactoring will not change accuracy; factorized feature extractors produce the *same* feature vectors as the original ones.

In all experiments, we compare a feature extractor and its refactored variant. For the factorization, we incentivized the junction tree to factor out base feature extractors that occurred most frequently in the feature extractor. For both tasks, we use existing feature representations that we briefly describe. We refer the reader to the original work that developed the feature representations for further details. For both the original and the factorized feature extractors, we report (a) the number of additions and conjunctions at the template level, and, (b) the time for feature extraction on the entire dataset. For the time measurements, we report average times for the original and factorized feature extractors over five paired runs to average out variations in system load.[1]

6.1 Text Chunking

We use data from the CoNLL 2000 shared task (Tjong Kim Sang and Buchholz, 2000) of text chunking and the feature set described by Martins et al. (2011), consisting of the following templates extracted at each word: (1) Up to 3-grams of POS tags within a window of size ten centered at the word, (2) up to 3-grams of words, within a window of size six centered at the word, and (3) up to 2-grams of word shapes, within a window of size four centered at the word. In all, there are 96 feature templates.

[1]We performed all our experiments on a server with 128GB RAM and 24 CPU cores, each clocking at 2600 MHz.

Setting	Size		Average feature extraction time (ms)
	+	&	
Original	47	75	17776.6
Factorized	47	54	4294.2

Table 1: Comparison of the original and factorized feature extractors for the text chunking task. The time improvement is statistically significant using the paired t-test at $p < 0.01$.

Setting	Size		Average feature extraction time (ms)
	+	&	
Original	43	19	8173.0
Factorized	43	11	6276.4

Table 2: Comparison of the original and factorized feature extractors for the relation extraction task. We measured time using 3191 training mention pairs. The time improvement is statistically significant using the paired t-test at $p < 0.01$.

We factorized the feature representation using Algorithm 1. Table 1 reports the number of operations (addition and conjunction) in the templates in the original and factorized versions of the feature extractor. The table also reports feature extraction time taken from the entire training set of 8,936 sentences, corresponding to 211,727 tokens. First, we see that the factorization reduces the number of feature conjunction operations. Thus, to produce exactly the same feature vector, the factorized feature extractor does less work. The time results show that this computational gain is not merely a theoretical one; it also manifests itself practically.

6.2 Relation Extraction

Our second experiment is based on the task of relation extraction using the English section of the ACE 2005 corpus (Walker et al., 2006). The goal is to identify semantic relations between two entity mentions in text. We use the feature representation developed by Zhou et al. (2005) as part of an investigation of how various lexical, syntactic and semantic sources of information affect the relation extraction task. To this end, the feature set consists of word level information about mentions, their entity types, their relationships with chunks, path features from parse trees, and semantic features based on WordNet and various word lists. Given the complexity of the features, we do not describe them here and refer the reader to the original work for details. Note that compared to the chunking features, these features are more diverse in their computational costs.

We report the results of our experiments in Ta-

ble 2. As before, we see that the number of conjunction operations decreases after factorization. Curiously, however, despite the complexity of the feature set, the actual number of operations is smaller than text chunking. Due to this, we see a more modest, yet significant decrease in the time for feature extraction after factorization.

7 Related Work and Discussion

Simplifying Expressions. The problem of simplifying expressions with an eye on computational efficiency is the focus of logic synthesis (cf. Hachtel and Somenzi, 2006), albeit largely geared towards analyzing and verifying digital circuits. Logic synthesis is NP-hard in general. In our case, the hardness is hidden in the fact that our approach does not guarantee that we will find the smallest (or most efficient) factorization. The junction tree construction determines the factorization quality.

Semirings in NLP. Semirings abound in NLP, though primarily as devices to design efficient inference algorithms for various graphical models (*e.g.* Wainwright and Jordan, 2008; Sutton et al., 2012). Goodman (1999) synthesized various parsing algorithms in terms of semiring operations. Since then, we have seen several explorations of the interplay between weighted dynamic programs and semirings for inference in tasks such as parsing and machine translation (*e.g.* Eisner et al., 2005; Li and Eisner, 2009; Lopez, 2009; Gimpel and Smith, 2009). Allauzen et al. (2003) developed efficient algorithms for constructing statistical language models by exploiting the algebraic structure of the probability semiring.

Feature Extraction and Modeling Languages. Much work around features in NLP is aimed at improving classifier accuracy. There is some work on developing languages to better construct feature spaces (Cumby and Roth, 2002; Broda et al., 2013; Sammons et al., 2016), but they do not formalize feature extraction from an algebraic perspective. We expect that the algorithm proposed in this paper can be integrated into such feature construction languages, and also into libraries geared towards designing feature rich models (*e.g.* McCallum et al., 2009; Chang et al., 2015).

Representation vs. Speed. As the recent successes (Goodfellow et al., 2016) of distributed representations show, the representational capacity of a feature space is of primary importance. Indeed, several recent lines of work that use distributed representations have independently identified the connection between conjunctions (of features or factors in a factor graph) and tensor products (Lei et al., 2014; Srikumar and Manning, 2014; Gormley et al., 2015; Yu et al., 2015; Lei et al., 2015; Primadhanty et al., 2015). They typically impose sparsity or low-rank requirements to induce better representations for learning. In this paper, we use the connection between tensor products and conjunctions to prove algebraic properties of feature extractors, leading to speed improvements via factorization.

In this context, we note that in both our experiments, the number of conjunctions are reduced by factorization. We argue that this is an important saving because conjunctions can be a more expensive operation. This is especially true when dealing with dense feature representations, as is increasingly common with word vectors and neural networks, because conjunctions of dense feature vectors are tensor products, which can be slow.

Finally, while training classifiers can be time consuming, when trained classifiers are deployed, feature extraction will dominate computation time over the classifier's lifetime. However, the prediction step includes both feature extraction and computing inner products between features and weights. Many features may be associated with zero weights because of sparsity-inducing learning (*e.g.* Andrew and Gao, 2007; Martins et al., 2011; Strubell et al., 2015). Since these two aspects are orthogonal to each other, the factorization algorithm presented in this paper can be used to speed up extraction of those features that have non-zero weights.

8 Conclusion

In this paper, we studied the process of feature extraction using an algebraic lens. We showed that the set of feature extractors form a commutative semiring over addition and conjunction. We exploited this characterization to develop a factorization algorithm that simplifies feature extractors to be more computationally efficient. We demonstrated the practical value of the refactoring algorithm by speeding up feature extraction for text chunking and relation extraction tasks.

Acknowledgments

The author thanks the anonymous reviewers for their insightful comments and feedback.

References

Srinivas M Aji and Robert J McEliece. 2000. The generalized distributive law. *IEEE Transactions on Information Theory* 46(2).

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *ACL*.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L_1-regularized log-linear models. In *ICML*.

Bartosz Broda, Paweł Kędzia, Michał Marcińczuk, Adam Radziszewski, Radosław Ramocki, and Adam Wardyński. 2013. Fextor: A feature extraction framework for natural language processing: A case study in word sense disambiguation, relation recognition and anaphora resolution. In *Computational Linguistics*, Springer, pages 41–62.

Kai-Wei Chang, Shyam Upadhyay, Ming-Wei Chang, Vivek Srikumar, and Dan Roth. 2015. IllinoisSL: A JAVA library for Structured Prediction. *arXiv preprint arXiv:1509.07179* .

Robert G Cowell. 2006. *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business Media.

Chad M Cumby and Dan Roth. 2002. Learning with feature description logics. In *Inductive logic programming*, Springer.

Jason Eisner, Eric Goldlust, and Noah A Smith. 2005. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *HLT-EMNLP*.

Kevin Gimpel and Noah A Smith. 2009. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *EACL*.

Jonathan S Golan. 2013. *Semirings and their Applications*. Springer Science & Business Media.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics* 25(4):573–605.

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *EMNLP*.

Gary D Hachtel and Fabio Somenzi. 2006. *Logic synthesis and verification algorithms*. Springer Science & Business Media.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory* 47(2):498–519.

Tao Lei, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *ACL*.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order lowrank tensors for semantic role labeling. In *NAACL*.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *EMNLP*.

Adam Lopez. 2009. Translation as weighted deduction. In *EACL*.

André FT Martins, Noah A Smith, Pedro MQ Aguiar, and Mário AT Figueiredo. 2011. Structured sparsity in structured prediction. In *CoNLL*.

Andrew McCallum, Karl Schultz, and Sameer Singh. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In *NIPS*.

Judea Pearl. 2014. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

Audi Primadhanty, Xavier Carreras, and Ariadna Quattoni. 2015. Low-rank regularization for sparse conjunctive feature spaces: An application to named entity classification. In *ACL*.

Raymond A Ryan. 1980. *Applications of topological tensor products to infinite dimensional holomorphy*. Ph.D. thesis, Trinity College.

Mark Sammons, Christos Christodoulopoulos, Parisa Kordjamshidi, Daniel Khashabi, Vivek Srikumar, and Dan Roth. 2016. EDISON: Feature Extraction for NLP. In *LREC*.

Vivek Srikumar and Christopher D. Manning. 2014. Learning distributed representations for structured output prediction. In *NIPS*.

Emma Strubell, Luke Vilnis, Kate Silverstein, and Andrew McCallum. 2015. Learning Dynamic Feature Selection for Fast Sequential Prediction. In *ACL*.

Charles Sutton, Andrew McCallum, et al. 2012. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning* 4(4):267–373.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *CoNLL*.

Martin J Wainwright and Michael I Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning* 1(1–2):1–305.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia* 57.

Mo Yu, Matthew R. Gormley, and Mark Dredze. 2015. Combining Word Embeddings and Feature Embeddings for Fine-grained Relation Extraction. In *NAACL*.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *ACL*.

Chunk-based Decoder for Neural Machine Translation

Shonosuke Ishiwatar[†*] **Jingtao Yao**[‡*] **Shujie Liu**[§] **Mu Li**[§] **Ming Zhou**[§]
Naoki Yoshinaga[¶] **Masaru Kitsuregawa**[‖¶] **Weijia Jia**[‡]
† The University of Tokyo ‡ Shanghai Jiao Tong University § Microsoft Research Asia
¶ Institute of Industrial Science, the University of Tokyo ‖ National Institute of Informatics
[†¶‖]{ishiwatari, ynaga, kitsure}@tkl.iis.u-tokyo.ac.jp
[‡]{yjt1995@, jia-wj@cs.}sjtu.edu.cn
[§]{shujliu, muli, mingzhou}@microsoft.com

Abstract

Chunks (or phrases) once played a pivotal role in machine translation. By using a chunk rather than a word as the basic translation unit, local (intra-chunk) and global (inter-chunk) word orders and dependencies can be easily modeled. The chunk structure, despite its importance, has not been considered in the decoders used for neural machine translation (NMT). In this paper, we propose chunk-based decoders for NMT, each of which consists of a chunk-level decoder and a word-level decoder. The chunk-level decoder models global dependencies while the word-level decoder decides the local word order in a chunk. To output a target sentence, the chunk-level decoder generates a chunk representation containing global information, which the word-level decoder then uses as a basis to predict the words inside the chunk. Experimental results show that our proposed decoders can significantly improve translation performance in a WAT '16 English-to-Japanese translation task.

1 Introduction

Neural machine translation (NMT) performs end-to-end translation based on a simple encoder-decoder model (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014b) and has now overtaken the classical, complex statistical machine translation (SMT) in terms of performance and simplicity (Sennrich et al., 2016; Luong and Manning, 2016; Cromieres et al., 2016; Neubig, 2016). In NMT, an encoder first maps a source sequence into vector representations and a decoder then maps the vectors into a target sequence (§ 2). This simple framework allows researchers to incorporate the structure of the source sentence as in SMT by leveraging various architectures as the encoder (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014b; Eriguchi et al., 2016b). Most of the NMT models, however, still rely on a sequential decoder based on a recurrent neural network (RNN) due to the difficulty in capturing the structure of a target sentence that is unseen during translation.

En: I wanted to go home earlier.

Ja: 早く (early) 家へ (home) 帰っ (go back) て しまい たい と 思っ (feel) た 。

Figure 1: Translation from English to Japanese. The function words are underlined.

With the sequential decoder, however, there are two problems to be solved. First, it is difficult to model long-distance dependencies (Bahdanau et al., 2015). A hidden state h_t in an RNN is only conditioned by its previous output y_{t-1}, previous hidden state h_{t-1}, and current input x_t. This makes it difficult to capture the dependencies between an older output y_{t-N} if they are too far from the current output. This problem can become more serious when the target sequence becomes longer. For example, in Figure 1, when we translate the English sentence into the Japanese one, after the decoder predicts the content word "帰っ (go back)", it has to predict four function words "て (suffix)", "しまい (*perfect tense*)", "たい (*desire*)", and "と (to)" before predicting the next content word "思っ (feel)". In such a case, the decoder is required to capture the longer dependencies in a target sentence.

Another problem with the sequential decoder is that it is expected to cover multiple possible word orders simply by memorizing the local word se-

*Contribution during internship at Microsoft Research.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1901–1912
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1174

quences in the limited training data. This problem can be more serious in free word-order languages such as Czech, German, Japanese, and Turkish. In the case of the example in Figure 1, the order of the phrase "早く (early)" and the phrase "家へ (to home)" is flexible. This means that simply memorizing the word order in training data is not enough to train a model that can assign a high probability to a correct sentence regardless of its word order.

In the past, chunks (or phrases) were utilized to handle the above problems in statistical machine translation (SMT) (Watanabe et al., 2003; Koehn et al., 2003) and in example-based machine translation (EBMT) (Kim et al., 2010). By using a chunk rather than a word as the basic translation unit, one can treat a sentence as a shorter sequence. This makes it easy to capture the longer dependencies in a target sentence. The order of words in a chunk is relatively fixed while that in a sentence is much more flexible. Thus, modeling intra-chunk (local) word orders and inter-chunk (global) dependencies independently can help capture the difference of the flexibility between the word order and the chunk order in free word-order languages.

In this paper, we refine the original RNN decoder to consider chunk information in NMT. We propose three novel NMT models that capture and utilize the chunk structure in the target language (§ 3). Our focus is the hierarchical structure of a sentence: each sentence consists of chunks, and each chunk consists of words. To encourage an NMT model to capture the hierarchical structure, we start from a hierarchical RNN that consists of a chunk-level decoder and a word-level decoder (Model 1). Then, we improve the word-level decoder by introducing inter-chunk connections to capture the interaction between chunks (Model 2). Finally, we introduce a feedback mechanism to the chunk-level decoder to enhance the memory capacity of previous outputs (Model 3).

We evaluate the three models on the WAT '16 English-to-Japanese translation task (§ 4). The experimental results show that our best model outperforms the best single NMT model reported in WAT '16 (Eriguchi et al., 2016b).

Our contributions are twofold: (1) chunk information is introduced into NMT to improve translation performance, and (2) a novel hierarchical decoder is devised to model the properties of chunk structure in the encoder-decoder framework.

2 Preliminaries: Attention-based Neural Machine Translation

In this section, we briefly introduce the architecture of the attention-based NMT model (Bahdanau et al., 2015), which is the basis of our proposed models.

2.1 Neural Machine Translation

An NMT model usually consists of two connected neural networks: an encoder and a decoder. After the encoder maps a source sentence into a fixed-length vector, the decoder maps the vector into a target sentence. The implementation of the encoder can be a convolutional neural network (CNN) (Kalchbrenner and Blunsom, 2013), a long short-term memory (LSTM) (Sutskever et al., 2014; Luong and Manning, 2016), a gated recurrent unit (GRU) (Cho et al., 2014b; Bahdanau et al., 2015), or a Tree-LSTM (Eriguchi et al., 2016b). While various architectures are leveraged as an encoder to capture the structural information in the source language, most of the NMT models rely on a standard sequential network such as LSTM or GRU as the decoder.

Following (Bahdanau et al., 2015), we use GRU as the recurrent unit in this paper. A GRU unit computes its hidden state vector $\boldsymbol{h}_i$ given an input vector $\boldsymbol{x}_i$ and the previous hidden state $\boldsymbol{h}_{i-1}$:

$$\boldsymbol{h}_i = \mathrm{GRU}(\boldsymbol{h}_{i-1}, \boldsymbol{x}_i). \tag{1}$$

The function $\mathrm{GRU}(\cdot)$ is calculated as

$$\boldsymbol{r}_i = \sigma(\boldsymbol{W}_r \boldsymbol{x}_i + \boldsymbol{U}_r \boldsymbol{h}_{i-1} + \boldsymbol{b}_r), \tag{2}$$
$$\boldsymbol{z}_i = \sigma(\boldsymbol{W}_z \boldsymbol{x}_i + \boldsymbol{U}_z \boldsymbol{h}_{i-1} + \boldsymbol{b}_z), \tag{3}$$
$$\tilde{\boldsymbol{h}}_i = \tanh(\boldsymbol{W} \boldsymbol{x}_i + \boldsymbol{U}(\boldsymbol{r}_i \odot \boldsymbol{h}_{i-1} + \boldsymbol{b})), \tag{4}$$
$$\boldsymbol{h}_i = (\boldsymbol{1} - \boldsymbol{z}_i) \odot \tilde{\boldsymbol{h}}_i + \boldsymbol{z}_i \odot \boldsymbol{h}_{i-1}, \tag{5}$$

where vectors $\boldsymbol{r}_i$ and $\boldsymbol{z}_i$ are reset gate and update gate, respectively. While the former gate allows the model to forget the previous states, the latter gate decides how much the model updates its content. All the $\boldsymbol{W}$s and $\boldsymbol{U}$s, or the $\boldsymbol{b}$s above are trainable matrices or vectors. $\sigma(\cdot)$ and $\odot$ denote the sigmoid function and element-wise multiplication operator, respectively.

In this simple model, we train a GRU function that encodes a source sentence $\{x_1, \cdots, x_I\}$ into a single vector $\boldsymbol{h}_I$. At the same time, we jointly train another GRU function that decodes $\boldsymbol{h}_I$ to the target sentence $\{y_1, \cdots, y_J\}$. Here, the j-th word in the

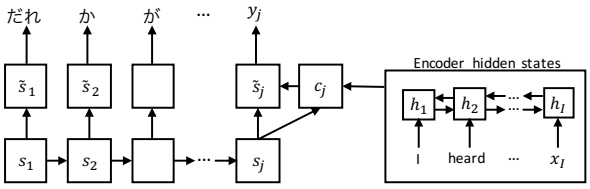


Figure 2: Standard word-based decoder.

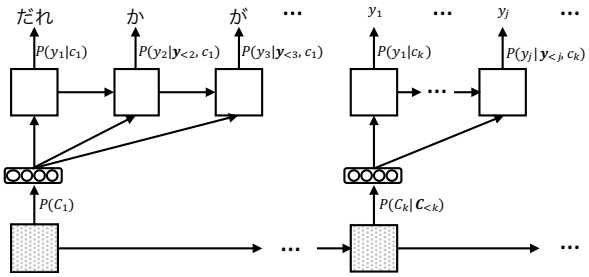


Figure 3: Chunk-based decoder. The top layer (word-level decoder) illustrates the first term in Eq. (15) and the bottom layer (chunk-level decoder) denotes the second term.

target sentence y_j can be predicted with this decoder GRU and a nonlinear function $g(\cdot)$ followed by a softmax layer, as

$$\boldsymbol{c} = \boldsymbol{h}_I, \tag{6}$$

$$\boldsymbol{s}_j = \text{GRU}(\boldsymbol{s}_{j-1}, [\boldsymbol{y}_{j-1}; \boldsymbol{c}]), \tag{7}$$

$$\tilde{\boldsymbol{s}}_j = g(\boldsymbol{y}_{j-1}, \boldsymbol{s}_j, \boldsymbol{c}), \tag{8}$$

$$P(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}) = \text{softmax}(\tilde{\boldsymbol{s}}_j), \tag{9}$$

where $\boldsymbol{c}$ is a context vector of the encoded sentence and $\boldsymbol{s}_j$ is a hidden state of the decoder GRU.

Following Bahdanau et al. (2015), we use a mini-batch stochastic gradient descent (SGD) algorithm with ADADELTA (Zeiler, 2012) to train the above two GRU functions (i.e., the encoder and the decoder) jointly. The objective is to minimize the cross-entropy loss of the training data $\boldsymbol{D}$, as

$$J = \sum_{(\boldsymbol{x},\boldsymbol{y})\in\boldsymbol{D}} -\log P(\boldsymbol{y}|\boldsymbol{x}). \tag{10}$$

2.2 Attention Mechanism for Neural Machine Translation

To use all the hidden states of the encoder and improve the translation performance of long sentences, Bahdanau et al. (2015) proposed using an attention mechanism. In the attention model, the context vector is not simply the last encoder state h_I but rather the weighted sum of all hidden states of the bidirectional GRU, as follows:

$$\boldsymbol{c}_j = \sum_{i=1}^{I} \alpha_{ji} \boldsymbol{h}_i. \tag{11}$$

Here, the weight α_{ji} decides how much a source word x_i contributes to the target word y_j. α_{ji} is computed by a feedforward layer and a softmax layer as

$$e_{ji} = \boldsymbol{v} \cdot \tanh(\boldsymbol{W}_e \boldsymbol{h}_i + \boldsymbol{U}_e \boldsymbol{s}_j + \boldsymbol{b}_e), \tag{12}$$

$$\alpha_{ji} = \frac{\exp(e_{ji})}{\sum_{j'=1}^{J} \exp(e_{j'i})}, \tag{13}$$

where $\boldsymbol{W}_e$, $\boldsymbol{U}_e$ are trainable matrices and the $\boldsymbol{v}$, $\boldsymbol{b}_e$ are trainable vectors.[1] In a decoder using the attention mechanism, the obtained context vector $\boldsymbol{c}_j$ in each time step replaces $\boldsymbol{c}$s in Eqs. (7) and (8). An illustration of the NMT model with the attention mechanism is shown in Figure 2.

The attention mechanism is expected to learn alignments between source and target words, and plays a similar role to the translation model in phrase-based SMT (Koehn et al., 2003).

3 Neural Machine Translation with Chunk-based Decoder

Taking non-sequential information such as chunks (or phrases) structure into consideration has proved helpful for SMT (Watanabe et al., 2003; Koehn et al., 2003) and EBMT (Kim et al., 2010). Here, we focus on two important properties of chunks (Abney, 1991): (1) The word order in a chunk is almost always fixed, and (2) A chunk consists of a few (typically one) content words surrounded by zero or more function words.

To fully utilize the above properties of a chunk, we propose modeling the intra-chunk and the inter-chunk dependencies independently with a "chunk-by-chunk" decoder (See Figure 3). In the standard word-by-word decoder described in § 2, a target word y_j in the target sentence $\boldsymbol{y}$ is predicted by taking the previous outputs $\boldsymbol{y}_{<j}$ and the source sentence $\boldsymbol{x}$ as input:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{j=1}^{J} P(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}), \tag{14}$$

where J is the length of the target sentence. Not

[1] We choose this implementation following (Luong et al., 2015b), while (Bahdanau et al., 2015) use s_{j-1} instead of s_j in Eq. (12).

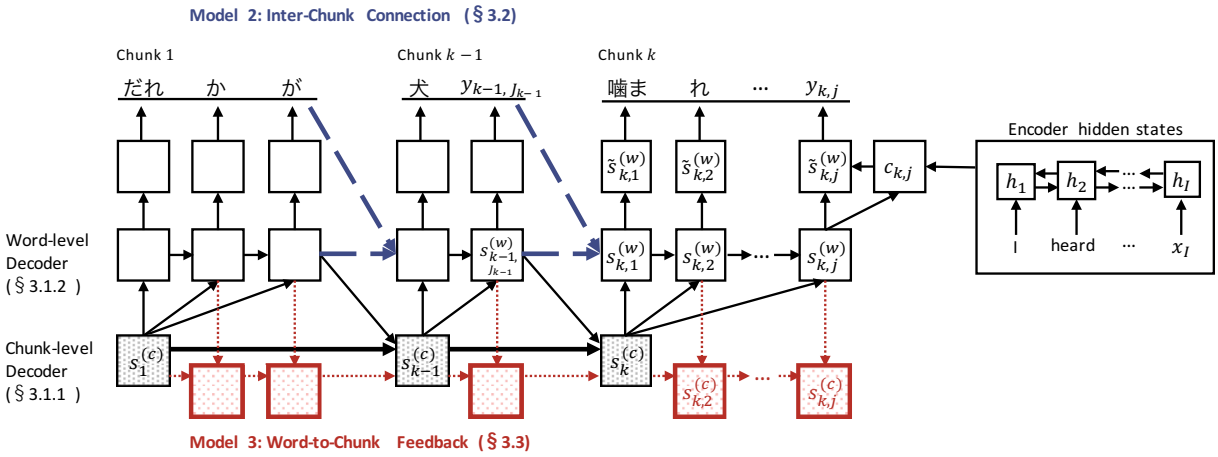


Figure 4: Proposed model: NMT with chunk-based decoder. A chunk-level decoder generates a chunk representation for each chunk while a word-level decoder uses the representation to predict each word. The solid lines in the figure illustrate Model 1. The dashed blue arrows in the word-level decoder denote the connections added in Model 2. The dotted red arrows in the chunk-level decoder denote the feedback states added in Model 3; the connections in the thick black arrows are replaced with the dotted red arrows.

assuming any structural information of the target language, the sequential decoder has to memorize long dependencies in a sequence. To release the model from the pressure of memorizing the long dependencies over a sentence, we redefine this problem as the combination of a word prediction problem and a chunk generation problem:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{k=1}^{K} \left\{ P(c_k|\boldsymbol{c}_{<k}, \boldsymbol{x}) \prod_{j=1}^{J_k} P(y_j|\boldsymbol{y}_{<j}, c_k, \boldsymbol{x}) \right\}, \tag{15}$$

where K is the number of chunks in the target sentence and J_k is the length of the k-th chunk (see Figure 3). The first term represents the generation probability of a chunk c_k and the second term indicates the probability of a word y_j in the chunk. We model the former term as a chunk-level decoder and the latter term as a word-level decoder. As demonstrated later in § 4, both K and J_k are much shorter than the sentence length J, which is why our decoders do not have to capture the long dependencies like the standard decoder does.

In the above formulation, we model the information of words and their orders in a chunk. No matter which language we target, we can assume that a chunk usually consists of some content words and function words, and the word order in the chunk is almost always fixed (Abney, 1991). Although our idea can be used in several languages, the optimal network architecture could depend on the word order of the target language. In this work, we design models for languages in which content words are followed by function words, such as Japanese and Korean. The details of our models are described in the following sections.

3.1 Model 1: Basic Chunk-based Decoder

The model described in this section is the basis of our proposed decoders. It consists of two parts: a chunk-level decoder (§ 3.1.1) and a word-level decoder (§ 3.1.2). The part drawn in black solid lines in Figure 4 illustrates the architecture of Model 1.

3.1.1 Chunk-level Decoder

Our chunk-level decoder (see Figure 3) outputs a chunk representation. The chunk representation contains the information about words that should be predicted by the word-level decoder.

To generate the representation of the k-th chunk $\tilde{\boldsymbol{s}}_k^{(c)}$, the chunk-level decoder (see the bottom layer in Figure 4) takes the last states of the word-level decoder $\boldsymbol{s}_{k-1,J_{k-1}}^{(w)}$ and updates its hidden state $\boldsymbol{s}_k^{(c)}$ as:

$$\boldsymbol{s}_k^{(c)} = \mathrm{GRU}(\boldsymbol{s}_{k-1}^{(c)}, \boldsymbol{s}_{k-1,J_{k-1}}^{(w)}), \tag{16}$$

$$\tilde{\boldsymbol{s}}_k^{(c)} = \boldsymbol{W}_c \boldsymbol{s}_k^{(c)} + \boldsymbol{b}_c. \tag{17}$$

The obtained chunk representation $\tilde{\boldsymbol{s}}_k^{(c)}$ continues to be fed into the word-level decoder until it outputs all the words in the current chunk.

3.1.2 Word-level Decoder

Our word-level decoder (see Figure 4) differs from the standard sequential decoder described in § 2 in

that it takes the chunk representation $\tilde{s}_k^{(c)}$ as input:

$$\boldsymbol{s}_{k,j}^{(w)} = \mathrm{GRU}(\boldsymbol{s}_{k,j-1}^{(w)}, [\tilde{\boldsymbol{s}}_k^{(c)}; \boldsymbol{y}_{k,j-1}; \boldsymbol{c}_{k,j-1}]), \quad (18)$$

$$\tilde{\boldsymbol{s}}_{k,j}^{(w)} = g(\boldsymbol{y}_{k,j-1}, \boldsymbol{s}_{k,j}^{(w)}, \boldsymbol{c}_{k,j}), \quad (19)$$

$$P(y_{k,j}|\boldsymbol{y}_{<j}, \boldsymbol{x}) = \mathrm{softmax}(\tilde{\boldsymbol{s}}_{k,j}^{(w)}). \quad (20)$$

In a standard sequential decoder, the hidden state iterates over the length of a target sentence and then generates an end-of-sentence token. In other words, its hidden layers are required to memorize the long-term dependencies and orders in the target language. In contrast, in our word-level decoder, the hidden state iterates only over the length of a chunk and then generates an end-of-chunk token. Thus, our word-level decoder is released from the pressure of memorizing the long (inter-chunk) dependencies and can focus on learning the short (intra-chunk) dependencies.

3.2 Model 2: Inter-Chunk Connection

The second term in Eq. (15) only iterates over one chunk ($j = 1$ to J_k). This means that the last state and the last output of a chunk are not being fed into the word-level decoder at the next time step (see the black part in Figure 4). In other words, $\boldsymbol{s}_{k,1}^{(w)}$ in Eq. (18) is always initialized before generating the first word in a chunk. This may have a bad influence on the word-level decoder because it cannot access any previous information at the first word of each chunk.

To address this problem, we add new connections to Model 1 between the first state in a chunk and the last state in the previous chunk, as

$$\boldsymbol{s}_{k,1}^{(w)} = \mathrm{GRU}(\boldsymbol{s}_{k-1,J_{k-1}}^{(w)}, [\tilde{\boldsymbol{s}}_k^{(c)}; \boldsymbol{y}_{k-1,J_{k-1}}; \boldsymbol{c}_{k-1,J_{k-1}}]). \quad (21)$$

The dashed blue arrows in Figure 4 illustrate the added inter-chunk connections.

3.3 Model 3: Word-to-Chunk Feedback

The chunk-level decoder in Eq. (16) is only conditioned by $\boldsymbol{s}_{k-1,J_{k-1}}^{(w)}$, the last word state in each chunk (see the black part in Figure 4). This may affect the chunk-level decoder because it cannot memorize what kind of information has already been generated by the word-level decoder. The information about the words in a chunk should not be included in the representation of the next chunk; otherwise, it may generate the same chunks multiple times, or forget to translate some words in the source sentence.

To encourage the chunk-level decoder to memorize the information about the previous outputs more carefully, we add feedback states to our chunk-level decoder in Model 2. The feedback state in the chunk-level decoder is updated at every time step $j(> 1)$ in k-th chunk, as

$$\boldsymbol{s}_{k,j}^{(c)} = \mathrm{GRU}(\boldsymbol{s}_{k,j-1}^{(c)}, \boldsymbol{s}_{k,j}^{(w)}). \quad (22)$$

The red part in Figure 4 illustrate the added feedback states and their connections. The connections in the thick black arrows are replaced with the dotted red arrows in Model 3.

4 Experiments

4.1 Setup

Data To examine the effectiveness of our decoders, we chose Japanese, a free word-order language, as the target language. Japanese sentences are easy to break into well-defined chunks (called *bunsetsus* (Hashimoto, 1934) in Japanese). For example, the accuracy of *bunsetsu*-chunking on newspaper articles is reported to be over 99% (Murata et al., 2000; Yoshinaga and Kitsuregawa, 2014). The effect of chunking errors in training the decoder can be suppressed, which means we can accurately evaluate the potential of our method. We used the English-Japanese training corpus in the Asian Scientific Paper Excerpt Corpus (ASPEC) (Nakazawa et al., 2016), which was provided in WAT '16. To remove inaccurate translation pairs, we extracted the first two million out of the 3 million pairs following the setting that gave the best performances in WAT '15 (Neubig et al., 2015).

Preprocessings For Japanese sentences, we performed tokenization using KyTea 0.4.7[2] (Neubig et al., 2011). Then we performed *bunsetsu*-chunking with J.DepP 2015.10.05[3] (Yoshinaga and Kitsuregawa, 2009, 2010, 2014). Special end-of-chunk tokens were inserted at the end of the chunks. Our word-level decoders described in § 3 will stop generating words after each end-of-chunk token. For English sentences, we performed the same preprocessings described on the WAT '16 Website.[4] To suppress having possible

[2] http://www.phontron.com/kytea/

[3] http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/jdepp/

[4] http://lotus.kuee.kyoto-u.ac.jp/WAT/baseline/dataPreparationJE.html

Corpus	# words	# chunks	# sentences
Train	49,671,230	15,934,129	1,663,780
Dev.	54,287	-	1,790
Test	54,088	-	1,812

Table 1: Statistics of the target language (Japanese) in extracted corpus after preprocessing.

ρ of ADADELTA	0.95
ϵ of ADADELTA	$1e^{-6}$
Initial learning rate	1.0
Gradient clipping	1.0
Mini-batch size	64
d_{hid} (dimension of hidden states)	1024
d_{emb} (dimension of word embeddings)	1024

Table 2: Hyperparameters for training.

chunking errors affect the translation quality, we removed extremely long chunks from the training data. Specifically, among the 2 million preprocessed translation pairs, we excluded sentence pairs that matched any of following conditions: (1) The length of the source sentence or target sentence is larger than 64 (3% of whole data); (2) The maximum length of a chunk in the target sentence is larger than 8 (14% of whole data); and (3) The maximum number of chunks in the target sentence is larger than 20 (3% of whole data). Table 1 shows the details of the extracted data.

Postprocessing To perform unknown word replacement (Luong et al., 2015a), we built a bilingual English-Japanese dictionary from all of the three million translation pairs. The dictionary was extracted with the MGIZA++ 0.7.0[5] (Och and Ney, 2003; Gao and Vogel, 2008) word alignment tool by automatically extracting the alignments between English words and Japanese words.

Model Architecture Any encoder can be combined with our decoders. In this work, we adopted a single-layer bidirectional GRU (Cho et al., 2014b; Bahdanau et al., 2015) as the encoder to focus on confirming the impact of the proposed decoders. We used single layer GRUs for the word-level decoder and the chunk-level decoder. The vocabulary sizes were set to 40k for source side and 30k for target side, respectively. The conditional probability of each target word was computed with a deep-output (Pascanu et al., 2014) layer with maxout (Goodfellow et al., 2013) units following (Bahdanau et al., 2015). The maximum number of output chunks was set to 20 and the maximum length of a chunk was set to 8.

Training Details The models were optimized using ADADELTA following (Bahdanau et al., 2015). The hyperparameters of the training procedure were fixed to the values given in Table 2. Note that the learning rate was halved when the BLEU score on the development set did not increase for 30,000 batches. All the parameters were initialized randomly with Gaussian distribution. It took about a week to train each model with an NVIDIA TITAN X (Pascal) GPU.

Evaluation Following the WAT '16 evaluation procedure, we used BLEU (Papineni et al., 2002) and RIBES (Isozaki et al., 2010) to evaluate our models. The BLEU scores were calculated with `multi-bleu.pl` in Moses 2.1.1[6] (Koehn et al., 2007); RIBES scores were calculated with `RIBES.py` 1.03.1[7] (Isozaki et al., 2010). Following Cho et al. (2014a), we performed beam search[8] with length-normalized log-probability to decode target sentences. We saved the trained models that performed best on the development set during training and used them to evaluate the systems with the test set.

Baseline Systems The baseline systems and the important hyperparamters are listed in Table 3. Eriguchi et al. (2016a)'s baseline system (the first line in Table 3) was the best single (w/o ensembling) word-based NMT system that were reported in WAT '16. For a more fair evaluation, we also reimplemented a standard attention-based NMT system that uses exactly the same encoder, training procedure, and the hyperparameters as our proposed models, but has a word-based decoder. We trained this system on the training data without chunk segmentations (the second line in Table 3) and with chunk segmentations given by J.DepP (the third line in Table 3). The chunked corpus fed to the third system is exactly the same as the training data of our proposed systems (sixth to eighth lines in Table 3). In addition, we also include the Tree-to-Sequence models (Eriguchi et al., 2016a,b) (the fourth and fifth lines in Table 3) to compare the impact of capturing the structure in the source language and that in

[5] https://github.com/moses-smt/mgiza

[6] http://www.statmt.org/moses/

[7] http://www.kecl.ntt.co.jp/icl/lirg/ribes/index.html

[8] Beam size is set to 20.

System Encoder Type	/	Decoder Type	Hyperparameter $\|V_{src}\|$	$\|V_{trg}\|$	d_{emb}	d_{hid}	BLEU	RIBES	Dec. time [ms/sent.]
Word-based	/	Word-based (Eriguchi et al., 2016a)	88k	66k	512	512	34.64	81.60	-
	/	Word-based (our implementation)	40k	30k	1024	1024	36.33	81.22	84.1
		+ chunked training data via J.DepP	40k	30k	1024	1024	35.71	80.89	101.5
Tree-based	/	Word-based (Eriguchi et al., 2016b)	88k	66k	512	512	34.91	81.66	(363.7)[9]
	/	Char-based (Eriguchi et al., 2016a)	88k	3k	256	512	31.52	79.39	(**8.8**)[9]
Word-based	/	Proposed Chunk-based (Model 1)	40k	30k	1024	1024	34.70	81.01	165.2
		+ Inter-chunk connection (Model 2)	40k	30k	1024	1024	35.81	81.29	165.2
		+ Word-to-chunk feedback (Model 3)	40k	30k	1024	1024	**37.26**	**82.23**	163.7

Table 3: The settings and results of the baseline systems and our systems. $|V_{src}|$ and $|V_{trg}|$ denote the vocabulary size of the source language and the target language, respectively. d_{emb} and d_{hid} are the dimension size of the word embeddings and hidden states, respectively. Only single NMT models (w/o ensembling) reported in WAT '16 are listed here. Full results are available on the WAT '16 Website.[10]

the target language. Note that all systems listed in Table 3, including our models, are single models without ensemble techniques.

4.2 Results

Proposed Models vs. Baselines Table 3 shows the experimental results on the ASPEC test set. We can observe that our best model (Model 3) outperformed all the single NMT models reported in WAT '16. The gain obtained by switching Word-based decoder to Chunk-based decoder ($+0.93$ BLEU and $+1.01$ RIBES) is larger than the gain obtained by switching word-based encoder to Tree-based encoder ($+0.27$ BLEU and $+0.06$ RIBES). This result shows that capturing the chunk structure in the target language is more effective than capturing the syntax structure in the source language. Compared with the character-based NMT model (Eriguchi et al., 2016a), our Model 3 performed better by $+5.74$ BLEU score and $+2.84$ RIBES score. One possible reason for this is that using a character-based model rather than a word-based model makes it more difficult to capture long-distance dependencies because the length of a target sequence becomes much longer in the character-based model.

Comparison between Baselines Among the five baselines, our reimplementation without chunk segmentations (the second line in Table 3) achieved the best BLEU score while the Eriguchi et al. (2016b)'s system (the fourth line in Table 3) achieved the best RIBES score. The most probable reasons for the superiority of our reimplementation over the Eriguchi et al. (2016a)'s word-based baseline (the first line in Table 3) is that the dimensions of word embeddings and hidden states in our systems are higher than theirs.

Feeding chunked training data to our baseline system (the third line in Table 3) instead of a normal data caused bad effects by -0.62 BLEU score and by -0.33 RIBES score. We evaluated the chunking ability of this system by comparing the positions of end-of-chunk tokens generated by this system with the chunk boundaries obtained by J.DepP. To our surprise, this word-based decoder could output chunk separations as accurate as our proposed Model 3 (both systems achieved F_1-score > 97). The results show that even a standard word-based decoder has the ability to predict chunk boundaries if they are given in training data. However, it is difficult for the word-based decoder to utilize the chunk information to improve the translation quality.

Decoding Speed Although the chunk-based decoder runs 2x slower than our word-based decoder, it is still practically acceptable (6 sentences per second). The character-based decoder (the fifth line in Table 3) is less time-consuming mainly because of its small vocabulary size ($|V_{trg}| = 3$k).

Chunk-level Evaluation To confirm that our models can capture local (intra-chunk) and global (inter-chunk) word orders well, we evaluated the translation quality at the chunk level. First, we performed *bunsetsu*-chunking on the reference translations in the test set. Then, for both reference translations and the outputs of our systems, we combined all the words in each chunk into a single token to regard a chunk as the basic translation unit instead of a word. Finally, we computed the chunk-based BLEU (C-BLEU) and RIBES

[9] Tree-to-Seq models are tested on CPUs instead of GPUs.

[10] http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation

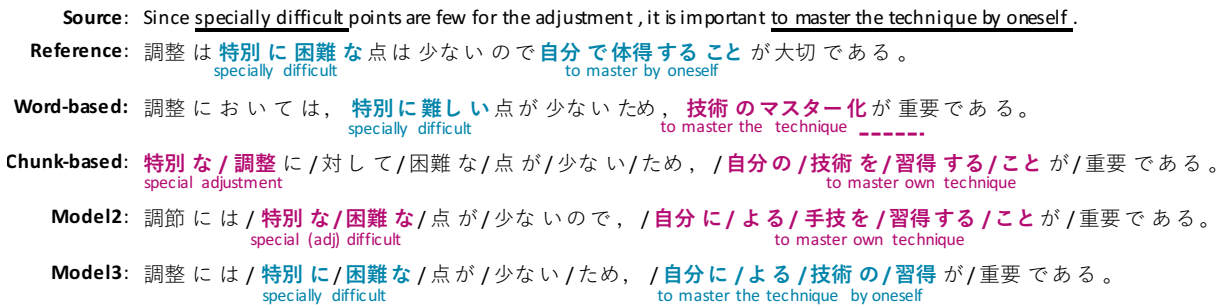


Figure 5: Translation examples. "/" denote chunk boundaries that are automatically determined by our decoders. Words colored blue and red respectively denote correct translations and wrong translations.

Decoder	C-BLEU	C-RIBES
Word-based (our implementation)	7.56	50.73
+ chunked training data via J.DepP	7.40	51.18
Proposed Chunk-based (Model 1)	7.59	50.47
+ Inter-chunk connection (Model 2)	7.78	51.48
+ Word-to-chunk feedback (Model 3)	**8.69**	**52.82**

Table 4: Chunk-based BLEU and RIBES with the systems using the word-based encoder.

(C-RIBES). The results are listed in Table 4. For the word-based decoder (the first line in Table 4), we performed *bunsetsu*-chunking by J.DepP on its outputs to obtain chunk boundaries. As another baseline (the second line in Table 4), we used the chunked sentences as training data instead of performing chunking after decoding. The results show that our models (Model 2 and Model 3) outperform the word-based decoders in both C-BLEU and C-RIBES. This indicates that our chunk-based decoders can produce more correct chunks in a more correct order than the word-based models.

Qualitative Analysis To clarify the qualitative difference between the word-based decoder and our chunk-based decoders, we show translation examples in Figure 5. Words in blue and red respectively denote correct translations and wrong translations. The word-based decoder (our implementation) has completely dropped the translation of "by oneself." On the other hand, Model 1 generated a slightly wrong translation "自分の技術を習得すること (to master own technique)." In addition, Model 1 has made another serious word-order error "特別な調整 (special adjustment)." These results suggest that Model 1 can capture longer dependencies in a long sequence than the word-based decoder. However, Model 1 is not good at modeling global word order because it cannot access enough information about previous outputs. The weakness of modeling word order was overcome in Model 2 thanks to the inter-chunk connections. However, Model 2 still suffered from the errors of function words: it still generates a wrong chunk "特別な (special)" instead of the correct one "特別に (specially)" and a wrong chunk "よる" instead of "より." Although these errors seem trivial, such mistakes with function words bring serious changes of sentence meaning. However, all of these problems have disappeared in Model 3. This phenomenon supports the importance of the feedback states to provide the decoder with a better ability to choose more accurate words in chunks.

5 Related Work

Much work has been done on using chunk (or phrase) structure to improve machine translation quality. The most notable work involved phrase-based SMT (Koehn et al., 2003), which has been the basis for a huge amount of work on SMT for more than ten years. Apart from this, Watanabe et al. (2003) proposed a chunk-based translation model that generates output sentences in a chunk-by-chunk manner. The chunk structure is effective not only for SMT but also for example-based machine translation (EBMT). Kim et al. (2010) proposed a chunk-based EBMT and showed that using chunk structures can help with finding better word alignments. Our work is different from theirs in that our models are based on NMT, but not SMT or EBMT. The decoders in the above studies can model the chunk structure by storing chunk pairs in a large table. In contrast, we do that by individually training a chunk generation model and a word prediction model with two RNNs.

While most of the NMT models focus on the conversion between sequential data, some works have tried to incorporate non-sequential informa-

tion into NMT (Eriguchi et al., 2016b; Su et al., 2017). Eriguchi et al. (2016b) use a Tree-based LSTM (Tai et al., 2015) to encode input sentence into context vectors. Given a syntactic tree of a source sentence, their tree-based encoder encodes words from the leaf nodes to the root nodes recursively. Su et al. (2017) proposed a lattice-based encoder that considers multiple tokenization results while encoding the input sentence. To prevent the tokenization errors from propagating to the whole NMT system, their attice-based encoder can utilize multiple tokenization results. These works focus on the encoding process and propose better encoders that can exploit the structures of the source language. In contrast, our work focuses on the decoding process to capture the structure of the target language. The encoders described above and our proposed decoders are complementary so they can be combined into a single network.

Considering that our Model 1 described in § 3.1 can be seen as a hierarchical RNN, our work is also related to previous studies that utilize multi-layer RNNs to capture hierarchical structures in data. Hierarchical RNNs are used for various NLP tasks such as machine translation (Luong and Manning, 2016), document modeling (Li et al., 2015; Lin et al., 2015), dialog generation (Serban et al., 2017), image captioning (Krause et al., 2016), and video captioning (Yu et al., 2016). In particular, Li et al. (2015) and Luong and Manning (2016) use hierarchical encoder-decoder models, but not for the purpose of learning syntactic structures of target sentences. Li et al. (2015) build hierarchical models at the sentence-word level to obtain better document representations. Luong and Manning (2016) build the word-character level to cope with the out-of-vocabulary problem. In contrast, we build a hierarchical models at the chunk-word level to explicitly capture the syntactic structure based on chunk segmentation.

In addition, the architecture of Model 3 is also related to stacked RNN, which has shown to be effective in improving the translation quality (Luong et al., 2015a; Sutskever et al., 2014). Although these architectures look similar to each other, there is a fundamental difference between the directions of the connection between two layers. A stacked RNN consists of multiple RNN layers that are connected from the input side to the output side at every time step. In contrast, our Model 3 has a different connection at each time step. Before it generates a chunk, there is a feed-forward connection from the chunk-level decoder to the word-level decoder. However, after generating a chunk representation, the connection is to be reversed to feed back the information from the word-level decoder to the chunk-level decoder. By switching the connections between two layers, our model can capture the chunk structure explicitly. This is the first work that proposes decoders for NMT that can capture plausible linguistic structures such as chunk.

Finally, we noticed that (Zhou et al., 2017) (which is accepted at the same time as this paper) have also proposed a chunk-based decoder for NMT. Their good experimental result on Chinese to English translation task also indicates the effectiveness of "chunk-by-chunk" decoders. Although their architecture is similar to our Model 2, there are several differences: (1) they adopt chunk-level attention instead of word-level attention; (2) their model predicts chunk tags (such as noun phrase), while ours only predicts chunk boundaries; and (3) they employ a boundary gate to decide the chunk boundaries, while we do that by simply having the model generate end-of-chunk tokens.

6 Conclusion

In this paper, we propose chunk-based decoders for NMT. As the attention mechanism in NMT plays a similar role to the translation model in phrase-based SMT, our chunk-based decoders are intended to capture the notion of chunks in chunk-based (or phrase-based) SMT. We utilize the chunk structure to efficiently capture long-distance dependencies and cope with the problem of free word-order languages such as Japanese. We designed three models that have hierarchical RNN-like architectures, each of which consists of a word-level decoder and a chunk-level decoder. We performed experiments on the WAT '16 English-to-Japanese translation task and found that our best model outperforms the strongest baselines by $+0.93$ BLEU score and by $+0.57$ RIBES score.

In future work, we will explore the optimal structures of chunk-based decoder for other free word-order languages such as Czech, German, and Turkish. In addition, we plan to combine our decoder with other encoders that capture language structure, such as a hierarchical RNN (Luong and Manning, 2016), a Tree-LSTM (Eriguchi et al., 2016b), or an order-free encoder, such as a CNN (Kalchbrenner and Blunsom, 2013).

Acknowledgements

This research was partially supported by the Research and Development on Real World Big Data Integration and Analysis program of the Ministry of Education, Culture, Sports, Science and Technology (MEXT) and RIKEN, Japan, and by the Chinese National Research Fund (NSFC) Key Project No. 61532013 and National China 973 Project No. 2015CB352401.

The authors appreciate Dongdong Zhang, Shuangzhi Wu, and Zhirui Zhang for the fruitful discussions during the first and second authors were interns at Microsoft Research Asia. We also thank Masashi Toyoda and his group for letting us use their computing resources. Finally, we thank the anonymous reviewers for their careful reading of our paper and insightful comments.

References

Steven P. Abney. 1991. Parsing by chunks. In *Principle-based parsing*, Springer, pages 257–278.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR)*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST)*. pages 103–111.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1724–1734.

Fabien Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto university participation to WAT 2016. In *Proceedings of the Third Workshop on Asian Translation (WAT)*. pages 166–174.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016a. Character-based decoding in tree-to-sequence attention-based neural machine translation. In *Proceedings of the Third Workshop on Asian Translation (WAT)*. pages 175–183.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016b. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 823–833.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*. pages 49–57.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*. pages 1319–1327.

Shinkichi Hashimoto. 1934. *Kokugoho Yosetsu*. Meiji Shoin.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 944–952.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1700–1709.

Jae Dong Kim, Ralf D. Brown, and Jaime G. Carbonell. 2010. Chunk-based EBMT. In *Proceedings of the 14th workshop of the European Association for Machine Translation (EAMT)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 177–180.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*. pages 48–54.

Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. 2016. A hierarchical approach for generating descriptive image paragraphs. In *arXiv:1611.06607 [cs.CV]*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. pages 1106–1115.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 899–907.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 1054–1063.

Minh-Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015a. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the Seventh International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. pages 11–19.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1412–1421.

Masaki Murata, Kiyotaka Uchimoto, Qing Ma, and Hitoshi Isahara. 2000. Bunsetsu identification using category-exclusive rules. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*. pages 565–571.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*. pages 2204–2208.

Graham Neubig. 2016. Lexicons and minimum risk training for neural machine translation: NAIST-CMU at WAT2016. In *Proceedings of the Third Workshop on Asian Translation (WAT)*. pages 119–125.

Graham Neubig, Makoto Morishita, and Satoshi Nakamura. 2015. Neural reranking improves subjective quality of machine translation: NAIST at WAT2015. In *Proceedings of the Second Workshop on Asian Translation (WAT)*. pages 35–41.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*. pages 529–533.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 311–318.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. In *Proceedings of the Second International Conference on Learning Representations (ICLR)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation (WMT)*. pages 371–376.

Iulian V. Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NIPS)*. pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. pages 1556–1566.

Taro Watanabe, Eiichiro Sumita, and Hiroshi G. Okuno. 2003. Chunk-based statistical translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 303–310.

Naoki Yoshinaga and Masaru Kitsuregawa. 2009. Polynomial to linear: Efficient classification with conjunctive features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1542–1551.

Naoki Yoshinaga and Masaru Kitsuregawa. 2010. Kernel slicing: Scalable online training with conjunctive features. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*. pages 1245–1253.

Naoki Yoshinaga and Masaru Kitsuregawa. 2014. A self-adaptive classifier for efficient text-stream processing. In *Proceedings of the 25th International*

Conference on Computational Linguistics (COLING). pages 1091–1102.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pages 4584–4593.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. In *arXiv:1212.5701 [cs.LG]*.

Hao Zhou, Zhaopeng Tu, Shujian Huang, Xiaohua Liu, Hang Li, and Jiajun Chen. 2017. Chunk-based bi-scale decoder for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Doubly-Attentive Decoder for Multi-modal Neural Machine Translation

Iacer Calixto
ADAPT Centre
School of Computing
Dublin City University
Dublin, Ireland

Qun Liu
ADAPT Centre
School of Computing
Dublin City University
Dublin, Ireland

Nick Campbell
ADAPT Centre
Speech Communication Lab
Trinity College Dublin
Dublin 2, Ireland

{iacer.calixto,qun.liu,nick.campbell}@adaptcentre.ie

Abstract

We introduce a Multi-modal Neural Machine Translation model in which a doubly-attentive decoder naturally incorporates *spatial* visual features obtained using pre-trained convolutional neural networks, bridging the gap between image description and translation. Our decoder learns to attend to source-language words and parts of an image independently by means of two *separate attention mechanisms* as it generates words in the target language. We find that our model can efficiently exploit not just back-translated in-domain multi-modal data but also large general-domain text-only MT corpora. We also report state-of-the-art results on the Multi30k data set.

1 Introduction

Neural Machine Translation (NMT) has been successfully tackled as a *sequence to sequence* learning problem (Kalchbrenner and Blunsom, 2013; Cho et al., 2014b; Sutskever et al., 2014) where each training example consists of one source and one target variable-length sequences, with no prior information on the alignment between the two.

In the context of NMT, Bahdanau et al. (2015) first proposed to use an *attention mechanism* in the decoder, which is trained to attend to the relevant source-language words as it generates each word of the target sentence. Similarly, Xu et al. (2015) proposed an attention-based model for the task of image description generation (IDG) where a model learns to attend to specific parts of an image representation (the source) as it generates its description (the target) in natural language.

We are inspired by recent successes in applying attention-based models to NMT and IDG. In this work, we propose an end-to-end attention-based multi-modal neural machine translation (MNMT) model which effectively incorporates *two independent attention mechanisms*, one over source-language words and the other over different areas of an image.

Our main contributions are:

- We propose a novel attention-based MNMT model which incorporates spatial visual features in a separate visual attention mechanism;
- We use a medium-sized, back-translated multi-modal in-domain data set and large general-domain text-only MT corpora to pre-train our models and show that our MNMT model can efficiently exploit both;
- We show that images bring useful information into an NMT model, e.g. in situations in which sentences describe objects illustrated in the image.

To the best of our knowledge, previous MNMT models in the literature that utilised spatial visual features did not significantly improve over a comparable model that used global visual features or even only textual features (Caglayan et al., 2016a; Calixto et al., 2016; Huang et al., 2016; Libovický et al., 2016; Specia et al., 2016). In this work, we wish to address this issue and propose an MNMT model that uses, in addition to an attention mechanism over the source-language words, an additional visual attention mechanism to incorporate spatial visual features, and still improves on simpler text-only and multi-modal attention-based NMT models.

The remainder of this paper is structured as follows. We first briefly revisit the attention-based NMT framework (§2) and expand it into an MNMT framework (§3). In §4, we introduce the

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1913–1924
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1175

datasets we use to train and evaluate our models, in §5 we discuss our experimental setup and analyse and discuss our results. Finally, in §6 we discuss relevant related work and in §7 we draw conclusions and provide avenues for future work.

2 Background and Notation

2.1 Attention-based NMT

In this section, we describe the attention-based NMT model introduced by Bahdanau et al. (2015). Given a source sequence $X = (x_1, x_2, \cdots, x_N)$ and its translation $Y = (y_1, y_2, \cdots, y_M)$, an NMT model aims to build a single neural network that translates X into Y by directly learning to model $p(Y \mid X)$. The entire network consists of one *encoder* and one *decoder* with one *attention mechanism*, typically implemented as two Recurrent Neural Networks (RNN) and one multilayer perceptron, respectively. Each x_i is a row index in a source lookup or word embedding matrix $\boldsymbol{E_x} \in \mathbb{R}^{|V_x| \times d_x}$, as well as each y_j being an index in a target lookup or word embedding matrix $\boldsymbol{E_y} \in \mathbb{R}^{|V_y| \times d_y}$, V_x and V_y are source and target vocabularies, and d_x and d_y are source and target word embeddings dimensionalities, respectively.

The encoder is a bi-directional RNN with GRU (Cho et al., 2014a), where a forward RNN $\overrightarrow{\Phi}_{\text{enc}}$ reads X word by word, from left to right, and generates a sequence of *forward annotation vectors* $(\overrightarrow{\boldsymbol{h}}_1, \overrightarrow{\boldsymbol{h}}_2, \cdots, \overrightarrow{\boldsymbol{h}}_N)$ at each encoder time step $i \in [1, N]$. Similarly, a backward RNN $\overleftarrow{\Phi}_{\text{enc}}$ reads X from right to left, word by word, and generates a sequence of *backward annotation vectors* $(\overleftarrow{\boldsymbol{h}}_N, \overleftarrow{\boldsymbol{h}}_{N-1}, \cdots, \overleftarrow{\boldsymbol{h}}_1)$. The final annotation vector is the concatenation of forward and backward vectors $\boldsymbol{h}_i = \left[\overrightarrow{\boldsymbol{h}_i}; \overleftarrow{\boldsymbol{h}_i}\right]$, and $C = (\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_N)$ is the set of source annotation vectors.

These annotation vectors are in turn used by the decoder, which is essentially a neural language model (LM) (Bengio et al., 2003) conditioned on the previously emitted words and the source sentence via an attention mechanism. A multilayer perceptron is used to initialise the decoder's hidden state $\boldsymbol{s}_0$ at time step $t = 0$, where the input to this network is the concatenation of the last forward and backward vectors $\left[\overrightarrow{\boldsymbol{h}_N}; \overleftarrow{\boldsymbol{h}_1}\right]$.

At each time step t of the decoder, a *time-dependent* source context vector $\boldsymbol{c}_t$ is computed based on the annotation vectors C and the decoder previous hidden state $\boldsymbol{s}_{t-1}$. This is part of the formulation of the *conditional GRU* and is described further in §2.2. In other words, the encoder is a bi-directional RNN with GRU and the decoder is an RNN with a conditional GRU.

Given a hidden state $\boldsymbol{s}_t$, the probabilities for the next target word are computed using one projection layer followed by a softmax layer as illustrated in eq. (1), where the matrices $\boldsymbol{L}_o$, $\boldsymbol{L}_s$, $\boldsymbol{L}_w$ and $\boldsymbol{L}_c$ are transformation matrices and $\boldsymbol{c}_t$ is a time-dependent source context vector generated by the conditional GRU.

2.2 Conditional GRU

The conditional GRU[1], illustrated in Figure 1, has three main components computed at each time step t of the decoder:

- REC_1 computes a hidden state proposal $\boldsymbol{s}'_t$ based on the previous hidden state $\boldsymbol{s}_{t-1}$ and the previously emitted word $\hat{y}_{t-1}$;
- ATT_{src}[2] is an attention mechanism over the hidden states of the source-language RNN and computes $\boldsymbol{c}_t$ using all source annotation vectors C and the hidden state proposal $\boldsymbol{s}'_t$;
- REC_2 computes the final hidden state $\boldsymbol{s}_t$ using the hidden state proposal $\boldsymbol{s}'_t$ and the time-dependent source context vector $\boldsymbol{c}_t$.

First, a single-layer feed-forward network is used to compute an *expected alignment* $e^{\text{src}}_{t,i}$ between each source annotation vector $\boldsymbol{h}_i$ and the target word $\hat{y}_t$ to be emitted at the current time step t, as shown in Equations (2) and (3):

$$e^{\text{src}}_{t,i} = (\boldsymbol{v}^{\text{src}}_a)^T \tanh(\boldsymbol{U}^{\text{src}}_a \boldsymbol{s}'_t + \boldsymbol{W}^{\text{src}}_a \boldsymbol{h}_i), \quad (2)$$

$$\alpha^{\text{src}}_{t,i} = \frac{\exp\left(e^{\text{src}}_{t,i}\right)}{\sum_{j=1}^{N} \exp\left(e^{\text{src}}_{t,j}\right)}, \quad (3)$$

where $\alpha^{\text{src}}_{t,i}$ is the normalised alignment matrix between each source annotation vector $\boldsymbol{h}_i$ and the word $\hat{y}_t$ to be emitted at time step t, and $\boldsymbol{v}^{\text{src}}_a$, $\boldsymbol{U}^{\text{src}}_a$ and $\boldsymbol{W}^{\text{src}}_a$ are model parameters.

Finally, a time-dependent source context vector $\boldsymbol{c}_t$ is computed as a weighted sum over the source annotation vectors, where each vector is weighted by the attention weight $\alpha^{\text{src}}_{t,i}$, as in eq. (4):

$$\boldsymbol{c}_t = \sum_{i=1}^{N} \alpha^{\text{src}}_{t,i} \boldsymbol{h}_i. \quad (4)$$

[1] https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf.

[2] ATT_{src} is named ATT in the original technical report.

$$p(y_t = k \mid \boldsymbol{y}_{<t}, \boldsymbol{c}_t) \propto \exp(\boldsymbol{L_o} \tanh(\boldsymbol{L_s} \boldsymbol{s}_t + \boldsymbol{L_w} \boldsymbol{E_y}[\hat{y}_{t-1}] + \boldsymbol{L_c} \boldsymbol{c}_t)). \quad (1)$$

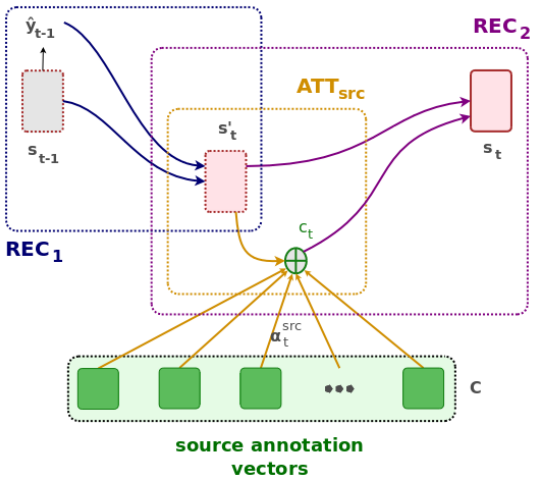


Figure 1: An illustration of the conditional GRU: the steps taken to compute the current hidden state $\boldsymbol{s}_t$ from the previous state $\boldsymbol{s}_{t-1}$, the previously emitted word $\hat{y}_{t-1}$, and the source annotation vectors C, including the candidate hidden state $\boldsymbol{s}'_t$ and the source-language attention vector $\boldsymbol{c}_t$.

3 Multi-modal NMT

Our MNMT model can be seen as an expansion of the attention-based NMT framework described in §2.1 with the addition of a *visual component* to incorporate spatial visual features.

We use publicly available pre-trained CNNs for image feature extraction. Specifically, we extract spatial image features for all images in our dataset using the 50-layer Residual network (ResNet-50) of He et al. (2015). These spatial features are the activations of the `res4f` layer, which can be seen as encoding an image in a 14×14 grid, where each of the entries in the grid is represented by a 1024D feature vector that only encodes information about that specific region of the image. We vectorise this 3-tensor into a 196×1024 matrix $A = (\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_L), \boldsymbol{a}_l \in \mathbb{R}^{1024}$ where each of the L = 196 rows consists of a 1024D feature vector and each column, i.e. feature vector, represents one grid in the image.

3.1 NMT$_{\text{SRC+IMG}}$: decoder with two independent attention mechanisms

Model NMT$_{\text{SRC+IMG}}$ integrates two separate attention mechanisms over the source-language words and visual features in a single decoder RNN. Our doubly-attentive decoder RNN is conditioned on the previous hidden state of the decoder and the previously emitted word, as well as the source sentence and the image via two independent attention mechanisms, as illustrated in Figure 2.

We implement this idea expanding the conditional GRU described in §2.2 onto a *doubly-conditional* GRU. To that end, in addition to the source-language attention, we introduce a new attention mechanism ATT$_{\text{img}}$ to the original conditional GRU proposal. This visual attention computes a *time-dependent* image context vector $\boldsymbol{i}_t$ given a hidden state proposal $\boldsymbol{s}'_t$ and the image annotation vectors $A = (\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_L)$ using the "soft" attention (Xu et al., 2015).

This attention mechanism is very similar to the source-language attention with the addition of a *gating scalar*, explained further below. First, a single-layer feed-forward network is used to compute an *expected alignment* $e^{\text{img}}_{t,l}$ between each image annotation vector $\boldsymbol{a}_l$ and the target word to be emitted at the current time step t, as in eqs. (5) and (6):

$$e^{\text{img}}_{t,l} = (\boldsymbol{v}^{\text{img}}_a)^T \tanh(\boldsymbol{U}^{\text{img}}_a \boldsymbol{s}'_t + \boldsymbol{W}^{\text{img}}_a \boldsymbol{a}_l), \quad (5)$$

$$\alpha^{\text{img}}_{t,l} = \frac{\exp(e^{\text{img}}_{t,l})}{\sum_{j=1}^{L} \exp(e^{\text{img}}_{t,j})}, \quad (6)$$

where $\alpha^{\text{img}}_{t,l}$ is the normalised alignment matrix between all the image patches $\boldsymbol{a}_l$ and the target word to be emitted at time step t, and $\boldsymbol{v}^{\text{img}}_a$, $\boldsymbol{U}^{\text{img}}_a$ and $\boldsymbol{W}^{\text{img}}_a$ are model parameters. Note that Equations (2) and (3), that compute the expected source alignment $e^{\text{src}}_{t,i}$ and the weight matrices $\alpha^{\text{src}}_{t,i}$, and eqs. (5) and (6) that compute the expected image alignment $e^{\text{img}}_{t,l}$ and the weight matrices $\alpha^{\text{img}}_{t,l}$, both compute similar statistics over the source and image annotations, respectively.

In eq. (7) we compute $\beta_t \in [0, 1]$, a gating scalar used to weight the expected importance of the image context vector in relation to the next target word at time step t:

$$\beta_t = \sigma(\boldsymbol{W}_\beta \boldsymbol{s}_{t-1} + \boldsymbol{b}_\beta), \quad (7)$$

where $\boldsymbol{W}_\beta$, $\boldsymbol{b}_\beta$ are model parameters. It is in turn used to compute the time-dependent image context vector $\boldsymbol{i}_t$ for the current decoder time step t, as in eq. (8):

$$\boldsymbol{i}_t = \beta_t \sum_{l=1}^{L} \alpha^{\text{img}}_{t,l} \boldsymbol{a}_l. \quad (8)$$

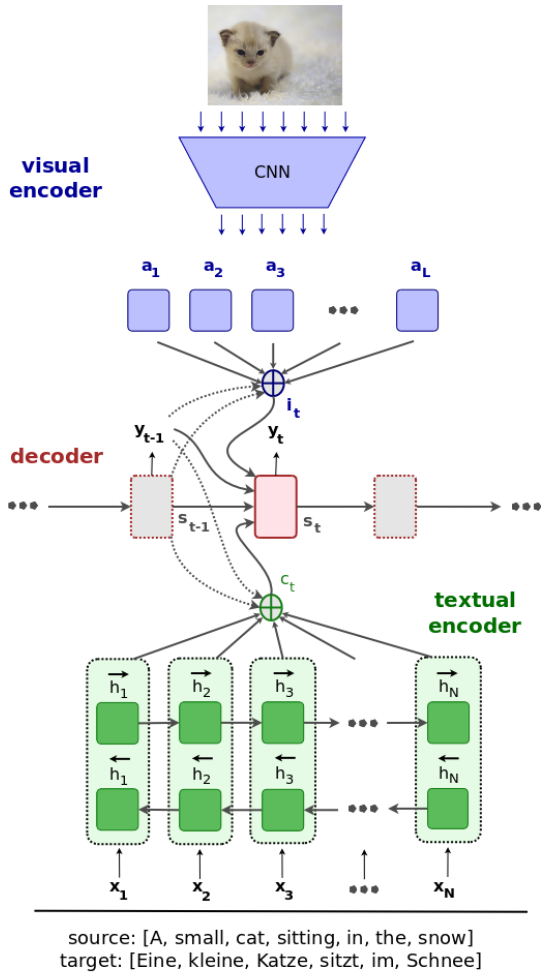


Figure 2: A doubly-attentive decoder learns to attend to image patches and source-language words independently when generating translations.

The only difference between Equations (4) (source context vector) and (8) (image context vector) is that the latter uses a gating scalar, whereas the former does not. We use β following Xu et al. (2015) who empirically found it to improve the variability of the image descriptions generated with their model.

Finally, we use the time-dependent image context vector $\boldsymbol{i}_t$ as an additional input to a modified version of REC$_2$ (§2.2), which now computes the final hidden state $\boldsymbol{s}_t$ using the hidden state proposal $\boldsymbol{s}'_t$, and the time-dependent source and image context vectors $\boldsymbol{c}_t$ and $\boldsymbol{i}_t$, as in eq. (9):

$$\begin{aligned}
\boldsymbol{z}_t &= \sigma(\boldsymbol{W}_z^{\text{src}}\boldsymbol{c}_t + \boldsymbol{W}_z^{\text{img}}\boldsymbol{i}_t + \boldsymbol{U}_z\boldsymbol{s}'_j),\\
\boldsymbol{r}_t &= \sigma(\boldsymbol{W}_r^{\text{src}}\boldsymbol{c}_t + \boldsymbol{W}_r^{\text{img}}\boldsymbol{i}_t + \boldsymbol{U}_r\boldsymbol{s}'_j),\\
\underline{\boldsymbol{s}}_t &= \tanh(\boldsymbol{W}^{\text{src}}\boldsymbol{c}_t + \boldsymbol{W}^{\text{img}}\boldsymbol{i}_t + \boldsymbol{r}_t \odot (\boldsymbol{U}\boldsymbol{s}'_t)),\\
\boldsymbol{s}_t &= (1 - \boldsymbol{z}_t) \odot \underline{\boldsymbol{s}}_t + \boldsymbol{z}_t \odot \boldsymbol{s}'_t. \qquad (9)
\end{aligned}$$

In Equation (10), the probabilities for the next target word are computed using the new multimodal hidden state $\boldsymbol{s}_t$, the previously emitted word $\hat{y}_{t-1}$, and the two context vectors $\boldsymbol{c}_t$ and $\boldsymbol{i}_t$, where $\boldsymbol{L}_o$, $\boldsymbol{L}_s$, $\boldsymbol{L}_w$, $\boldsymbol{L}_{cs}$ and $\boldsymbol{L}_{ci}$ are projection matrices and trained with the model.

4 Data

The Flickr30k data set contains 30k images and 5 descriptions in English for each image (Young et al., 2014). In this work, we use the Multi30k dataset (Elliott et al., 2016), which consists of two multilingual expansions of the original Flickr30k: one with translated data and another one with comparable data, henceforth referred to as M30k$_\text{T}$ and M30k$_\text{C}$, respectively.

For each of the 30k images in the Flickr30k, the M30k$_\text{T}$ has one of the English descriptions manually translated into German by a professional translator. Training, validation and test sets contain 29k, 1,014 and 1k images respectively, each accompanied by a sentence pair (the original English sentence and its translation into German). For each of the 30k images in the Flickr30k, the M30k$_\text{C}$ has five descriptions in German collected independently from the English descriptions. Training, validation and test sets contain 29k, 1,014 and 1k images respectively, each accompanied by five sentences in English and five sentences in German.

We use the entire M30k$_\text{T}$ training set for training our MNMT models, its validation set for model selection with BLEU (Papineni et al., 2002), and its test set for evaluation. In addition, since the amount of training data available is small, we build a back-translation model using the text-only NMT model described in §2.1 trained on the Multi30k$_\text{T}$ data set (German→English and English→German), without images. We use this model to back-translate the 145k German (English) descriptions in the Multi30k$_\text{C}$ into English (German) and include the triples (synthetic English description, German description, image) when translating into German, and the triples (synthetic German description, English description, image) when translating into English, as additional training data (Sennrich et al., 2016a).

We also use the WMT 2015 text-only parallel corpora available for the English–German language pair, consisting of about 4.3M sentence pairs (Bojar et al., 2015). These include the Eu-

$$p(y_t = k \mid \boldsymbol{y}_{<t}, C, A) \propto \exp(\boldsymbol{L_o} \tanh(\boldsymbol{L_s} \boldsymbol{s}_t + \boldsymbol{L_w} \boldsymbol{E_y}[\hat{y}_{t-1}] + \boldsymbol{L_{cs}} \boldsymbol{c}_t + \boldsymbol{L_{ci}} \boldsymbol{i}_t)). \quad (10)$$

roparl v7 (Koehn, 2005), News Commentary and Common Crawl corpora, which are concatenated and used for pre-training.

We use the scripts in the Moses SMT Toolkit (Koehn et al., 2007) to normalise and tokenize English and German descriptions, and we also convert space-separated tokens into subwords (Sennrich et al., 2016b). All models use a common vocabulary of $83,093$ English and $91,141$ German subword tokens. If sentences in English or German are longer than 80 tokens, they are discarded. We train models to translate from English into German, as well as for German into English, and report evaluation of cased, tokenized sentences with punctuation.

5 Experimental setup

Our encoder is a bidirectional RNN with GRU, one 1024D single-layer forward and one 1024D single-layer backward RNN. Source and target word embeddings are 620D each and trained jointly with the model. Word embeddings and other non-recurrent matrices are initialised by sampling from a Gaussian $\mathcal{N}(0, 0.01^2)$, recurrent matrices are random orthogonal and bias vectors are all initialised to zero.

Visual features are obtained by feeding images to the pre-trained ResNet-50 and using the activations of the `res4f` layer (He et al., 2015). We apply dropout with a probability of 0.5 in the encoder bidirectional RNN, the image features, the decoder RNN and before emitting a target word. We follow Gal and Ghahramani (2016) and apply dropout to the encoder bidirectional and the decoder RNN using one same mask in all time steps.

All models are trained using stochastic gradient descent with ADADELTA (Zeiler, 2012) with minibatches of size 80 (text-only NMT) or 40 (MNMT), where each training instance consists of one English sentence, one German sentence and one image (MNMT). We apply early stopping for model selection based on BLEU4, so that if a model does not improve on BLEU4 in the validation set for more than 20 epochs, training is halted.

The translation quality of our models is evaluated quantitatively in terms of BLEU4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), TER (Snover et al., 2006), and chrF3 (Popović, 2015).[3] We report statistical significance with approximate randomisation for the first three metrics with MultEval (Clark et al., 2011).

5.1 Baselines

We train a text-only phrase-based SMT (PBSMT) system and a text-only NMT model for comparison (English→German and German→English). Our PBSMT baseline is built with Moses and uses a 5–gram LM with modified Kneser-Ney smoothing (Kneser and Ney, 1995). It is trained on the English→German (German→English) descriptions of the M30k$_{\mathrm{T}}$, whereas its LM is trained on the German (English) descriptions only. We use minimum error rate training to tune the model with BLEU (Och, 2003). The text-only NMT baseline is the one described in §2.1 and is trained on the M30k$_{\mathrm{T}}$'s English–German descriptions, again in both language directions.

When translating into German, we also compare our model against two publicly available results obtained with multi-modal attention-based NMT models. The first model is Huang et al. (2016)'s best model trained on the same data, and the second is their best model using additional object detections, respectively models `m1` (image at head) and `m3` in the authors' paper.

5.2 Results

In Table 1, we show results for the two text-only baselines NMT and PBSMT, the multi-modal models of Huang et al. (2016), and our MNMT models trained on the M30k$_{\mathrm{T}}$ and pre-trained on the in-domain back-translated M30k$_{\mathrm{C}}$ and the general-domain text-only English-German MT corpora from WMT 2015. All models are trained to translate from English into German.

Training on M30k$_{\mathrm{T}}$ One main finding is that our model consistently outperforms the comparable model of Huang et al. (2016) when translating into German, with improvements of $+1.4$ BLEU and $+2.7$ METEOR. In fact, even when their model has access to more data our model still improves by $+0.9$ METEOR.

Moreover, we can also conclude from Table 1 that PBSMT performs better at recall-oriented

[3] We specifically compute character 6-gram F3, and additionally character precision and recall for comparison.

English→German						
Model	**Training data**	**BLEU4↑**	**METEOR↑**	**TER↓**	**chrF3↑** (prec. / recall)	
NMT	$M30k_T$	33.7	52.3	46.7	65.2	(67.7 / 65.0)
PBSMT	$M30k_T$	32.9	54.3†	45.1†	**67.4**	(66.5 / 67.5)
Huang et al. (2016)	$M30k_T$	35.1 (↑ 1.4)	52.2 (↓ 2.1)	—	—	—
	+ RCNN	**36.5** (↑ 2.8)	54.1 (↓ 0.2)	—	—	—
$NMT_{SRC+IMG}$	$M30k_T$	**36.5**†‡	**55.0**†	**43.7**†‡	67.3	(66.8 / 67.4)
Improvements						
$NMT_{SRC+IMG}$ vs. **NMT**		↑ **2.8**	↑ **2.7**	↓ **3.0**	↑ 2.1	↓ 0.9 / ↑ 2.4
$NMT_{SRC+IMG}$ vs. **PBSMT**		↑ **3.6**	↑ 0.7	↓ **1.4**	↓ 0.1	↑ 0.3 / ↓ 0.1
$NMT_{SRC+IMG}$ vs. Huang		↑ 1.4	↑ 2.8	—	—	—
$NMT_{SRC+IMG}$ vs. Huang (+RCNN)		↑ 0.0	↑ 0.9	—	—	—
Pre-training data set: back-translated $M30k_C$ (in-domain)						
PBSMT (LM)	$M30k_T$	34.0	**55.0**†	44.7	**68.0**	(66.8 / 68.1)
NMT	$M30k_T$	35.5‡	53.4	43.3‡	65.2	(67.7 / 65.0)
$NMT_{SRC+IMG}$	$M30k_T$	**37.1**†‡	54.5†	**42.8**†‡	66.6	(67.2 / 66.5)
$NMT_{SRC+IMG}$ vs. best PBSMT		↑ **3.1**	↓ 0.5	↓ **1.9**	↓ 1.4	↑ 0.4 / ↓ 1.6
$NMT_{SRC+IMG}$ vs. NMT		↑ **1.6**	↑ **1.1**	↓ **0.5**	↑ 1.4	↓ 0.5 / ↑ 1.5
Pre-training data set: WMT'15 English-German corpora (general domain)						
PBSMT (concat)	$M30k_T$	32.6	53.9	46.1	67.3	(66.3 / 67.4)
PBSMT (LM)	$M30k_T$	32.5	54.1	46.0	67.3	(66.0 / 67.4)
NMT	$M30k_T$	37.8	56.7	41.0	69.2	(69.7 / 69.1)
$NMT_{SRC+IMG}$	$M30k_T$	**39.0**†‡	**56.8** ‡	**40.6** ‡	**69.6**	(69.6 / 69.6)
$NMT_{SRC+IMG}$ vs. best PBSMT		↑ **6.4**	↑ **2.7**	↓ **5.4**	↑ 2.3	↑ 3.3 / ↑ 2.2
$NMT_{SRC+IMG}$ vs. NMT		↑ **1.2**	↑ 0.1	↓ 0.4	↑ 0.4	↓ 0.1 / ↑ 0.5

Table 1: BLEU4, METEOR, chrF3, character-level precision and recall (higher is better) and TER scores (lower is better) on the translated Multi30k ($M30k_T$) test set. Best text-only baselines results are underlined and best overall results appear in bold. We show Huang et al. (2016)'s improvements over the best text-only baseline in parentheses. Results are significantly better than the NMT baseline (†) and the SMT baseline (‡) with $p < 0.01$ (no pre-training) or $p < 0.05$ (when pre-training either on the back-translated $M30k_C$ or WMT'15 corpora).

metrics, i.e. METEOR and chrF3, whereas NMT is better at precision-oriented ones, i.e. BLEU4. This is somehow expected, since the attention mechanism in NMT (Bahdanau et al., 2015) does not explicitly take attention weights from previous time steps into account, an thus lacks the notion of source coverage as in SMT (Koehn et al., 2003; Tu et al., 2016). We note that these ideas are complementary and incorporating coverage into model $NMT_{SRC+IMG}$ could lead to more improvements, especially in recall-oriented metrics. Nonetheless, our doubly-attentive model shows consistent gains in both precision- and recall-oriented metrics in comparison to the text-only NMT baseline, i.e. it is significantly better according to BLEU4, METEOR and TER ($p < 0.01$), and it also improves chrF3 by $+2.1$. In comparison to the PBSMT baseline, our proposed model still significantly improves according to both BLEU4 and TER ($p < 0.01$), also increasing METEOR by $+0.7$ but with an associated p-value of $p = 0.071$, therefore not significant for $p < 0.05$. Although chrF3 is the only metric in which the PBSMT model scores best, the difference between our model and the latter is only 0.1, meaning that they are practically equivalent. We note that model $NMT_{SRC+IMG}$ consistently increases character recall in comparison to the text-only NMT baseline. Although it can happen at the expense of character precision, gains in recall are always much higher than any eventual loss in precision, leading to consistent improvements in chrF3.

In Table 2, we observe that when translating into English and training on the original $M30k_T$, model $NMT_{SRC+IMG}$ outperforms both baselines by a large margin, according to all four metrics evaluated. We also note that both model $NMT_{SRC+IMG}$'s character-level precision and re-

German→English				
Model	**BLEU4↑**	**METEOR↑**	**TER↓**	**chrF3↑**
PBSMT	32.8	34.8	43.9	61.8
NMT	38.2	35.8	40.2	62.8
$\text{NMT}_{\text{SRC+IMG}}$	**40.6**†‡	**37.5**†‡	**37.7**†‡	**65.2**
Improvements				
Ours vs. NMT	↑ **2.4**	↑ **1.7**	↓ **2.5**	↑ **2.4**
Ours vs. PBSMT	↑ **7.8**	↑ **2.7**	↓ **6.2**	↑ **3.4**
Pre-training data set: back-translated M30k_{C} (in-domain)				
PBSMT	36.8	36.4	40.8	64.5
NMT	42.6	38.9	36.1	67.6
$\text{NMT}_{\text{SRC+IMG}}$	**43.2**‡	**39.0**‡	**35.5**‡	**67.7**
Improvements				
Ours vs. PBSMT	↑ **6.4**	↑ **2.6**	↓ **5.3**	↑ **3.2**
Ours vs. NMT	↑ 0.6	↑ 0.1	↓ 0.6	↑ 0.1

Table 2: BLEU4, METEOR, chrF3 (higher is better), and TER scores (lower is better) on the translated Multi30k (M30k_{T}) test set. Best text-only baselines results are underlined and best overall results appear in bold. Results are significantly better than the NMT baseline (†) and the SMT baseline (‡) with $p < 0.01$.

call are higher than those of the two baselines, in contrast to results obtained when translating from English into German. This suggests that model $\text{NMT}_{\text{SRC+IMG}}$ might better integrate the image features when translating into an "easier" language, i.e. a language with less morphology, although experiments involving more language pairs are necessary to confirm whether this is indeed the case.

Pre-training We now discuss results for models pre-trained using different data sets. We first pre-trained the two text-only baselines PBSMT and NMT, and our MNMT model on the back-translated M30k_{C}, a medium-sized in-domain image description data set (145k training instances), in both directions. We also pre-trained the same models on the English–German parallel sentences of much larger MT data sets, i.e. the concatenation of the Europarl (Koehn, 2005), Common Crawl and News Commentary corpora, used in WMT 2015 (∼4.3M parallel sentences). Model PBSMT (concat.) used the concatenation of the pre-training and training data for training, and model PBSMT (LM) used the general-domain German sentences as additional data to train the LM. From Tables 1 and 2, it is clear that model $\text{NMT}_{\text{SRC+IMG}}$ can learn from both in-domain, multi-modal pre-training data sets as well as text-only, general domain ones.

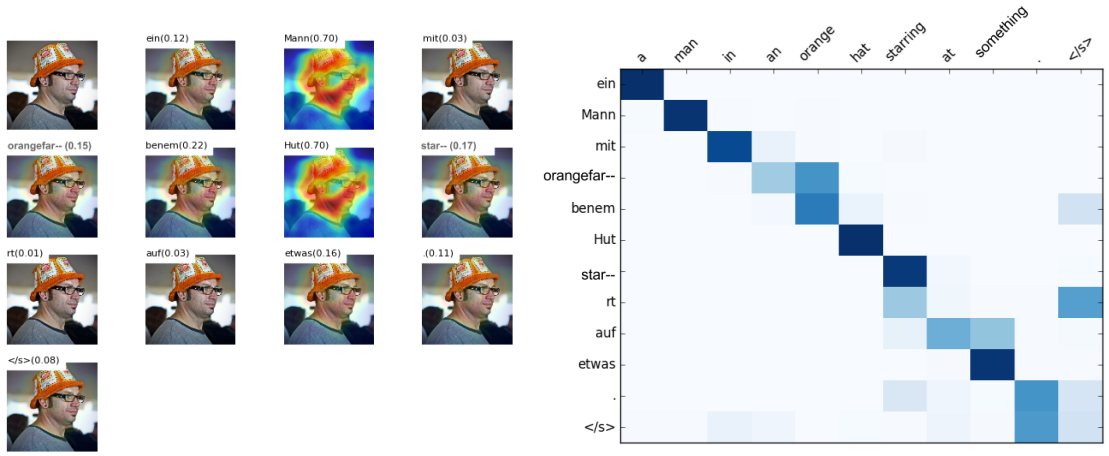
Pre-training on M30k_{C} When pre-training on the back-translated M30k_{C} and translating into German, the recall-oriented chrF3 shows a difference of 1.4 points between PBSMT and our model, mostly due to character recall; nonetheless, our model still improved by the same margin on the text-only NMT baseline. Our model still outperforms the PBSMT baseline according to BLEU4 and TER, and the text-only NMT baseline according to all metrics ($p < .05$).

When translating into English, model $\text{NMT}_{\text{SRC+IMG}}$ still consistently scores higher according to all metrics evaluated, although the differences between its translations and those obtained with the NMT baseline are no longer statistically significant ($p < 0.01$).

Pre-training on WMT 2015 corpora We also pre-trained our English–German models on the WMT 2015 corpora, which took 10 days, i.e. ∼6–7 epochs. Results show that model $\text{NMT}_{\text{SRC+IMG}}$ improves significantly over the NMT baseline according to BLEU4, and is consistently better than the PBSMT baseline according to all four metrics.[4] This is a strong indication that model $\text{NMT}_{\text{SRC+IMG}}$ can exploit the additional pre-training data efficiently, both general- and in-domain. While the PBSMT model is still competitive when using additional in-domain data—according to METEOR and chrF3— the same cannot be said when using general-domain pre-training corpora. From our experiments, NMT models in general, and especially model $\text{NMT}_{\text{SRC+IMG}}$, thrive when training and test domains are mixed, which is a very common real-world scenario.

Textual and visual attention In Figure 3, we visualise the visual and textual attention weights for an entry of the M30k_{T} test set. In the visual attention, the β gate (written in parentheses after each word) caused the image features to be used mostly to generate the words *Mann* (man) and *Hut* (hat), two highly visual terms in the sentence. We observe that in general visually grounded terms, e.g. Mann and Hut, usually have a high associated β value, whereas other less visual terms like *mit* (with) or *auf* (at) do not. That causes the model to use the image features when it is describing a visual concept in the sentence, which is an interest-

[4] In order for PBSMT models to remain competitive, we believe more advanced data selection techniques are needed, which are out of the scope of this work.



(a) Image–target word alignments.

(b) Source–target word alignments.

Figure 3: Visualisation of image– and source–target word alignments for the M30k$_{\text{T}}$ test set.

ing feature of our model. Interestingly, our model is very selective when choosing to use image features: it only assigned $\beta > 0.5$ for 20% of the outputted target words, and $\beta > 0.8$ to only 8%. A manual inspection of translations shows that these words are mostly concrete nouns with a strong visual appeal.

Lastly, using two independent attention mechanisms is a good compromise between model compactness and flexibility. While the attention-based NMT model baseline has ∼200M parameters, model NMT$_{\text{SRC+IMG}}$ has ∼213M, thus using just ∼6.6% more parameters than the latter.

6 Related work

Multi-modal MT was just recently addressed by the MT community by means of a shared task (Specia et al., 2016). However, there has been a considerable amount of work on natural language generation from non-textual inputs. Mao et al. (2014) introduced a multi-modal RNN that integrates text and visual features and applied it to the tasks of image description generation and image–sentence ranking. In their work, the authors incorporate global image features in a separate multi-modal layer that merges the RNN textual representations and the global image features. Vinyals et al. (2015) proposed an influential neural IDG model based on the sequence-to-sequence framework, which is trained end-to-end. Elliott et al. (2015) put forward a model to generate multilingual descriptions of images by learning and transferring features between two independent, non-attentive neural image description models.[5] Venugopalan et al. (2015) introduced a model trained end-to-end to generate textual descriptions of open-domain videos from the video frames based on the sequence-to-sequence framework. Finally, Xu et al. (2015) introduced the first attention-based IDG model where an attentive decoder learns to attend to different parts of an image as it generates its description in natural language.

In the context of NMT, Zoph and Knight (2016) introduced a multi-source attention-based NMT model trained to translate a pair of sentences in two different source languages into a target language, and reported considerable improvements over a single-source baseline. Dong et al. (2015) proposed a multi-task learning approach where a model is trained to translate from one source language into multiple target languages. Firat et al. (2016) put forward a multi-way model trained to translate between many different source and target languages. Instead of one attention mechanism per language pair as in Dong et al. (2015), which would lead to a quadratic number of attention mechanisms in relation to language pairs, they use a shared attention mechanism where each target language has one attention shared by all source languages. Luong et al. (2016) proposed a multi-task approach where they train a model using two tasks and a shared decoder: the main task is to translate from German into English and the sec-

[5] Although their model has not been devised with translation as its primary goal, theirs is one of the baselines of the first shared task in multi-modal MT in WMT 2016 (Specia et al., 2016).

ondary task is to generate English image descriptions. They show improvements in the main translation task when also training for the secondary image description task. Although not an NMT model, Hitschler et al. (2016) recently used image features to re-rank translations of image descriptions generated by an SMT model and reported significant improvements.

Although no purely neural multi-modal model to date significantly improves on both text-only NMT and SMT models (Specia et al., 2016), different research groups have proposed to include global and spatial visual features in re-ranking n-best lists generated by an SMT system or directly in an NMT framework with some success (Caglayan et al., 2016a; Calixto et al., 2016; Huang et al., 2016; Libovický et al., 2016; Shah et al., 2016). To the best of our knowledge, the best published results of a purely MNMT model are those of Huang et al. (2016), who proposed to use global visual features extracted with the VGG19 network (Simonyan and Zisserman, 2015) for an entire image, and also for regions of the image obtained using the RCNN of Girshick et al. (2014). Their best model improves over a strong text-only NMT baseline and is comparable to results obtained with an SMT model trained on the same data. For that reason, their models are used as baselines in our experiments whenever possible.

Our work differs from previous work in that, first, we propose attention-based MNMT models. This is an important difference since the use of attention in NMT has become standard and is the current state-of-the-art (Jean et al., 2015; Luong et al., 2015; Firat et al., 2016; Sennrich et al., 2016b). Second, we propose a *doubly-attentive model* where we effectively fuse two mono-modal attention mechanisms into one multi-modal decoder, training the entire model jointly and end-to-end. Additionally, we are interested in how to merge textual and visual representations into multi-modal representations when generating words in the target language, which differs substantially from text-only translation tasks even when these translate from many source languages and/or into many target languages (Dong et al., 2015; Firat et al., 2016; Zoph and Knight, 2016). To the best of our knowledge, we are among the first[6] to integrate multi-modal inputs in NMT via independent attention mechanisms.

Applications Initial experiments with model $\text{NMT}_{\text{SRC+IMG}}$ have been reported in Calixto et al. (2016). Additionally, $\text{NMT}_{\text{SRC+IMG}}$ has been applied to the machine translation of user-generated product listings from an e-commerce website, while also making use of the product images to improve translations (Calixto et al., 2017b,a).

7 Conclusions and Future Work

We have introduced a novel attention-based, multi-modal NMT model to incorporate spatial visual information into NMT. We have reported state-of-the-art results on the M30k_{T} test set, improving on previous multi-modal attention-based models. We have also showed that our model can be efficiently pre-trained on both medium-sized back-translated in-domain multi-modal data as well as also large general-domain text-only MT corpora, finding that it is able to exploit the additional data regardless of the domain. Our model also compares favourably to both NMT and PB-SMT baselines evaluated on the same training data. In the future, we will incorporate coverage into our model and study how to apply it to other Natural Language Processing tasks.

Acknowledgements

This project has received funding from Science Foundation Ireland in the ADAPT Centre for Digital Content Technology (www.adaptcentre.ie) at Dublin City University funded under the SFI Research Centres Programme (Grant 13/RC/2106) co-funded under the European Regional Development Fund and the European Union Horizon 2020 research and innovation programme under grant agreement 645452 (QT21). The authors would like to thank Chris Hokamp, Peyman Passban, and Dasha Bogdanova for insightful discussions at early stages of this work, Andy Way for proof-reading and providing many good suggestions of improvements, as well as our anonymous reviewers for their valuable comments and feedback.

Reproducibility

Code and pre-trained models for this paper are available at `https://github.com/iacercalixto/nmt_doubly_attentive`.

[6] As pointed out by an anonymous reviewer, Caglayan et al. (2016b) have also experimented with attention-based multi-modal NMT.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations, ICLR 2015*. San Diego, California.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 3:1137–1155. http://dl.acm.org/citation.cfm?id=944919.944966.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal, pages 1–46. http://aclweb.org/anthology/W15-3001.

Ozan Caglayan, Walid Aransa, Yaxing Wang, Marc Masana, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, and Joost van de Weijer. 2016a. Does multimodality help human and machine for translation and image captioning? In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 627–633. http://www.aclweb.org/anthology/W/W16/W16-2358.

Ozan Caglayan, Loïc Barrault, and Fethi Bougares. 2016b. Multimodal Attention for Neural Machine Translation. *CoRR* abs/1609.03976. http://arxiv.org/abs/1609.03976.

Iacer Calixto, Desmond Elliott, and Stella Frank. 2016. DCU-UvA Multimodal MT System Report. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 634–638. http://www.aclweb.org/anthology/W/W16/W16-2359.

Iacer Calixto, Daniel Stein, Evgeny Matusov, Sheila Castilho, and Andy Way. 2017a. Human Evaluation of Multi-modal Neural Machine Translation: A Case-Study on E-Commerce Listing Titles. In *Proceedings of the Sixth Workshop on Vision and Language*. Valencia, Spain, pages 31–37. http://www.aclweb.org/anthology/W17-2004.

Iacer Calixto, Daniel Stein, Evgeny Matusov, Pintu Lohar, Sheila Castilho, and Andy Way. 2017b. Using Images to Improve Machine-Translating E-Commerce Product Listings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain, pages 637–643. http://www.aclweb.org/anthology/E17-2101.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*. page 103.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. Portland, Oregon, HLT '11, pages 176–181. http://dl.acm.org/citation.cfm?id=2002736.2002774.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-Task Learning for Multiple Language Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1723–1732. http://www.aclweb.org/anthology/P15-1166.

Desmond Elliott, Stella Frank, and Eva Hasler. 2015. Multi-Language Image Description with Neural Sequence Models. *CoRR* abs/1510.04709. http://arxiv.org/abs/1510.04709.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual English-German Image Descriptions. In *Proceedings of the 5th Workshop on Vision and Language, VL@ACL 2016*. Berlin, Germany. http://aclweb.org/anthology/W/W16/W16-3210.pdf.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 866–875. http://www.aclweb.org/anthology/N16-1101.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems, NIPS*, Barcelona, Spain,

pages 1019–1027. http://papers.nips.cc/paper/6241-a-theoretically-grounded-application-of-dropout-in-recurrent-neural-networks.pdf.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA, CVPR '14, pages 580–587. https://doi.org/10.1109/CVPR.2014.81.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385* .

Julian Hitschler, Shigehiko Schamoni, and Stefan Riezler. 2016. Multimodal Pivots for Image Caption Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 2399–2409. http://www.aclweb.org/anthology/P16-1227.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based Multimodal Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 639–645. http://www.aclweb.org/anthology/W/W16/W16-2360.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1–10. http://www.aclweb.org/anthology/P15-1001.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*. Seattle, US., pages 1700–1709.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Detroit, Michigan, volume I, pages 181–184.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*. AAMT, AAMT, Phuket, Thailand, pages 79–86. http://mt-archive.info/MTS-2005-Koehn.pdf.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Prague, Czech Republic, ACL '07, pages 177–180. http://dl.acm.org/citation.cfm?id=1557769.1557821.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Edmonton, Canada, NAACL '03, pages 48–54. https://doi.org/10.3115/1073445.1073462.

Jindřich Libovický, Jindřich Helcl, Marek Tlustý, Ondřej Bojar, and Pavel Pecina. 2016. CUNI System for WMT16 Automatic Post-Editing and Multimodal Translation Tasks. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 646–654. http://www.aclweb.org/anthology/W/W16/W16-2361.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-Task Sequence to Sequence Learning. In *Proceedings of the International Conference on Learning Representations (ICLR), 2016*. San Juan, Puerto Rico.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal, pages 1412–1421.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. 2014. Explain Images with Multimodal Recurrent Neural Networks. http://arxiv.org/abs/1410.1090.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. Sapporo, Japan, ACL '03, pages 160–167. https://doi.org/10.3115/1075096.1075117.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, Pennsylvania, ACL '02, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal, pages 392–395. http://aclweb.org/anthology/W15-3049.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving Neural Machine Translation

Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 86–96. http://www.aclweb.org/anthology/P16-1009.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 1715–1725. http://www.aclweb.org/anthology/P16-1162.

Kashif Shah, Josiah Wang, and Lucia Specia. 2016. SHEF-Multimodal: Grounding Machine Translation on Images. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 660–665. http://www.aclweb.org/anthology/W/W16/W16-2363.

K. Simonyan and A. Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*. San Diego, CA.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*. Cambridge, MA, pages 223–231.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A Shared Task on Multimodal Machine Translation and Crosslingual Image Description. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016*. Berlin, Germany, pages 543–553. http://aclweb.org/anthology/W/W16/W16-2346.pdf.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*. Montréal, Canada, pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 76–85. http://www.aclweb.org/anthology/P16-1008.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence - video to text. In *2015 IEEE International Conference on Computer Vision, ICCV 2015*. Santiago, Chile, pages 4534–4542. https://doi.org/10.1109/ICCV.2015.515.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*. Boston, Massachusetts, pages 3156–3164.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. JMLR Workshop and Conference Proceedings, Lille, France, pages 2048–2057. http://jmlr.org/proceedings/papers/v37/xuc15.pdf.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* 2:67–78.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Barret Zoph and Kevin Knight. 2016. Multi-Source Neural Translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 30–34. http://www.aclweb.org/anthology/N16-1004.

A Teacher-Student Framework for Zero-Resource Neural Machine Translation

Yun Chen[†], **Yang Liu**[‡*], **Yong Cheng**[+], **Victor O.K. Li**[†]
[†]Department of Electrical and Electronic Engineering, The University of Hong Kong
[‡]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China
[+]Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
yun.chencreek@gmail.com; liuyang2011@tsinghua.edu.cn;
chengyong3001@gmail.com; vli@eee.hku.hk

Abstract

While end-to-end neural machine translation (NMT) has made remarkable progress recently, it still suffers from the data scarcity problem for low-resource language pairs and domains. In this paper, we propose a method for zero-resource NMT by assuming that parallel sentences have close probabilities of generating a sentence in a third language. Based on the assumption, our method is able to train a source-to-target NMT model ("student") without parallel corpora available guided by an existing pivot-to-target NMT model ("teacher") on a source-pivot parallel corpus. Experimental results show that the proposed method significantly improves over a baseline pivot-based model by +3.0 BLEU points across various language pairs.

1 Introduction

Neural machine translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015), which directly models the translation process in an end-to-end way, has attracted intensive attention from the community. Although NMT has achieved state-of-the-art translation performance on resource-rich language pairs such as English-French and German-English (Luong et al., 2015; Jean et al., 2015; Wu et al., 2016; Johnson et al., 2016), it still suffers from the unavailability of large-scale parallel corpora for translating low-resource languages. Due to the large parameter space, neural models usually learn poorly from low-count events, resulting in a poor choice for low-resource language pairs. Zoph et al. (2016) indicate that NMT obtains much worse translation quality than a statistical machine translation (SMT) system on low-resource languages.

As a result, a number of authors have endeavored to explore methods for translating language pairs without parallel corpora available. These methods can be roughly divided into two broad categories: *multilingual* and *pivot-based*. Firat et al. (2016b) present a multi-way, multilingual model with shared attention to achieve zero-resource translation. They fine-tune the attention part using pseudo bilingual sentences for the zero-resource language pair. Another direction is to develop a universal NMT model in multilingual scenarios (Johnson et al., 2016; Ha et al., 2016). They use parallel corpora of multiple languages to train one single model, which is then able to translate a language pair without parallel corpora available. Although these approaches prove to be effective, the combination of multiple languages in modeling and training leads to increased complexity compared with standard NMT.

Another direction is to achieve source-to-target NMT without parallel data via a *pivot*, which is either text (Cheng et al., 2016a) or image (Nakayama and Nishida, 2016). Cheng et al. (2016a) propose a pivot-based method for zero-resource NMT: it first translates the source language to a pivot language, which is then translated to the target language. Nakayama and Nishida (2016) show that using multimedia information as pivot also benefits zero-resource translation. However, pivot-based approaches usually need to divide the decoding process into two steps, which is not only more computationally expensive, but also potentially suffers from the error propagation problem (Zhu et al., 2013).

In this paper, we propose a new method for zero-resource neural machine translation. Our

*Corresponding author: Yang Liu.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1925–1935

https://doi.org/10.18653/v1/P17-1176

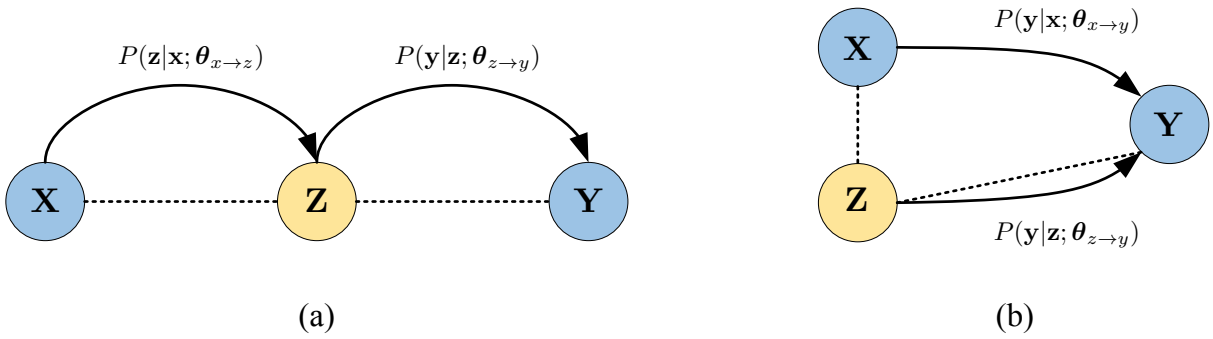


Figure 1: (a) The pivot-based approach and (b) the teacher-student approach to zero-resource neural machine translation. $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ denote source, target, and pivot languages, respectively. We use a dashed line to denote that there is a parallel corpus available for the connected language pair. Solid lines with arrows represent translation directions. The pivot-based approach leverages a pivot to achieve indirect source-to-target translation: it first translates $\mathbf{x}$ into $\mathbf{z}$, which is then translated into $\mathbf{y}$. Our training algorithm is based on the translation equivalence assumption: if $\mathbf{x}$ is a translation of $\mathbf{z}$, then $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y})$ should be close to $P(\mathbf{y}|\mathbf{z};\boldsymbol{\theta}_{z\to y})$. Our approach directly trains the intended source-to-target model $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y})$ ("student") on a source-pivot parallel corpus, with the guidance of an existing pivot-to-target model $P(\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y})$ ("teacher").

method assumes that parallel sentences should have close probabilities of generating a sentence in a third language. To train a source-to-target NMT model without parallel corpora ("student"), we leverage an existing pivot-to-target NMT model ("teacher") to guide the learning process of the student model on a source-pivot parallel corpus. Compared with pivot-based approaches (Cheng et al., 2016a), our method allows direct parameter estimation of the intended NMT model, without the need to divide decoding into two steps. This strategy not only improves efficiency but also avoids error propagation in decoding. Experiments on the Europarl and WMT datasets show that our approach achieves significant improvements in terms of both translation quality and decoding efficiency over a baseline pivot-based approach to zero-resource NMT on Spanish-French and German-French translation tasks.

2 Background

Neural machine translation (Sutskever et al., 2014; Bahdanau et al., 2015) advocates the use of neural networks to model the translation process in an end-to-end manner. As a data-driven approach, NMT treats parallel corpora as the major source for acquiring translation knowledge.

Let $\mathbf{x}$ be a source-language sentence and $\mathbf{y}$ be a target-language sentence. We use $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y})$ to denote a source-to-target neural translation model, where $\boldsymbol{\theta}_{x\to y}$ is a set of model parameters. Given a source-target parallel corpus $D_{x,y}$, which is a set of parallel source-target sentences, the model parameters can be learned by maximizing the log-likelihood of the parallel corpus:

$$\hat{\boldsymbol{\theta}}_{x\to y} = \underset{\boldsymbol{\theta}_{x\to y}}{\operatorname{argmax}} \left\{ \sum_{\langle \mathbf{x},\mathbf{y}\rangle \in D_{x,y}} \log P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y}) \right\}.$$

Given learned model parameters $\hat{\boldsymbol{\theta}}_{x\to y}$, the decision rule for finding the translation with the highest probability for a source sentence $\mathbf{x}$ is given by

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \left\{ P(\mathbf{y}|\mathbf{x};\hat{\boldsymbol{\theta}}_{x\to y}) \right\}. \quad (1)$$

As a data-driven approach, NMT heavily relies on the availability of large-scale parallel corpora to deliver state-of-the-art translation performance (Wu et al., 2016; Johnson et al., 2016). Zoph et al. (2016) report that NMT obtains much lower BLEU scores than SMT if only small-scale parallel corpora are available. Therefore, the heavy dependence on the quantity of training data poses a severe challenge for NMT to translate zero-resource language pairs.

Simple and easy-to-implement, pivot-based methods have been widely used in SMT for

translating zero-resource language pairs (de Gispert and Mariño, 2006; Cohn and Lapata, 2007; Utiyama and Isahara, 2007; Wu and Wang, 2007; Bertoldi et al., 2008; Wu and Wang, 2009; Zahabi et al., 2013; Kholy et al., 2013). As pivot-based methods are agnostic to model structures, they have been adapted to NMT recently (Cheng et al., 2016a; Johnson et al., 2016).

Figure 1(a) illustrates the basic idea of pivot-based approaches to zero-resource NMT (Cheng et al., 2016a). Let $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ denote source, target, and pivot languages. We use dashed lines to denote language pairs with parallel corpora available and solid lines with arrows to denote translation directions.

Intuitively, the source-to-target translation can be indirectly modeled by bridging two NMT models via a pivot:

$$\begin{aligned} & P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to z},\boldsymbol{\theta}_{z\to y}) \\ = & \sum_{\mathbf{z}} P(\mathbf{z}|\mathbf{x};\boldsymbol{\theta}_{x\to z})P(\mathbf{y}|\mathbf{z};\boldsymbol{\theta}_{z\to y}). \quad (2) \end{aligned}$$

As shown in Figure 1(a), pivot-based approaches assume that the source-pivot parallel corpus $D_{x,z}$ and the pivot-target parallel corpus $D_{z,y}$ are available. As it is impractical to enumerate all possible pivot sentences, the two NMT models are trained separately in practice:

$$\hat{\boldsymbol{\theta}}_{x\to z} = \underset{\theta_{x\to z}}{\operatorname{argmax}} \left\{ \sum_{\langle \mathbf{x},\mathbf{z}\rangle \in D_{x,z}} \log P(\mathbf{z}|\mathbf{x};\boldsymbol{\theta}_{x\to z}) \right\},$$

$$\hat{\boldsymbol{\theta}}_{z\to y} = \underset{\theta_{z\to y}}{\operatorname{argmax}} \left\{ \sum_{\langle \mathbf{z},\mathbf{y}\rangle \in D_{z,y}} \log P(\mathbf{y}|\mathbf{z};\boldsymbol{\theta}_{z\to y}) \right\}.$$

Due to the exponential search space of pivot sentences, the decoding process of translating an unseen source sentence $\mathbf{x}$ has to be divided into two steps:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmax}} \left\{ P(\mathbf{z}|\mathbf{x};\hat{\boldsymbol{\theta}}_{x\to z}) \right\}, \quad (3)$$

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \left\{ P(\mathbf{y}|\hat{\mathbf{z}};\hat{\boldsymbol{\theta}}_{z\to y}) \right\}. \quad (4)$$

The above two-step decoding process potentially suffers from the error propagation problem (Zhu et al., 2013): the translation errors made in the first step (i.e., source-to-pivot translation) will affect the second step (i.e., pivot-to-target translation).

Therefore, it is necessary to explore methods to directly model source-to-target translation without parallel corpora available.

3 Approach

3.1 Assumptions

In this work, we propose to directly model the intended source-to-target neural translation based on a teacher-student framework. The basic idea is to use a pre-trained pivot-to-target model ("teacher") to guide the learning process of a source-to-target model ("student") without training data available on a source-pivot parallel corpus. One advantage of our approach is that Equation (1) can be used as the decision rule for decoding, which avoids the error propagation problem faced by two-step decoding in pivot-based approaches.

As shown in Figure 1(b), we still assume that a source-pivot parallel corpus $D_{x,z}$ and a pivot-target parallel corpus $D_{z,y}$ are available. Unlike pivot-based approaches, we first use the pivot-target parallel corpus $D_{z,y}$ to obtain a **teacher model** $P(\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y})$, where $\hat{\boldsymbol{\theta}}_{z\to y}$ is a set of learned model parameters. Then, the teacher model "teaches" the **student model** $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y})$ on the source-pivot parallel corpus $D_{x,z}$ based on the following assumptions.

Assumption 1 *If a source sentence* $\mathbf{x}$ *is a translation of a pivot sentence* $\mathbf{z}$*, then the probability of generating a target sentence* $\mathbf{y}$ *from* $\mathbf{x}$ *should be close to that from its counterpart* $\mathbf{z}$*.*

We can further introduce a word-level assumption:

Assumption 2 *If a source sentence* $\mathbf{x}$ *is a translation of a pivot sentence* $\mathbf{z}$*, then the probability of generating a target word* y *from* $\mathbf{x}$ *should be close to that from its counterpart* $\mathbf{z}$*, given the already obtained partial translation* $\mathbf{y}_{<j}$*.*

The two assumptions are empirically verified in our experiments (see Table 2). In the following subsections, we will introduce two approaches to zero-resource neural machine translation based on the two assumptions.

3.2 Sentence-Level Teaching

Given a source-pivot parallel corpus $D_{x,z}$, our training objective based on Assumption 1 is defined as follows:

$$\begin{aligned} & \mathcal{J}_{\text{SENT}}(\boldsymbol{\theta}_{x\to y}) \\ = & \sum_{\langle \mathbf{x},\mathbf{z}\rangle \in D_{x,z}} \text{KL}\Big(P(\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y}) \Big|\Big| P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y})\Big), \quad (5) \end{aligned}$$

where the KL divergence sums over all possible target sentences:

$$\mathrm{KL}\Big(P(\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y})\Big|\Big|P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y})\Big) = \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y}) \log \frac{P(\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y})}{P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y})}. \quad (6)$$

As the teacher model parameters are fixed, the training objective can be equivalently written as

$$\mathcal{J}_{\mathrm{SENT}}(\boldsymbol{\theta}_{x\to y}) = -\sum_{\langle \mathbf{x},\mathbf{z}\rangle \in D_{x,z}} \mathbb{E}_{\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y}}\Big[\log P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y})\Big]. \quad (7)$$

In training, our goal is to find a set of source-to-target model parameters that minimizes the training objective:

$$\hat{\boldsymbol{\theta}}_{x\to y} = \operatorname*{argmin}_{\boldsymbol{\theta}_{x\to y}} \Big\{\mathcal{J}_{\mathrm{SENT}}(\boldsymbol{\theta}_{x\to y})\Big\}. \quad (8)$$

With learned source-to-target model parameters $\hat{\boldsymbol{\theta}}_{x\to y}$, we use the standard decision rule as shown in Equation (1) to find the translation $\hat{\mathbf{y}}$ for a source sentence $\mathbf{x}$.

However, a major difficulty faced by our approach is the intractability in calculating the gradients because of the exponential search space of target sentences. To address this problem, it is possible to construct a sub-space by either sampling (Shen et al., 2016), generating a k-best list (Cheng et al., 2016b) or mode approximation (Kim and Rush, 2016). Then, standard stochastic gradient descent algorithms can be used to optimize model parameters.

3.3 Word-Level Teaching

Instead of minimizing the KL divergence between the teacher and student models at the sentence level, we further define a training objective at the word level based on Assumption 2:

$$\mathcal{J}_{\mathrm{WORD}}(\boldsymbol{\theta}_{x\to y}) = \sum_{\langle \mathbf{x},\mathbf{z}\rangle \in D_{x,z}} \mathbb{E}_{\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y}}\Big[J(\mathbf{x},\mathbf{y},\mathbf{z},\hat{\boldsymbol{\theta}}_{z\to y},\boldsymbol{\theta}_{x\to y})\Big], \quad (9)$$

where

$$J(\mathbf{x},\mathbf{y},\mathbf{z},\hat{\boldsymbol{\theta}}_{z\to y},\boldsymbol{\theta}_{x\to y}) = \sum_{j=1}^{|\mathbf{y}|} \mathrm{KL}\Big(P(y|\mathbf{z},\mathbf{y}_{<j};\hat{\boldsymbol{\theta}}_{z\to y})\Big|\Big| P(y|\mathbf{x},\mathbf{y}_{<j};\boldsymbol{\theta}_{x\to y})\Big). \quad (10)$$

Equation (9) suggests that the teacher model $P(y|\mathbf{z},\mathbf{y}_{<j};\hat{\boldsymbol{\theta}}_{z\to y})$ "teaches" the student model $P(y|\mathbf{x},\mathbf{y}_{<j};\boldsymbol{\theta}_{x\to y})$ in a word-by-word way. Note that the KL-divergence between two models is defined at the word level:

$$\mathrm{KL}\Big(P(y|\mathbf{z},\mathbf{y}_{<j};\hat{\boldsymbol{\theta}}_{z\to y})\Big|\Big|P(y|\mathbf{x},\mathbf{y}_{<j};\boldsymbol{\theta}_{x\to y})\Big) = \sum_{y\in\mathcal{V}_y} P(y|\mathbf{z},\mathbf{y}_{<j};\hat{\boldsymbol{\theta}}_{z\to y}) \log \frac{P(y|\mathbf{z},\mathbf{y}_{<j};\hat{\boldsymbol{\theta}}_{z\to y})}{P(y|\mathbf{x},\mathbf{y}_{<j};\boldsymbol{\theta}_{x\to y})},$$

where $\mathcal{V}_y$ is the target vocabulary. As the parameters of the teacher model are fixed, the training objective can be equivalently written as:

$$\mathcal{J}_{\mathrm{WORD}}(\boldsymbol{\theta}_{x\to y}) = -\sum_{\langle \mathbf{x},\mathbf{z}\rangle \in D_{x,z}} \mathbb{E}_{\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y}}\Big[S(\mathbf{x},\mathbf{y},\mathbf{z},\hat{\boldsymbol{\theta}}_{z\to y},\boldsymbol{\theta}_{x\to y})\Big], \quad (11)$$

where

$$S(\mathbf{x},\mathbf{y},\mathbf{z},\hat{\boldsymbol{\theta}}_{z\to y},\boldsymbol{\theta}_{x\to y}) = \sum_{j=1}^{|\mathbf{y}|}\sum_{y\in\mathcal{V}_y} P(y|\mathbf{z},\mathbf{y}_{<j};\hat{\boldsymbol{\theta}}_{z\to y}) \times \log P(y|\mathbf{x},\mathbf{y}_{<j};\boldsymbol{\theta}_{x\to y}). \quad (12)$$

Therefore, our goal is to find a set of source-to-target model parameters that minimizes the training objective:

$$\hat{\boldsymbol{\theta}}_{x\to y} = \operatorname*{argmin}_{\boldsymbol{\theta}_{x\to y}} \Big\{\mathcal{J}_{\mathrm{WORD}}(\boldsymbol{\theta}_{x\to y})\Big\}. \quad (13)$$

We use similar approaches as described in Section 3.2 for approximating the full search space with sentence-level teaching. After obtaining $\hat{\boldsymbol{\theta}}_{x\to y}$, the same decision rule as shown in Equation (1) can be utilized to find the most probable target sentence $\hat{\mathbf{y}}$ for a source sentence $\mathbf{x}$.

4 Experiments

4.1 Setup

We evaluate our approach on the Europarl (Koehn, 2005) and WMT corpora. To compare with pivot-based methods, we use the same dataset as (Cheng et al., 2016a). All the sentences are tokenized by the `tokenize.perl` script. All the experiments treat English as the pivot language and French as the target language.

For the Europarl corpus, we evaluate our proposed methods on Spanish-French (Es-Fr) and German-French (De-Fr) translation tasks in a

Corpus	Direction	Train	Dev.	Test
Europarl	Es→ En	850K	2,000	2,000
	De→ En	840K	2,000	2,000
	En→ Fr	900K	2,000	2,000
WMT	Es→ En	6.78M	3,003	3,003
	En→ Fr	9.29M	3,003	3,003

Table 1: Data statistics. For the Europarl corpus, we evaluate our approach on Spanish-French (Es-Fr) and German-French (De-Fr) translation tasks. For the WMT corpus, we evaluate approach on the Spanish-French (Es-Fr) translation task. English is used as a pivot language in all experiments.

zero-resource scenario. To avoid the trilingual corpus constituted by the source-pivot and pivot-target corpora, we split the overlapping pivot sentences of the original source-pivot and pivot-target corpora into two equal parts and merge them separately with the non-overlapping parts for each language pair. The development and test sets are from WMT 2006 shared task.[1] The evaluation metric is case-insensitive BLEU (Papineni et al., 2002) as calculated by the `multi-bleu.perl` script. To deal with out-of-vocabulary words, we adopt byte pair encoding (BPE) (Sennrich et al., 2016) to split words into sub-words. The size of sub-words is set to 30K for each language.

For the WMT corpus, we evaluate our approach on a Spanish-French (Es-Fr) translation task with a zero-resource setting. We combine the following corpora to form the Es-En and En-Fr parallel corpora: Common Crawl, News Commentary, Europarl v7 and UN. All the sentences are tokenized by the `tokenize.perl` script. Newstest2011 serves as the development set and Newstest2012 and Newstest2013 serve as test sets. We use case-sensitive BLEU to evaluate translation results. BPE is also used to reduce the vocabulary size. The size of sub-words is set to 43K, 33K, 43K for Spanish, English and French, respectively. See Table 1 for detailed statistics for the Europarl and WMT corpora.

We leverage an open-source NMT toolkit *dl4mt* implemented by Theano [2] for all the experiments and compare our approach with state-of-the-art multilingual methods (Firat et al., 2016b) and pivot-based methods (Cheng et al., 2016a). Two variations of our framework are used in the experiments:

1. Sentence-Level Teaching: for simplicity, we use the mode as suggested in (Kim and Rush, 2016) to approximate the target sentence space in calculating the expected gradients with respect to the expectation in Equation (7). We run beam search on the pivot sentence with the teacher model and choose the highest-scoring target sentence as the mode. Beam size with $k = 1$ (greedy decoding) and $k = 5$ are investigated in our experiments, denoted as *sent-greedy* and *sent-beam*, respectively.[3]

2. Word-Level Teaching: we use the same mode approximation approach as in sentence-level teaching to approximate the expectation in Equation 12, denoted as *word-greedy* (beam search with $k = 1$) and *word-beam* (beam search with $k = 5$) respectively. Besides, Monte Carlo estimation by sampling from the teacher model is also investigated since it introduces more diverse data, denoted as *word-sampling*.

4.2 Assumptions Verification

To verify the assumptions in Section 3.1, we train a source-to-target translation model $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y})$ and a pivot-to-target translation model $P(\mathbf{y}|\mathbf{z};\boldsymbol{\theta}_{z\to y})$ using the trilingual Europarl corpus. Then, we measure the sentence-level and word-level KL divergence from the source-to-target model $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}_{x\to y})$ at different iterations to the trained pivot-to-target model $P(\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y})$ by caculating $\mathcal{J}_{\text{SENT}}$ (Equation (5)) and $\mathcal{J}_{\text{WORD}}$

[1] http://www.statmt.org/wmt07/shared-task.html

[2] *dl4mt-tutorial*: https://github.com/nyu-dl

[3] We can also adopt sampling and k-best list for approximation. Random sampling brings a large variance (Sutskever et al., 2014; Ranzato et al., 2015; He et al., 2016) for sentence-level teaching. For k-best list, we renormalize the probabilities

$$P(\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y}) \sim \frac{P(\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y})^{\alpha}}{\sum_{\mathbf{y}\in\mathcal{Y}_k} P(\mathbf{y}|\mathbf{z};\hat{\boldsymbol{\theta}}_{z\to y})^{\alpha}},$$

where $\mathcal{Y}_k$ is the k-best list from beam search of the teacher model and α is a hyperparameter controling the sharpness of the distribution (Och, 2003). We set $k = 5$ and $\alpha = 5\times10^{-3}$. The results on test set for Eureparl Corpus are 32.24 BLEU over Spanish-French translation and 24.91 BLEU over German-French translation, which are slightly better than the sent-beam method. However, considering the traing time and the memory consumption, we think mode approximation is already a good way to approximate the target sentence space for sentence-level teaching.

	Approx.	Iterations				
		0	2w	4w	6w	8w
$\mathcal{J}_{\text{SENT}}$	greedy	313.0	73.1	61.5	56.8	55.1
	beam	323.5	73.1	60.7	55.4	54.0
$\mathcal{J}_{\text{WORD}}$	greedy	274.0	51.5	43.1	39.4	38.8
	beam	288.7	52.7	43.3	39.2	38.4
	sampling	268.6	53.8	46.6	42.8	42.4

Table 2: Verification of sentence-level and word-level assumptions by evaluating approximated KL divergence from the source-to-target model to the pivot-to-target model over training iterations of the source-to-target model. The pivot-to-target model is trained and kept fixed.

	Method	Es→ Fr	De→ Fr
Cheng et al. (2016a)	pivot	29.79	23.70
	hard	29.93	23.88
	soft	30.57	23.79
	likelihood	32.59	25.93
Ours	sent-beam	31.64	24.39
	word-sampling	33.86	27.03

Table 3: Comparison with previous work on Spanish-French and German-French translation tasks from the Europarl corpus. English is treated as the pivot language. The likelihood method uses 100K parallel source-target sentences, which are not available for other methods.

(Equation (9)) on 2,000 parallel source-pivot sentences from the development set of WMT 2006 shared task.

Table 2 shows the results. The source-to-target model is randomly initialized at iteration 0. We find that $\mathcal{J}_{\text{SENT}}$ and $\mathcal{J}_{\text{WORD}}$ decrease over time, suggesting that the source-to-target and pivot-to-target models do have small KL divergence at both sentence and word levels.

4.3 Results on the Europarl Corpus

Table 3 gives BLEU scores on the Europarl corpus of our best performing sentence-level method (sent-beam) and word-level method (word-sampling) compared with pivot-based methods (Cheng et al., 2016a). We use the same data preprocessing as (Cheng et al., 2016a). We find that both the sent-beam and word-sampling methods outperform the pivot-based approaches in a zero-resource scenario across language pairs. Our word-sampling method improves over the best performing zero-resource pivot-based method (soft) on Spanish-French translation by +3.29 BLEU points and German-French translation by +3.24 BLEU points. In addition, the word-sampling mothod surprisingly obtains improvement over the likelihood method, which leverages a source-target parallel corpus. The

Method	Es→ Fr		De→ Fr	
	dev	test	dev	test
sent-greedy	31.00	31.05	22.34	21.88
sent-beam	31.57	31.64	24.95	24.39
word-greedy	31.37	31.92	24.72	25.15
word-beam	30.81	31.21	24.64	24.19
word-sampling	33.65	33.86	26.99	27.03

Table 4: Comparison of our proposed methods on Spanish-French and German-French translation tasks from the Europarl corpus. English is treated as the pivot language.

significant improvements can be explained by the error propagation problem of pivot-based methods that translation error of the source-to-pivot translation process is propagated to the pivot-to-target translation process.

Table 4 shows BLEU scores on the Europarl corpus of our proposed methods. For sentence-level approaches, the sent-beam method outperforms the sent-greedy method by +0.59 BLEU points over Spanish-French translation and +2.51 BLEU points over German-French translation on the test set. The results are in line with our observation in Table 2 that sentence-level KL divergence by beam approximation is smaller than that by greedy approximation. However, as the

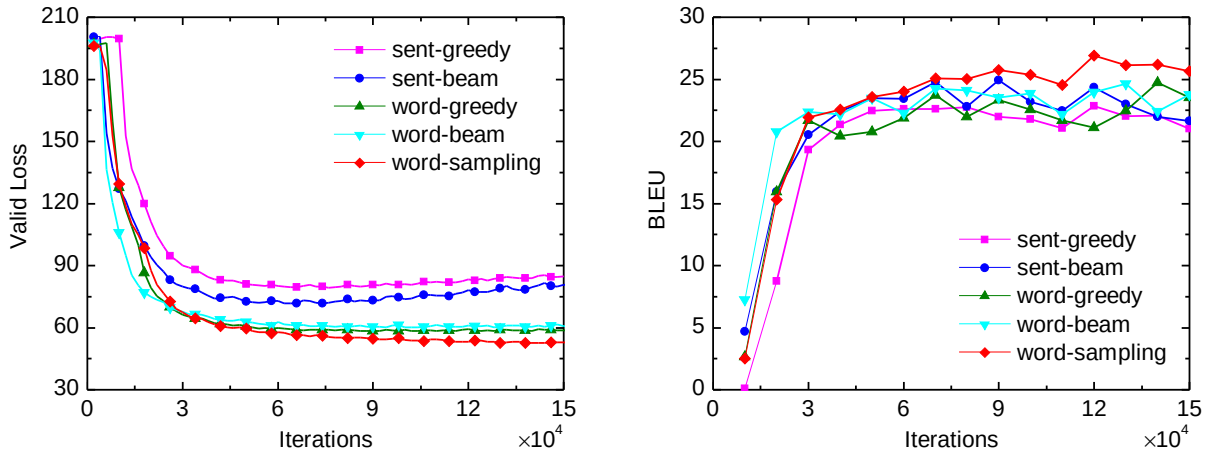


Figure 2: Validation loss and BLEU across iterations of our proposed methods.

	Method	Training			BLEU	
		Es→ En	En→ Fr	Es→ Fr	Newstest2012	Newstest2013
Existing zero-resource NMT systems						
Cheng et al. (2016a)†	pivot	6.78M	9.29M	-	24.60	-
Cheng et al. (2016a)†	likelihood	6.78M	9.29M	100K	25.78	-
Firat et al. (2016b)	one-to-one	34.71M	65.77M	-	17.59	17.61
Firat et al. (2016b)†	many-to-one	34.71M	65.77M	-	21.33	21.19
Our zero-resource NMT system						
	word-sampling	6.78M	9.29M	-	28.06	27.03

Table 5: Comparison with previous work on Spanish-French translation in a zero-resource scenario over the WMT corpus. The BLEU scores are case sensitive. †: the method depends on two-step decoding.

time complexity grows linearly with the number of beams k, the better performance is achieved at the expense of search time.

For word-level experiments, we observe that the word-sampling method performs much better than the other two methods: +1.94 BLEU points on Spanish-French translation and +1.88 BLEU points on German-French translation over the word-greedy method; +2.65 BLEU points on Spanish-French translation and +2.84 BLEU points on German-French translation over the word-beam method. Although Table 2 shows that word-level KL divergence approximated by sampling is larger than that by greedy or beam, sampling approximation introduces more data diversity for training, which dominates the effect of KL divergence difference.

We plot validation loss[4] and BLEU scores over iterations on the German-French translation task in Figure 2. We observe that word-level models tend to have lower validation loss compared with sentence-level methods. Generally, models with lower validation loss tend to have higher BLEU. Our results indicate that this is not necessarily the case: the sent-beam method converges to +0.31 BLEU points on the validation set with +13 validation loss compared with the word-beam method. Kim and Rush (2016) claim a similar observation in data distillation for NMT and provide an explanation that student distributions are more peaked for sentence-level methods. This is indeed the case in our result: on German-French translation task the argmax for the sent-beam student model (on average) approximately accounts for 3.49% of the total probability mass, while the corresponding number is 1.25% for the word-beam student model and 2.60% for the teacher model.

4.4 Results on the WMT Corpus

The word-sampling method obtains the best performance in our five proposed approaches according to experiments on the Europarl corpus. To further verify this approach, we conduct ex-

[4] *Validation loss*: the average negative log-likelihood of sentence pairs on the validation set.

groundtruth	source	Os sentáis al volante en la costa oeste , en San Francisco , y vuestra misión es llegar los primeros a Nueva York .
	pivot	You get in the car on the west coast , in San Francisco , and your task is to be the first one to reach New York .
	target	Vous vous asseyez derrière le volant sur la côte ouest à San Francisco et votre mission est d' arriver le premier à New York .
pivot	pivot	You 'll feel at *the west coast in San Francisco , and your* mission *is to* get *the first* to *New York .* [BLEU: 33.93]
	target	*Vous vous* sentirez comme chez vous *à San Francisco , et votre mission est d'* obtenir *le premier à New York .* [BLEU: 44.52]
likelihood	pivot	You feel at *the west coast , in San Francisco , and your* mission *is to* reach *the first* to *New York .* [BLEU: 47.22]
	target	*Vous vous* sentez à *la côte ouest , à San Francisco , et votre mission est d'* atteindre *le premier à New York .* [BLEU: 49.44]
word-sampling	target	*Vous vous* sentez au *volant sur la côte ouest , à San Francisco et votre mission est d' arriver le premier à New York .* [BLEU: 78.78]

Table 6: Examples and corresponding sentence BLEU scores of translations using the pivot and likelihood methods in (Cheng et al., 2016a) and the proposed word-sampling method. We observe that our approach generates better translations than the methods in (Cheng et al., 2016a). We italicize *correct translation segments* which are no short than 2-grams.

periments on the large scale WMT corpus for Spanish-French translation. Table 5 shows the results of our word-sampling method in comparison with other state-of-the-art baselines. Cheng et al. (2016a) use the same datasets and the same preprocessing as ours. Firat et al. (2016b) utilize a much larger training set.[5] Our method obtains significant improvement over the pivot baseline by +3.46 BLEU points on Newstest2012 and over many-to-one by +5.84 BLEU points on Newstest2013. Note that both methods depend on a source-pivot-target decoding path. Table 6 shows translation examples of the pivot and likelihood methods proposed in (Cheng et al., 2016a) and our proposed word-sampling method. For the pivot and likelihood methods, the Spainish sentence segment 'sentáis al volante' is lost when translated to English. Therefore, both methods miss this information in the translated French sentence. However, the word-sampling method generates 'volant sur', which partially translates 'sentáis al volante', resulting in improved translation quality of target-language sentence.

4.5 Results with Small Source-Pivot Data

The word-sampling method can also be applied to zero-resource NMT with a small source-pivot corpus. Specifically, the size of the source-pivot corpus is orders of magnitude smaller than that of the pivot-target corpus. This setting makes sense in applications. For example, there are significantly fewer Urdu-English corpora available than English-French corpora.

To fulfill this task, we combine our best performing word-sampling method with the initialization and parameter freezing strategy proposed in (Zoph et al., 2016). The Europarl corpus is used in the experiments. We set the size of German-English training data to 100K and use the same teacher model trained with 900K English-French sentences.

Table 7 gives the BLEU score of our method on German-French translation compared with three other methods. Note that our task is much harder than transfer learning (Zoph et al., 2016) since it depends on a parallel German-French corpus. Surprisingly, our method outperforms all other methods. We significantly improve the baseline pivot method by +5.63 BLEU points and the state-of-the-art transfer learning method by +0.56 BLEU points.

Method	Corpus			BLEU
	De-En	De-Fr	En-Fr	
MLE	×	√	×	19.30
transfer	×	√	√	22.39
pivot	√	×	√	17.32
Ours	√	×	√	22.95

Table 7: Comparison on German-French translation task from the Europarl corpus with 100K German-English sentences. English is regarded as the pivot language. Transfer represents the transfer learning method in (Zoph et al., 2016). 100K parallel German-French sentences are used for the MLE and transfer methods.

[5] Their training set does not include the Common Crawl corpus.

5 Related Work

Training NMT models in a zero-resource scenario by leveraging other languages has attracted intensive attention in recent years. Firat et al. (2016b) propose an approach which delivers the multi-way, multilingual NMT model proposed by (Firat et al., 2016a) to zero-resource translation. They use the multi-way NMT model trained by other language pairs to generate a pseudo parallel corpus and fine-tune the attention mechanism of the multi-way NMT model to enable zero-resource translation. Several authors propose a universal encoder-decoder network in multilingual scenarios to perform zero-shot learning (Johnson et al., 2016; Ha et al., 2016). This universal model extracts translation knowledge from multiple different languages, making zero-resource translation feasible without direct training.

Besides multilingual NMT, another important line of research is bridging source and target languages via a pivot language. This idea is widely used in SMT (de Gispert and Mariño, 2006; Cohn and Lapata, 2007; Utiyama and Isahara, 2007; Wu and Wang, 2007; Bertoldi et al., 2008; Wu and Wang, 2009; Zahabi et al., 2013; Kholy et al., 2013). Cheng et al. (2016a) propose pivot-based NMT by simultaneously improving source-to-pivot and pivot-to-target translation quality in order to improve source-to-target translation quality. Nakayama and Nishida (2016) achieve zero-resource machine translation by utilizing image as a pivot and training multimodal encoders to share common semantic representation.

Our work is also related to knowledge distillation, which trains a compact model to approximate the function learned by a larger, more complex model or an ensemble of models (Bucila et al., 2006; Ba and Caurana, 2014; Li et al., 2014; Hinton et al., 2015). Kim and Rush (2016) first introduce knowledge distillation in neural machine translation. They suggest to generate a pseudo corpus to train the student network. Compared with their work, we focus on zero-resource learning instead of model compression.

6 Conclusion

In this paper, we propose a novel framework to train the student model without parallel corpora under the guidance of the pre-trained teacher model on a source-pivot parallel corpus. We introduce sentence-level and word-level teaching to guide the learning process of the student model. Experiments on the Europarl and WMT corpora across languages show that our proposed word-level sampling method can significantly outperforms the state-of-the-art pivot-based methods and multilingual methods in terms of translation quality and decoding efficiency.

We also analyze zero-resource translation with small source-pivot data, and combine our word-level sampling method with initialization and parameter freezing suggested by (Zoph et al., 2016). The experiments on the Europarl corpus show that our approach obtains an significant improvement over the pivot-based baseline.

In the future, we plan to test our approach on more diverse language pairs, e.g., zero-resource Uyghur-English translation using Chinese as a pivot. It is also interesting to extend the teacher-student framework to other cross-lingual NLP applications as our method is transparent to architectures.

Acknowledgments

This work was done while Yun Chen is visiting Tsinghua University. This work is partially supported by the National Natural Science Foundation of China (No.61522204, No. 61331013) and the 863 Program (2015AA015407).

References

Jimmy Ba and Rich Caurana. 2014. Do deep nets really need to be deep? In *NIPS*.

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

Nicola Bertoldi, Madalina Barbaiani, Marcello Federico, and Roldano Cattoni. 2008. Phrase-based statistical machine translation with pivot languages. In *IWSLT*.

Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *KDD*.

Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. 2016a. Neural machine translation with pivot languages. *CoRR* abs/1611.04928.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016b. Semi-supervised learning for neural machine translation .

Trevor Cohn and Mirella Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *ACL*.

Adrià de Gispert and José B. Mariño. 2006. Catalan-english statistical machine translation without parallel corpus: bridging through spanish. In *Proceedings of 5th International Conference on Language Resources and Evaluation (LREC)*. Citeseer, pages 65–68.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *HLT-NAACL*.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman-Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *EMNLP*.

Thanh-Le Ha, Jan Niehues, and Alexander H. Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *CoRR* abs/1611.04798.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *NIPS*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR* abs/1611.04558.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Ahmed El Kholy, Nizar Habash, Gregor Leusch, Evgeny Matusov, and Hassan Sawaf. 2013. Language independent connectivity strength features for phrase pivot statistical machine translation.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Philipp Koehn. 2005. Europarl: a parallel corpus for statistical machine translation.

Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong. 2014. Learning small-size dnn with output-distribution-based criteria. In *INTERSPEECH*.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *ACL*.

Hideki Nakayama and Noriki Nishida. 2016. Zero-resource machine translation by multimodal encoder-decoder network with multimedia pivot. *CoRR* abs/1611.04503.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR* abs/1511.06732.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units .

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation .

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks .

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *HLT-NAACL*.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation* 21:165–181.

Hua Wu and Haifeng Wang. 2009. Revisiting pivot language approach for machine translation. In *ACL/IJCNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144.

Samira Tofighi Zahabi, Somayeh Bakhshaei, and Shahram Khadivi. 2013. Using context vectors in improving a machine translation system with bridge language. In *ACL*.

Xiaoning Zhu, Zhongjun He, Hua Wu, Haifeng Wang, Conghui Zhu, and Tiejun Zhao. 2013. Improving pivot-based statistical machine translation using random walk. In *EMNLP*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *EMNLP*.

Improved Neural Machine Translation with a Syntax-Aware Encoder and Decoder

Huadong Chen[†], **Shujian Huang**[†]*, **David Chiang**[‡], **Jiajun Chen**[†]
[†]State Key Laboratory for Novel Software Technology, Nanjing University
{chenhd,huangsj,chenjj}@nlp.nju.edu.cn
[‡]Department of Computer Science and Engineering, University of Notre Dame
dchiang@nd.edu

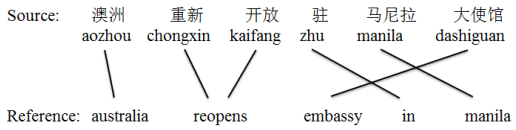
Abstract

Most neural machine translation (NMT) models are based on the sequential encoder-decoder framework, which makes no use of syntactic information. In this paper, we improve this model by explicitly incorporating source-side syntactic trees. More specifically, we propose (1) a *bidirectional tree encoder* which learns both sequential and tree structured representations; (2) a *tree-coverage model* that lets the attention depend on the source-side syntax. Experiments on Chinese-English translation demonstrate that our proposed models outperform the sequential attentional model as well as a stronger baseline with a bottom-up tree encoder and word coverage.[1]

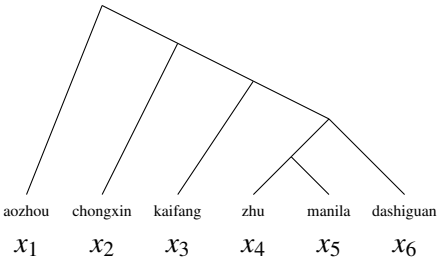
1 Introduction

Recently, neural machine translation (NMT) models (Sutskever et al., 2014; Bahdanau et al., 2015) have obtained state-of-the-art performance on many language pairs. Their success depends on the representation they use to bridge the source and target language sentences. However, this representation, a sequence of fixed-dimensional vectors, differs considerably from most theories about mental representations of sentences, and from traditional natural language processing pipelines, in which semantics is built up compositionally using a recursive syntactic structure.

Perhaps as evidence of this, current NMT models still suffer from syntactic errors such as attachment (Shi et al., 2016). We argue that instead of letting the NMT model rely solely on the implicit structure it learns during training (Cho et al., 2014a), we can improve its performance by augmenting it with explicit structural information and using this information throughout the model. This has two benefits.



(a) example sentence pair with alignments



(b) binarized source side tree

Figure 1: An example sentence pair (a), with its binarized source side tree (b). We use x_i to represent the i-th word in the source sentence. We will use this sentence pairs as the running example throughout this paper.

First, the explicit syntactic information will help the encoder generate better source side representations. Li et al. (2015) show that for tasks in which long-distance semantic dependencies matter, representations learned from recursive models using syntactic structures may be more powerful than those from sequential recurrent models. In the NMT case, given syntactic information, it will be easier for the encoder to incorporate long distance dependencies into better representations, which is especially important for the translation of long sentences.

Second, it becomes possible for the decoder to

* Corresponding author.

[1]Our code is publicly available at https://github.com/howardchenhd/Syntax-awared-NMT/

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1936–1945
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1177

use syntactic information to guide its reordering decisions better (especially for language pairs with significant reordering, like Chinese-English). Although the attention model (Bahdanau et al., 2015) and the coverage model (Tu et al., 2016; Mi et al., 2016) provide effective mechanisms to control the generation of translation, these mechanisms work at the word level and cannot capture phrasal cohesion between the two languages (Fox, 2002; Kim et al., 2017). With explicit syntactic structure, the decoder can generate the translation more in line with the source syntactic structure. For example, when translating the phrase *zhu manila dashiguan* in Figure 1, the tree structure indicates that *zhu* 'in' and *manila* form a syntactic unit, so that the model can avoid breaking this unit up to make an incorrect translation like "in embassy of manila" [2].

In this paper, we propose a novel encoder-decoder model that makes use of a precomputed source-side syntactic tree in both the encoder and decoder. In the encoder (§3.3), we improve the tree encoder of Eriguchi et al. (2016) by introducing a *bidirectional tree encoder*. For each source tree node (including the source words), we generate a representation containing information both from below (as with the original bottom-up encoder) and from above (using a top-down encoder). Thus, the annotation of each node summarizes the surrounding sequential context, as well as the entire syntactic context.

In the decoder (§3.4), we incorporate source syntactic tree structure into the attention model via an extension of the coverage model of Tu et al. (2016). With this *tree-coverage model*, we can better guide the generation phase of translation, for example, to learn a preference for phrasal cohesion (Fox, 2002). Moreover, with a tree encoder, the decoder may try to translate both a parent and a child node, even though they overlap; the tree-coverage model enables the decoder to learn to avoid this problem.

To demonstrate the effectiveness of the proposed model, we carry out experiments on Chinese-English translation. Our experiments show that: (1) our bidirectional tree encoder based NMT system achieves significant improvements over the standard attention-based NMT system, and (2) incorporating source tree structure into the attention model yields a further improvement.

[2] According to the source sentence, "embassy" belongs to "australia", not "manila".

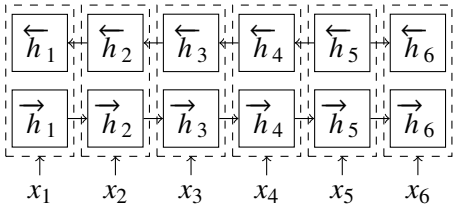


Figure 2: Illustration of the bidirectional sequential encoder. The dashed rectangle represents the annotation of word x_i.

In all, we demonstrate an improvement of +3.54 BLEU over a standard attentional NMT system, and +1.90 BLEU over a stronger NMT system with a Tree-LSTM encoder (Eriguchi et al., 2016) and a coverage model (Tu et al., 2016). To the best of our knowledge, this is the first work that uses source-side syntax in both the encoder and decoder of an NMT system.

2 Neural Machine Translation

Most NMT systems follow the encoder-decoder framework with attention, first proposed by Bahdanau et al. (2015). Given a source sentence $\mathbf{x} = x_1 \cdots x_i \cdots x_I$ and a target sentence $\mathbf{y} = y_1 \cdots y_j \cdots y_J$, NMT aims to directly model the translation probability:

$$P(\mathbf{y} \mid \mathbf{x}; \theta) = \prod_{1}^{J} P(y_j \mid \mathbf{y}_{<\mathbf{j}}, \mathbf{x}; \theta), \qquad (1)$$

where θ is a set of parameters and $y_{<j}$ is the sequence of previously generated target words. Here, we briefly describe the underlying framework of the encoder-decoder NMT system.

2.1 Encoder Model

Following Bahdanau et al. (2015), we use a bidirectional gated recurrent unit (GRU) (Cho et al., 2014b) to encode the source sentence, so that the annotation of each word contains a summary of both the preceding and following words. The bidirectional GRU consists of a forward and a backward GRU, as shown in Figure 2. The forward GRU reads the source sentence from left to right and calculates a sequence of forward hidden states $(\overrightarrow{h_1}, \ldots, \overrightarrow{h_I})$. The backward GRU scans the source sentence from right to left, resulting in a sequence of backward hidden states $(\overleftarrow{h_1}, \ldots, \overleftarrow{h_I})$. Thus

$$\begin{aligned} \overrightarrow{h_i} &= \mathrm{GRU}(\overrightarrow{h_{i-1}}, s_i) \\ \overleftarrow{h_i} &= \mathrm{GRU}(\overleftarrow{h_{i-1}}, s_i) \end{aligned} \qquad (2)$$

where s_i is the i-th source word's word embedding, and GRU is a gated recurrent unit; see the paper by Cho et al. (2014b) for a definition.

The *annotation* of each source word x_i is obtained by concatenating the forward and backward hidden states:

$$\overleftrightarrow{h_i} = \begin{bmatrix} \overrightarrow{h_i} \\ \overleftarrow{h_i} \end{bmatrix}.$$

The whole sequence of these annotations is used by the decoder.

2.2 Decoder Model

The decoder is a forward GRU predicting the translation $\mathbf{y}$ word by word. The probability of generating the j-th word y_j is:

$$P(y_j \mid \mathbf{y}_{<\mathbf{j}}, \mathbf{x}; \theta) = \text{softmax}(t_{j-1}, d_j, c_j) \quad (3)$$

where t_{j-1} is the word embedding of the $(j-1)$-th target word, d_j is the decoder's hidden state of time j, and c_j is the *context vector* at time j. The state d_j is computed as

$$d_j = \text{GRU}(d_{j-1}, t_{j-1}, c_j), \quad (4)$$

where GRU$(\cdot)$ is extended to more than two arguments by first concatenating all arguments except the first.

The attention mechanism computes the context vector c_i as a weighted sum of the source annotations,

$$c_j = \sum_{i=1}^{I} \alpha_{j,i} \overleftrightarrow{h_i} \quad (5)$$

where the attention weight $\alpha_{j,i}$ is

$$\alpha_{j,i} = \frac{\exp(e_{j,i})}{\sum_{i'=1}^{I} \exp(e_{j,i'})} \quad (6)$$

and

$$e_{j,i} = v_a^{\mathrm{T}} \tanh(W_a d_{j-1} + U_a \overleftrightarrow{h_i}) \quad (7)$$

where v_a, W_a and U_a are the weight matrices of the attention model, and $e_{j,i}$ is an attention model that scores how well d_{j-1} and $\overleftrightarrow{h_i}$ match.

With this strategy, the decoder can attend to the source annotations that are most relevant at a given time.

3 Tree Structure Enhanced Neural Machine Translation

Although syntax has shown its effectiveness in non-neural statistical machine translation (SMT) systems (Yamada and Knight, 2001; Koehn et al., 2003; Liu et al., 2006; Chiang, 2007), most proposed NMT models (a notable exception being that of Eriguchi et al. (2016)) process a sentence only as a sequence of words, and do not explicitly exploit the inherent structure of natural language sentences. In this section, we present models which directly incorporate source syntactic trees into the encoder-decoder framework.

3.1 Preliminaries

Like Eriguchi et al. (2016), we currently focus on source side syntactic trees, which can be computed prior to translation. Whereas Eriguchi et al. (2016) use HPSG trees, we use phrase-structure trees as in the Penn Chinese Treebank (Xue et al., 2005). Currently, we are only using the structure information from the tree without the syntactic labels. Thus our approach should be applicable to any syntactic grammar that provides such a tree structure (Figure 1(b)).

More formally, the encoder is given a source sentence $\mathbf{x} = x_1 \cdots x_I$ as well as a source tree whose leaves are labeled $x_1, \ldots, x_I$. We assume that this tree is strictly binary branching. For convenience, each node is assigned an index. The leaf nodes get indices $1, \ldots, I$, which is the same as their word indices. For any node with index k, let $p(k)$ denote the index of the node's parent (if it exists), and $L(k)$ and $R(k)$ denote the indices of the node's left and right children (if they exist).

3.2 Tree-GRU Encoder

We first describe tree encoders (Tai et al., 2015; Eriguchi et al., 2016), and then discuss our improvements.

Following Eriguchi et al. (2016), we build a tree encoder on top of the sequential encoder (as shown in Figure 3(a)). If node k is a leaf node, its hidden state is the annotation produced by the sequential encoder:

$$h_k^{\uparrow} = \overleftrightarrow{h_k}.$$

Thus, the encoder is able to capture both sequential context and syntactic context.

If node k is an interior node, its hidden state is the combination of its previously calculated left

child hidden state $h_{L(k)}$ and right child hidden state $h_{R(k)}$:

$$h_k^{\uparrow} = f(h_{L(k)}^{\uparrow}, h_{R(k)}^{\uparrow}) \tag{8}$$

where $f(\cdot)$ is a nonlinear function, originally a Tree-LSTM (Tai et al., 2015; Eriguchi et al., 2016).

The first improvement we make to the above tree encoder is that, to be consistent with the sequential encoder model, we use Tree-GRU units instead of Tree-LSTM units. Similar to Tree-LSTMs, the Tree-GRU has gating mechanisms to control the information flow inside the unit for every node without separate memory cells. Then, Eq. 8 is calculated by a Tree-GRU as follows:

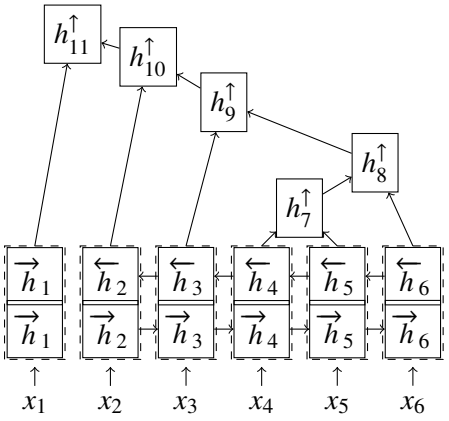
$$\begin{aligned}
r_L &= \sigma(U_L^{(rL)} h_{L(k)}^{\uparrow} + U_R^{(rL)} h_{R(k)}^{\uparrow} + b^{(rL)}) \\
r_R &= \sigma(U_L^{(rR)} h_{L(k)}^{\uparrow} + U_R^{(rR)} h_{R(k)}^{\uparrow} + b^{(rR)}) \\
z_L &= \sigma(U_L^{(zL)} h_{L(k)}^{\uparrow} + U_R^{(zL)} h_{R(k)}^{\uparrow} + b^{(zL)}) \\
z_R &= \sigma(U_L^{(zR)} h_{L(k)}^{\uparrow} + U_R^{(zR)} h_{R(k)}^{\uparrow} + b^{(zR)}) \\
z &= \sigma(U_L^{(z)} h_{L(k)}^{\uparrow} + U_R^{(z)} h_{R(k)}^{\uparrow} + b^{(z)}) \\
\tilde{h}_k^{\uparrow} &= \tanh\left(U_L(r_L \odot h_{L(k)}^{\uparrow}) + U_R(r_R \odot h_{R(k)}^{\uparrow})\right) \\
h_k^{\uparrow} &= z_L \odot h_{L(k)}^{\uparrow} + z_R \odot h_{R(k)}^{\uparrow} + z \odot \tilde{h}_k^{\uparrow}
\end{aligned}$$

where r_L, r_R are the reset gates and z_L, z_R are the update gates for the left and right children, and z is the update gate for the internal hidden state $\tilde{h}_k^{\uparrow}$. The $U^{(\cdot)}$ and $b^{(\cdot)}$ are the weight matrices and bias vectors.

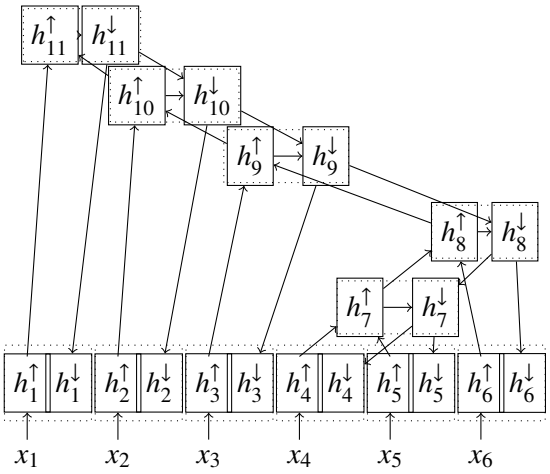
3.3 Bidirectional Tree Encoder

Although the bottom-up tree encoder can take advantage of syntactic structure, the learned representation of a node is based on its subtree only; it contains no information from higher up in the tree. In particular, the representation of leaf nodes is still the sequential one. Thus no syntactic information is fed into words. By analogy with the bidirectional sequential encoder, we propose a natural extension of the bottom-up tree encoder: the bidirectional tree encoder (Figure 3(b)).

Unlike the bottom-up tree encoder or the right-to-left sequential encoder, the top-down encoder by itself would have no lexical information as input. To address this issue, we feed the hidden states of the bottom-up encoder to the top-down encoder. In this way, the information of the whole syntactic tree is handed to the root node and propagated to its offspring by the top-down encoder.



(a) Tree-GRU Encoder



(b) Bidirectional Tree Encoder

Figure 3: Illustration of the proposed encoder models for the running example. The non-leaf nodes are assigned with index 7-11. The annotations $h_i^{\uparrow}$ of leaf nodes in (b) are identical to the annotations (dashed rectangles) of leaf nodes in (a). The dotted rectangles in (b) indicate the annotation produced by the bidirectional tree encoder.

In the top-down encoder, each hidden state has only one predecessor. In fact, the top-down path from root of a tree to any node can be viewed as a sequential recurrent neural network. We can calculate the hidden states of each node top-down using a standard sequential GRU.

First, the hidden state of the root node ρ is simply computed as follows:

$$h_{\rho}^{\downarrow} = \tanh\left(W h_{\rho}^{\uparrow} + b\right) \tag{9}$$

where W and b are a weight matrix and bias vector.

Then, other nodes are calculated by a GRU. For hidden state $h_k^{\downarrow}$:

$$h_k^{\downarrow} = \mathrm{GRU}(h_{p(k)}^{\downarrow}, h_k^{\uparrow}) \tag{10}$$

where $p(k)$ is the parent index of k. We replace the weight matrices W^r, U^r, W^z, U^z, W and U in the standard GRU with P^r_D, Q^r_D, P^z_D, Q^z_D, P_D, and Q_D, respectively. The subscript D is either L or R depending on whether node k is a left or right child, respectively.

Finally, the annotation of each node is obtained by concatenating its bottom-up hidden state and top-down hidden state:

$$h^{\updownarrow}_k = \begin{bmatrix} h^{\uparrow}_k \\ h^{\downarrow}_k \end{bmatrix}.$$

This allows the tree structure information flow from the root to the leaves (words). Thus, all the annotations are based on the full context of word sequence and syntactic tree structure.

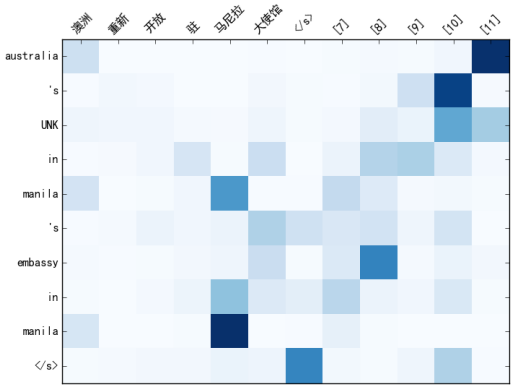
Kokkinos and Potamianos (2017) propose a similar bidirectional Tree-GRU for sentiment analysis, which differs from ours in several respects: in the bottom-up encoder, we use separate reset/update gates for left and right children, analogous to Tree-LSTMs (Tai et al., 2015); in the top-down encoder, we use separate weights for left and right children.

Teng and Zhang (2016) also propose a bidirectional Tree-LSTM encoder for classification tasks. They use a more complex head-lexicalization scheme to feed the top-down encoder. We will compare their model with ours in the experiments.

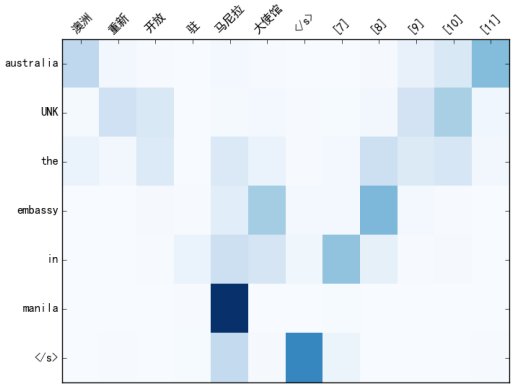
3.4 Tree-Coverage Model

We also extend the decoder to incorporate information about the source syntax into the attention model. We have observed two issues in translations produced using the tree encoder. First, a syntactic phrase in the source sentence is often incorrectly translated into discontinuous words in the output. Second, since the non-leaf node annotations contain more information than the leaf node annotations, the attention model prefers to attend to the non-leaf nodes, which may aggravate the over-translation problem (translating the same part of the sentence more than once).

As shown in Figure 4(a), almost all the non-leaf nodes are attended too many times during decoding. As a result, the Chinese phrase *zhu manila* is translated twice because the model attends to the node spanning *zhu manila* even though both words have already been translated; there is no mechanism to prevent this.



(a) Tree-GRU Encoder



(b) + Tree-Coverage Model

Figure 4: The attention heapmap plotting the attention weights during different translation steps, for translating the sentence in Figure 1(a). The nodes [7]-[11] correspond to non-leaf nodes indexed in Figure 3. Incorporating Tree-Coverage Model produces more concentrated alignments and alleviates the over-translation problem.

Inspired by the approaches of Cohn et al. (2016), Feng et al. (2016), Tu et al. (2016) and Mi et al. (2016), we propose to use prior knowledge to control the attention mechanism. In our case, the prior knowledge is the source syntactic information.

In particular, we build our model on top of the word coverage model proposed by Tu et al. (2016), which alleviate the problems of over-translation and under-translation (failing to translate part of a sentence). The word coverage model makes the attention at a given time step j dependent on the attention at previous time steps via *coverage vectors*:

$$C_{j,i} = \text{GRU}(C_{j-1,i}, \alpha_{j,i}, d_{j-1}, h_i). \quad (11)$$

The coverage vectors are, in turn, used to update the attention at the next time step, by a small modification to the calculation of $e_{j,i}$ in Eq. (7):

$$e_{j,i} = v_a^{\mathrm{T}} \tanh\left(W_a d_{j-1} + U_a h_i + V_a C_{j-1,i}\right). \quad (12)$$

The word coverage model could be interpreted as a control mechanism for the attention model. Like the standard attention model, this coverage model sees the source-sentence annotations as a bag of vectors; it knows nothing about word order, still less about syntactic structure.

For our model, we extend the word coverage model to coverage on the tree structure by adding a coverage vector for each node in the tree. We further incorporate source tree structure information into the calculation of the coverage vector by requiring each node's coverage vector to depend on its children's coverage vectors and attentions at the previous time step:

$$\begin{aligned} C_{j,i} = \mathrm{GRU}(&C_{j-1,i}, \alpha_{j,i}, d_{j-1}, h_i, \\ &C_{j-1,L(i)}, \alpha_{j,L(i)}, \\ &C_{j-1,R(i)}, \alpha_{j,R(i)}). \end{aligned} \quad (13)$$

Although both child and parent nodes of a subtree are helpful for translation, they may supply redundant information. With our mechanism, when the child node is used to produce a translation, the coverage vector of its parent node will reflect this fact, so that the decoder may avoid using the redundant information in the parent node. Figure 4(b) shows a heatmap of the attention of our tree structure enhanced attention model. The attention of non-leaf nodes becomes more concentrated and the over-translation of *zhu manila* is corrected.

4 Experiments

4.1 Data

We conduct experiments on the NIST Chinese-English translation task. The parallel training data consists of 1.6M sentence pairs extracted from LDC corpora,[3] with 46.6M Chinese words and 52.5M English words, respectively. We use NIST MT02 as development data, and NIST MT03–06 as test data. These data are mostly in the same genre (newswire), avoiding the extra consideration of domain adaptation. Table 1 shows the statistics of the data sets. The Chinese side of the corpora is word segmented using ICTCLAS.[4] We parse the Chinese sentences with the Berkeley Parser[5] (Petrov and Klein, 2007) and binarize the resulting trees following Zhang and Clark (2009). The English side of the corpora is lowercased and tokenized.

Data	Usage	Sents.
LDC	train	1.6M
MT02	dev	878
MT03	test	919
MT04	test	1,597
MT05	test	1,082
MT06	test	1,664

Table 1: Experiment data and statistics.

We filter out any translation pairs whose source sentences fail to be parsed. For efficient training, we also filter out the sentence pairs whose source or target lengths are longer than 50. We use a shortlist of the 30,000 most frequent words in each language to train our models, covering approximately 98.2% and 99.5% of the Chinese and English tokens, respectively. All out-of-vocabulary words are mapped to a special symbol `UNK`.

4.2 Model and Training Details

We compare our proposed models with several state-of-the-art NMT systems and techniques:

- **NMT**: the standard attentional NMT model (Bahdanau et al., 2015).
- **Tree-LSTM**: the attentional NMT model extended with the Tree-LSTM encoder (Eriguchi et al., 2016).
- **Coverage**: the attentional NMT model extended with word coverage (Tu et al., 2016).

We used the dl4mt implementation of the attentional model,[6] reimplementing the tree encoder and word coverage models. The word embedding dimension is 512. The hidden layer sizes of both forward and backward sequential encoder are 1024 (except where indicated). Since our Tree-GRU encoders are built on top of the bidirectional sequential encoder, the size of the hidden layer (in each direction) is 2048. For the coverage model, we set the size of coverage vectors to 50.

[3] LDC2002E18, LDC2003E14, the Hansards portion of LDC2004T08, and LDC2005T06.

[4] http://ictclas.nlpir.org

[5] https://github.com/slavpetrov/berkeleyparser

[6] https://github.com/nyu-dl/dl4mt-tutorial

#	Encoder	Coverage	MT02	MT03	MT04	MT05	MT06	Average
1	Sequential	no	33.76	31.88	33.15	30.55	27.47	30.76
2	Tree-LSTM	no	33.83	33.15	33.81	31.22	27.86	31.51(+0.75)
3	Tree-GRU	no	35.39	33.62	35.1	32.55	28.26	32.38(+1.62)
4	Bidirectional	no	35.52	33.91	35.51	33.34	29.91	33.17(+2.41)
5	Sequential	word	34.21	32.73	34.17	31.64	28.29	31.71(+0.95)
6	Tree-LSTM	word	35.81	33.62	34.84	32.6	28.52	32.40(+1.64)
7	Tree-GRU	word	35.91	33.71	35.46	33.02	29.14	32.84(+2.08)
8	Bidirectional	word	36.14	35.00	36.07	33.74	30.40	33.80(+3.04)
9	Tree-LSTM	tree	34.97	33.91	35.21	33.08	29.38	32.90(+2.14)
10	Tree-GRU	tree	35.67	34.25	35.72	33.47	29.95	33.35(+2.59)
11	Bidirectional	tree	**36.57**	**35.64**	**36.63**	**34.35**	**30.57**	**34.30(+3.54)**

Table 2: BLEU scores of different systems. "Sequential", "Tree-LSTM", "Tree-GRU" and "Bidirectional" denote the encoder part for the standard sequential encoder, Tree-LSTM encoder, Tree-GRU encoder and the bidirectional tree encoder, respectively. "no", "word" and "tree" in column "Coverage" represents the decoder part for using no coverage (standard attention), word coverage (Tu et al., 2016) and our proposed tree-coverage model, respectively.

#	System	Coverage	MT02	MT03	MT04	MT05	MT06	Average
12′	Seq-LSTM	no	34.98	32.81	34.08	31.39	28.03	31.58(+0.82)
13′	SeqTree-LSTM	no	35.28	33.56	34.94	32.64	29.26	32.60(+1.84)

Table 3: BLEU scores of different systems based on LSTM. "Seq-LSTM" denotes both the encoder and decoder parts for the sequential model are based on LSTM; "SeqTree-LSTM" means using Tree-LSTM encoder on top of "Seq-LSTM".

We use Adadelta (Zeiler, 2012) for optimization using a mini-batch size of 32. All other settings are the same as in Bahdanau et al. (2015).

We use case insensitive 4-gram BLEU (Papineni et al., 2002) for evaluation, as calculated by `multi-bleu.perl` in the Moses toolkit.[7]

4.3 Tree Encoders

This set of experiments evaluates the effectiveness of our proposed tree encoders. Table 2, row 2 confirms the finding of Eriguchi et al. (2016) that a Tree-LSTM encoder helps, and row 3 shows that our Tree-GRU encoder gets a better result (+0.87 BLEU, v.s. row 2). To verify our assumption that model consistency is important for performance, we also conduct experiments to compare Tree-LSTM and Tree-GRU on top of LSTM-based encoder-decoder settings. Tree-Lstm with LSTM based sequential model can obtain 1.02 BLEU improvement(Table 3, row 13′), while Tree-LSTM with GRU based sequential model only gets 0.75 BLEU improvement. Although Tree-Lstm with LSTM based sequential model obtain a slightly better result(+0.22 BLEU, v.s. Table 2, row 3), it has more parameters(+1.6M) and takes 1.3 times longer for training.

Since the annotation size of our bidirectional tree encoder is twice of the Tree-LSTM encoder, we halved the size of the hidden layers in the sequential encoder to 512 in each direction, to make fair comparison. These results are shown in Table 4. Row 4′ shows that, even with the same annotation size, our bidirectional tree encoder works better than the original Tree-LSTM encoder (row 2). In fact, our halved-sized unidirectional Tree-GRU encoder (row 3′) also works better than the Tree-LSTM encoder (row 2) with half of its annotation size.

We also compared our bidirectional tree encoder with the head-lexicalization based bidirectional tree encoder proposed by Teng and Zhang (2016), which forms the input vector for each non-leaf node by a bottom-up head propagation mechanism (Table 4, row 14′). Our bidirectional tree encoder gives a better result, suggesting that head word information may not be as helpful for machine translation as it is for syntactic parsing.

When we set the hidden size back to 1024, we found that training the bidirectional tree encoder

[7] http://www.statmt.org/moses

#	Encoder	Coverage	MT02	MT03	MT04	MT05	MT06	Average
3′	Tree-GRU	no	34.92	32.79	34.16	32.03	28.75	31.93(+1.17)
4′	Bidirectional	no	35.02	32.64	35.04	32.50	29.72	32.48(+1.72)
14′	Bidirectional-head	no	34.66	33.17	34.78	31.70	28.47	32.03(+1.27)

Table 4: Experiments with 512 hidden units in each direction of the sequential encoder. The bidirectional tree encoder using head-lexicalization (Bidirectional-head), proposed by (Teng and Zhang, 2016), does not work as well as our simpler bidirectional tree encoder (Bidirectional).

was more difficult. Therefore, we adopted a two-phase training strategy: first, we train the parameters of the bottom-up encoder based NMT system; then, with the initialization of bottom-up encoder and random initialization of the top-down part and decoder, we train the bidirectional tree encoder based NMT system. Table 2, row 4 shows the results of this two-phase training: the bidirectional model (row 4) is 0.79 BLEU better than our unidirectional Tree-GRU (row 3).

4.4 Tree-Coverage Model

Rows 5–8 in Table 2 show that the word coverage model of Tu et al. (2016) consistently helps when used with our proposed tree encoders, with the bidirectional tree encoder remaining the best. However, the improvements of the tree encoder models are smaller than that of the baseline system. This may be caused by the fact that the word coverage model neglects the relationship among the trees, e.g. the relationship between children and parent nodes. Our tree-coverage model consistently improves performance further (rows 9–11).

Our best model combines our bidirectional tree encoder with our tree-coverage model (row 11), yielding a net improvement of +3.54 BLEU over the standard attentional model (row 1), and +1.90 BLEU over the stronger baseline that implements both the bottom-up tree encoder and coverage model from previous work (row 6).

As noted before, the original coverage model does not take word order into account. For comparison, we also implement an extension of the coverage model that lets each coverage vector also depend on those of its left and right neighbors at the previous time step. This model does not help; in fact, it reduces BLEU by about 0.2.

4.5 Analysis By Sentence Length

Following Bahdanau et al. (2015), we bin the development and test sentences by length and show BLEU scores for each bin in Figure 5. The proposed bidirectional tree encoder outperforms the sequential NMT system and the Tree-GRU encoder across all lengths. The improvements become larger for sentences longer than 20 words, and the biggest improvement is for sentences longer than 50 words. This provides some evidence for the importance of syntactic information for long sentences.

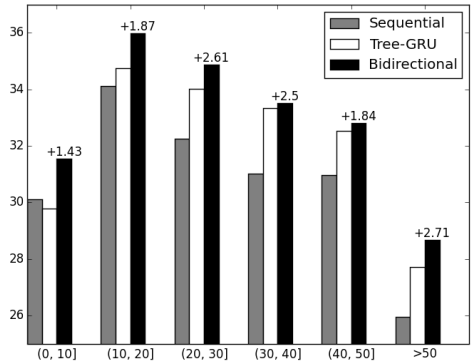


Figure 5: Performance of translations with respect to the lengths of the source sentences. "+" indicates the improvement over the baseline sequential model.

5 Related Work

Recently, many studies have focused on using explicit syntactic tree structure to help learn sentence representations for various sentence classification tasks. For example, Teng and Zhang (2016) and Kokkinos and Potamianos (2017) extend the bottom-up model to a bidirectional model for classification tasks, using Tree-LSTMs with head lexicalization and Tree-GRUs, respectively. We draw on some of these ideas and apply them to machine translation. We use the representation learnt from tree structures to enhance the original sequential model, and make use of these syntactic information during the generation phase.

In NMT systems, the attention model (Bahdanau et al., 2015) becomes a crucial part of the

decoder model. Cohn et al. (2016) and Feng et al. (2016) extend the attentional model to include structural biases from word based alignment models. Kim et al. (2017) incorporate richer structural distributions within deep networks to extend the attention model. Our contribution to the decoder model is to directly exploit structural information in the attention model combined with a coverage mechanism.

6 Conclusion

We have investigated the potential of using explicit source-side syntactic trees in NMT by proposing a novel syntax-aware encoder-decoder model. Our experiments have demonstrated that a top-down encoder is a useful enhancement for the original bottom-up tree encoder (Eriguchi et al., 2016); and incorporating syntactic structure information into the decoder can better control the translation. Our analysis suggests that the benefit of source-side syntax is especially strong for long sentences.

Our current work only uses the structure part of the syntactic tree, without the labels. For future work, it will be interesting to make use of node labels from the tree, or to use syntactic information on the target side, as well.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work is supported by the National Science Foundation of China (No. 61672277, 61300158, 61472183). Part of Huadong Chen's contribution was made when visiting University of Notre Dame. His visit was supported by the joint PhD program of China Scholarship Council.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*. http://arxiv.org/abs/1409.0473.

David Chiang. 2007. Hierarchical phrase-based translation. *Compututational Linguistics* 33(2):201–228. https://doi.org/10.1162/coli.2007.33.2.201.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proc. Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. pages 103–111. http://www.aclweb.org/anthology/W14-4012.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. EMNLP*. pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proc. NAACL HLT*. pages 876–885. http://www.aclweb.org/anthology/N16-1102.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proc. ACL*. pages 823–833. http://www.aclweb.org/anthology/P16-1078.

Shi Feng, Shujie Liu, Nan Yang, Mu Li, Ming Zhou, and Kenny Q. Zhu. 2016. Improving attention modeling with implicit distortion and fertility for machine translation. In *Proc. COLING*. pages 3082–3092. http://aclweb.org/anthology/C16-1290.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proc. EMNLP*. pages 304–3111. https://doi.org/10.3115/1118693.1118732.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *Proc. ICLR*. http://arxiv.org/abs/1702.00887.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. NAACL HLT*. pages 48–54. https://doi.org/10.3115/1073445.1073462.

Filippos Kokkinos and Alexandros Potamianos. 2017. Structural attention neural networks for improved sentiment analysis. In *Proc. EACL*. pages 586–591. http://www.aclweb.org/anthology/E17-2093.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proc. EMNLP*. pages 2304–2314. http://aclweb.org/anthology/D15-1278.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. ACL*. pages 609–616. https://doi.org/10.3115/1220175.1220252.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proc. EMNLP*. pages 955–960. https://aclweb.org/anthology/D16-1096.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL*. pages 311–318. https://doi.org/10.3115/1073083.1073135.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. NAACL HLT*. pages 404–411. http://www.aclweb.org/anthology/N/N07/N07-1051.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proc. EMNLP*. pages 1526–1534. https://aclweb.org/anthology/D16-1159.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. ACL-IJCNLP*. pages 1556–1566. http://www.aclweb.org/anthology/P15-1150.

Zhiyang Teng and Yue Zhang. 2016. Bidirectional tree-structured LSTM with head lexicalization. arXiv:1611.06788. http://arxiv.org/abs/1611.06788.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proc. ACL*. pages 76–85. http://www.aclweb.org/anthology/P16-1008.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.* 11(2):207–238. https://doi.org/10.1017/S135132490400364X.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. ACL*. pages 523–530. https://doi.org/10.3115/1073012.1073079.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the Chinese Treebank using a global discriminative model. In *Proc. IWPT*. pages 162–171. http://www.aclweb.org/anthology/W09-3825.

Cross-lingual Name Tagging and Linking for 282 Languages

Xiaoman Pan[1], **Boliang Zhang**[1], **Jonathan May**[2],
Joel Nothman[3], **Kevin Knight**[2], **Heng Ji**[1]
[1] Computer Science Department, Rensselaer Polytechnic Institute
{panx2,zhangb8,jih}@rpi.edu
[2] Information Sciences Institute, University of Southern California
{jonmay,knight}@isi.edu
[3] Sydney Informatics Hub, University of Sydney
joel.nothman@gmail.com

Abstract

The ambitious goal of this work is to develop a cross-lingual name tagging and linking framework for 282 languages that exist in Wikipedia. Given a document in any of these languages, our framework is able to identify name mentions, assign a coarse-grained or fine-grained type to each mention, and link it to an English Knowledge Base (KB) if it is linkable. We achieve this goal by performing a series of new KB mining methods: generating "silver-standard" annotations by transferring annotations from English to other languages through cross-lingual links and KB properties, refining annotations through self-training and topic selection, deriving language-specific morphology features from anchor links, and mining word translation pairs from cross-lingual links. Both name tagging and linking results for 282 languages are promising on Wikipedia data and on-Wikipedia data. All the data sets, resources and systems for 282 languages are made publicly available as a new benchmark [1].

1 Introduction

Information provided in languages which people can understand saves lives in crises. For example, language barrier was one of the main difficulties faced by humanitarian workers responding to the Ebola crisis in 2014. We propose to break language barriers by extracting information (e.g., entities) from a massive variety of languages and ground the information into an existing knowledge base which is accessible to a user in his/her own language (e.g., a reporter from the World Health Organization who speaks English only).

Wikipedia is a massively multi-lingual resource that currently hosts 295 languages and contains naturally annotated markups [2] and rich informational structures through crowd-sourcing for 35 million articles in 3 billion words. Name mentions in Wikipedia are often labeled as anchor links to their corresponding referent pages. Each entry in Wikipedia is also mapped to external knowledge bases such as DBpedia[3], YAGO (Mahdisoltani et al., 2015) and Freebase (Bollacker et al., 2008) that contain rich properties. Figure 1 shows an example of Wikipedia markups and KB properties. We leverage these markups for developing a language universal framework to automatically extract name mentions from documents in

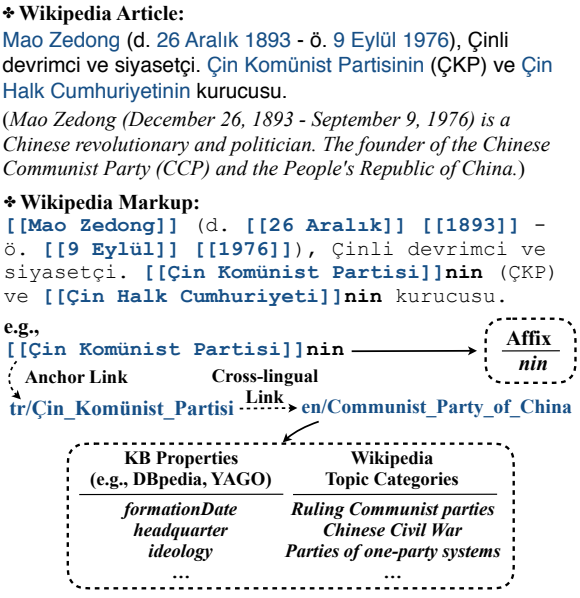


Figure 1: Examples of Wikipedia Markups and KB Properties

[1] http://nlp.cs.rpi.edu/wikiann

[2] https://en.wikipedia.org/wiki/Help:Wiki_markup

[3] http://wiki.dbpedia.org

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1946–1958
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1178

282 languages, and link them to an English KB (Wikipedia in this work). The major challenges and our new solutions are summarized as follows.

Creating "Silver-standard" through cross-lingual entity transfer. The first step is to classify English Wikipedia entries into certain entity types and then propagate these labels to other languages. We exploit the English Abstract Meaning Representation (AMR) corpus (Banarescu et al., 2013) which includes both name tagging and linking annotations for fine-grained entity types to train an automatic classifier. Furthermore, we exploit each entry's properties in DBpedia as features and thus eliminate the need of language-specific features and resources such as part-of-speech tagging as in previous work (Section 2.2).

Refine annotations through self-training. The initial annotations obtained from above are too incomplete and inconsistent. Previous work used name string match to propagate labels. In contrast, we apply self-training to label other mentions without links in Wikipedia articles even if they have different surface forms from the linked mentions (Section 2.4).

Customize annotations through cross-lingual topic transfer. For the first time, we propose to customize name annotations for specific downstream applications. Again, we use a cross-lingual knowledge transfer strategy to leverage the widely available English corpora to choose entities with specific Wikipedia topic categories (Section 2.5).

Derive morphology analysis from Wikipedia markups. Another unique challenge for morphologically rich languages is to segment each token into its stemming form and affixes. Previous methods relied on either high-cost supervised learning (Roth et al., 2008; Mahmoudi et al., 2013; Ahlberg et al., 2015), or low-quality unsupervised learning (Grönroos et al., 2014; Ruokolainen et al., 2016). We exploit Wikipedia markups to automatically learn affixes as language-specific features (Section 2.3).

Mine word translations from cross-lingual links. Name translation is a crucial step to generate candidate entities in cross-lingual entity linking. Only a small percentage of names can be directly translated by matching against cross-lingual Wikipedia title pairs. Based on the observation that Wikipedia titles within any language tend to follow a consistent style and format, we propose an effective method to derive word translation pairs from these titles based on automatic alignment (Section 3.2).

2 Name Tagging

2.1 Overview

Our first step is to generate "silver-standard" name annotations from Wikipedia markups and train a universal name tagger. Figure 2 shows our overall procedure and the following subsections will elaborate each component.

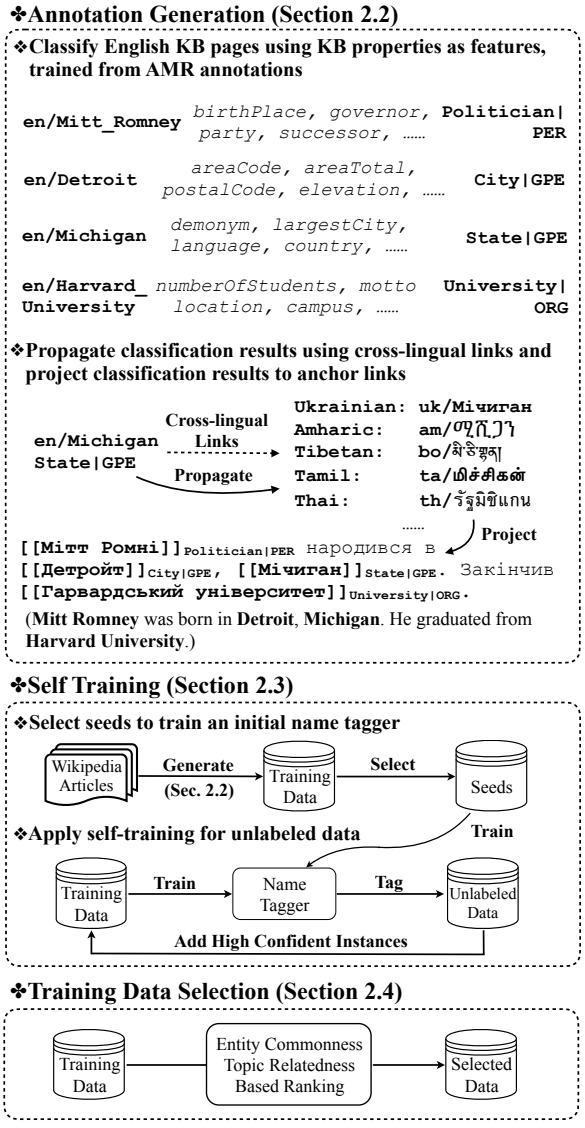


Figure 2: Name Tagging Annotation Generation and Training

2.2 Initial Annotation Generation

We start by assigning an entity type or "other" to each English Wikipedia entry. We utilize the AMR corpus where each entity name mention is manually labeled as one of 139 types

and linked to Wikipedia if it's linkable. In total we obtain 2,756 entity mentions, along with their AMR entity types, Wikipedia titles, YAGO entity types and DBpedia properties. For each pair of AMR entity type t^a and YAGO entity type t^y, we compute the Pointwise Mutual Information (PMI) (Ward Church and Hanks, 1990) of mapping t^a to t^y across all mentions in the AMR corpus. Therefore, each name mention is also assigned a list of YAGO entity types, ranked by their PMI scores with AMR types. In this way, our framework produces three levels of entity typing schemas with different granularity: 4 main types (Person (PER), Organization (ORG), Geo-political Entity (GPE), Location (LOC)), 139 types in AMR, and 9,154 types in YAGO.

Then we leverage an entity's properties in DBpedia as features for assigning types. For example, an entity with a birth date is likely to be a person, while an entity with a population property is likely to be a geo-political entity. Using all DBpedia entity properties as features (60,231 in total), we train Maximum Entropy models to assign types with three levels of granularity to all English Wikipedia pages. In total we obtained 10 million English pages labeled as entities of interest.

Nothman et al. (2013) manually annotated 4,853 English Wikipedia pages with 6 coarse-grained types (Person, Organization, Location, Other, Non-Entity, Disambiguation Page). Using this data set for training and testing, we achieved 96.0% F-score on this initial step, slightly better than their results (94.6% F-score).

Next, we propagate the label of each English Wikipedia page to all entity mentions in all languages in the entire Wikipedia through monolingual redirect links and cross-lingual links.

2.3 Learning Model and KB Derived Features

We use a typical neural network architecture that consists of Bi-directional Long Short-Term Memory and Conditional Random Fields (CRFs) network (Lample et al., 2016) as our underlying learning model for the name tagger for each language. In the following we will describe how we acquire linguistic features.

When a Wikipedia user tries to link an entity mention in a sentence to an existing page, she/he will mark the title (the entity's canonical form, without affixes) within the mention using brackets "`[[]]`", from which we can naturally derive a word's stem and affixes for free. For example, from the Wikipedia markups of the following Turkish sentence: "`Kıta Fransası, güneyde` **`[[Akdeniz]]den`** `kuzeyde [[Manş Denizi]] ve` **`[[Kuzey Denizi]]ne,`** `doğuda` **`[[Ren Nehri]]nden`** `batıda` **`[[Atlas Okyanusu]]na`** `kadar yayılan topraklarda yer alır.` *(Metropolitan France extends from the Mediterranean Sea to the English Channel and the North Sea, and from the Rhine to the Atlantic Ocean.)*", we can learn the following suffixes: "*den*", "*ne*", "*nden*" and "*na*". We use such affix lists to perform basic word stemming, and use them as additional features to determine name boundary and type. For example, "*den*" is a noun suffix which indicates ablative case in Turkish. `[[Akdeniz]]den` means "*from Mediterranean Sea*". Note that this approach can only perform morphology analysis for words whose stem forms and affixes are directly concatenated.

Table 1 summarizes name tagging features.

Features	Descriptions
Form	Lowercase forms of (w_{-1}, w_0, w_{+1})
Case	Case of w_0
Syllable	The first and the last character of w_0
Stem	Stems of (w_{-1}, w_0, w_{+1})
Affix	Affixes of (w_{-1}, w_0, w_{+1})
Gazetteer	Cross-lingual gazetteers learned from training data
Embeddings	Character embeddings and word embeddings [4]learned from training data

Table 1: Name Tagging Features

2.4 Self-Training to Enrich and Refine Labels

The name annotations acquired from the above procedure are far from complete to compete with manually labeled gold-standard data. For example, if a name mention appears multiple times in a Wikipedia article, only the first mention is labeled with an anchor link. We apply self-training to propagate and refine the labels.

We first train an initial name tagger using seeds selected from the labeled data. We adopt an idea from (Guo et al., 2014) which computes Normalized Pointwise Mutual Information (NPMI) (Bouma, 2009) between a tag and a token:

[4]For languages that don't have word segmentation, we consider each character as a token, and use character embeddings only.

$$NPMI(tag, token) = \frac{\ln \frac{p(tag,token)}{p(tag)p(token)}}{-\ln p(tag, token)} \quad (1)$$

Then we select the sentences in which all annotations satisfy $NPMI(tag, token) > \tau$ as seeds [5].

For all Wikipedia articles in a language, we cluster the unlabeled sentences into n clusters [6] by collecting sentences with low cross-entropy into the same cluster. Then we apply the initial tagger to the first unlabeled cluster, select the automatically labeled sentences with high confidence, add them back into the training data, and then re-train the tagger. This procedure is repeated n times until we scan through all unlabeled data.

2.5 Final Training Data Selection for Populous Languages

For some populous languages that have many millions of pages in Wikipedia, we obtain many sentences from self-training. In some emergent settings such as natural disasters it's important to train a system rapidly. Therefore we develop the following effective methods to rank and select high-quality annotated sentences.

Commonness: we prefer sentences that include common entities appearing frequently in Wikipedia. We rank names by their frequency and dynamically set the frequency threshold to select a list of common names. We first initialize the name frequency threshold S to 40. If the number of the sentences is more than a desired size D for training [7], we set the threshold $S = S + 5$, otherwise $S = S - 5$. We iteratively run the selection algorithm until the size of the training set reaches D for a certain S.

Topical Relatedness: Various criteria should be adopted for different scenarios. Our previous work on event extraction (Li et al., 2011) found that by carefully select 1/3 topically related training documents for a test set, we can achieve the same performance as a model trained from the entire training set. Using an emergent disaster setting as a use case, we prefer sentences that include entities related to disaster related topics. We run an English name tagger (Manning et al., 2014) and entity linker (Pan et al., 2015) on the Leidos corpus released by the DARPA LORELEI program [8]. The Leidos corpus consists of documents related to various disaster topics. Based on the linked Wikipedia pages, we rank the frequency of Wikipedia categories and select the top 1% categories (4,035 in total) for our experiments. Some top-ranked topic labels include "*International medical and health organizations*", "*Human rights organizations*", "*International development agencies*", "*Western Asian countries*", "*Southeast African countries*"and "*People in public health*". Then we select the annotated sentences including names (e.g., "*World Health Organization*") in all languages labeled with these topic labels to train the final model.

3 Cross-lingual Entity Linking

3.1 Overview

After we extract names from test documents in a source language, we translate them into English by automatically mined word translation pairs (Section 3.2), and then link translated English mentions to an external English KB (Section 3.3). The overall linking process is illustrated in Figure 3.

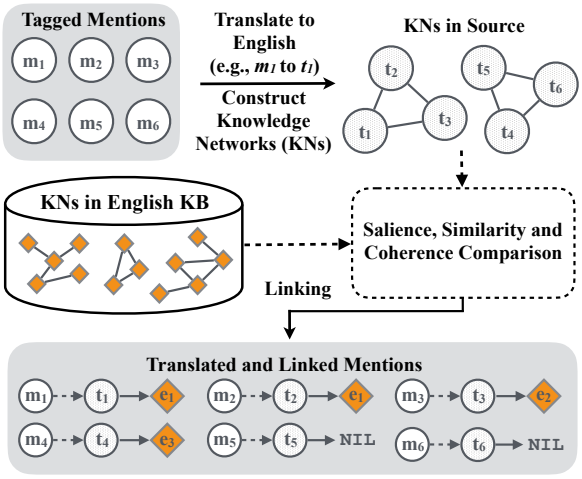


Figure 3: Cross-lingual Entity Linking Overview

3.2 Name Translation

The cross-lingual Wikipedia title pairs, generated through crowd-sourcing, generally follow a consistent style and format in each language. From Table 2 we can see that the order of modifier and head word keeps consistent in Turkish and English titles.

[5] $\tau = 0$ in our experiment.
[6] $n = 20$ in our experiment.
[7] D = 30,000 in our experiment.
[8] http://www.darpa.mil/program/low-resource-languages-for-emergent-incidents

Extracted Cross-lingual Wikipedia Title Pairs	
"*Pekin*"	
Pekin	Beijing
Pekin metrosu	Beijing Subway
Pekin Ulusal Stadyumu	Beijing National Stadium
"*Teknoloji*"	
Nükleer **teknoloji**	Nuclear technology
Teknoloji transferi	Technology transfer
Teknoloji eğitimi	Technology education
"*Enstitüsü*"	
Torchwood **Enstitüsü**	Torchwood Institute
Hudson **Enstitüsü**	Hudson Institute
Smolny **Enstitüsü**	Smolny Institute
"*Pekin Teknoloji*" *[NONE]*	
"*Teknoloji Enstitüsü*"	
Kraliyet **Teknoloji Enstitüsü**	Royal Institute of Technology
Karlsruhe **Teknoloji Enstitüsü**	Karlsruhe Institute of Technology
Georgia **Teknoloji Enstitüsü**	Georgia Institute of Technology
"*Pekin Teknoloji Enstitüsü*" *[NONE]*	

Mined Word Translation Pairs		
Word	Translation	Alignment Confidence
pekin	**Beijing**	*Exact Match*
	beijing	0.5263
	peking	0.3158
teknoloji	**technology**	0.8833
	technological	0.0167
	singularity	0.0167
enstitüsü	**institute**	0.2765
	of	0.2028
	for	0.0221

Table 2: Word Translation Mining from Cross-lingual Wikipedia Title Pairs

For each name mention, we generate all possible combinations of continuous tokens. For example, no Wikipedia titles contain the Turkish name "*Pekin Teknoloji Enstitüsü (Beijing Institute of Technology)*". We generate the following 6 combinations: "*Pekin*", "*Teknoloji*", "*Enstitüsü*", "*Pekin Teknoloji*", "*Teknoloji Enstitüsü*" and "*Pekin Teknoloji Enstitüsü*", and then extract all cross-lingual Wikipedia title pairs containing each combination. Finally we run GIZA++ (Josef Och and Ney, 2003) to extract word for word translations from these title pairs, as shown in Table 2.

3.3 Entity Linking

Given a set of tagged name mentions $M = \{m_1, m_2, ..., m_n\}$, we first obtain their English translations $T = \{t_1, t_2, ..., t_n\}$ using the approach described above. Then we apply an unsupervised collective inference approach to link T to the KB, similar to our previous work (Pan et al., 2015). The only difference is that we construct knowledge networks (KNs) $g(t_i)$ for T based on their co-occurrence within a context window [9] instead of their AMR relations, because AMR parsing is not available for foreign languages. For each translated name mention t_i, an initial list of candidate entities $E(t_i) = \{e_1, e_2, ..., e_k\}$ is generated based on a surface form dictionary mined from KB properties (e.g., *redirects*, *names*, *aliases*). If no surface form can be matched then we determine the mention as unlinkable. Then we construct KNs $g(e_j)$ for each entity candidate e_j in t_i's entity candidate list $E(t_i)$. We compute the similarity between $g(t_i)$ and $g(e_j)$ based on three measures: salience, similarity and coherence, and select the candidate entity with the highest score.

4 Experiments

4.1 Performance on Wikipedia Data

We first conduct an evaluation using Wikipedia data as "silver-standard". For each language, we use 70% of the selected sentences for training and 30% for testing. For entity linking, we don't have ground truth for unlinkable mentions, so we only compute linking accuracy for linkable name mentions. Table 3 presents the overall performance for three coarse-grained entity types: PER, ORG and GPE/LOC, sorted by the number of name mentions. Figure 4 and Figure 5 summarize the performance, with some example languages marked for various ranges of data size.

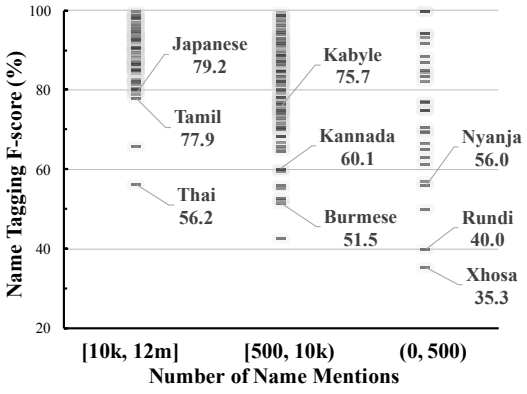


Figure 4: Summary of Name Tagging F-score (%) on Wikipedia Data

Not surprisingly, name tagging performs better for languages with more training mentions. The

[9] In our experiments, we use the previous four and next four name mentions as a context window.

F-score is generally higher than 80% when there are more than 10K mentions, and it significantly drops when there are less than 250 mentions. The languages with low name tagging performance can be categorized into three types: (1) the number of mentions is less than 2K, such as Atlantic-Congo (Wolof), Berber (Kabyle), Chadic (Hausa), Oceanic (Fijian), Hellenic (Greek), Igboid (Igbo), Mande (Bambara), Kartvelian (Georgian, Mingrelian), Timor-Babar (Tetum), Tupian (Guarani) and Iroquoian (Cherokee) language groups; Precision is generally higher than recall for most of these languages, because the small number of linked mentions is not enough to cover a wide variety of entities. (2) there is no space between words, including Chinese, Thai and Japanese; (3) they are not written in latin script, such as the Dravidian group (Tamil, Telugu, Kannada, Malayalam).

The training instances for various entity types are quite imbalanced for some languages. For example, Latin data includes 11% PER names, 84% GPE/LOC names and 5% ORG names. As a result, the performance of ORG is the lowest, while GPE and LOC achieve higher than 75% F-scores for most languages.

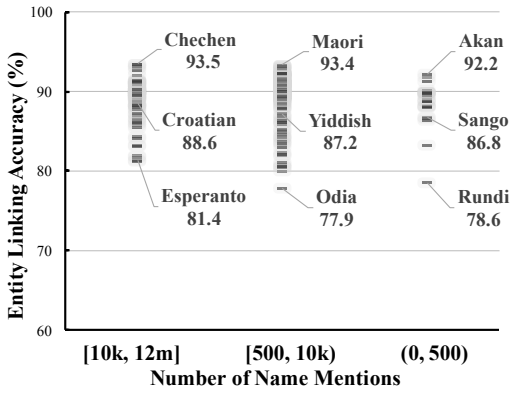


Figure 5: Summary of Entity Linking Accuracy (%) on Wikipedia Data

The linking accuracy is higher than 80% for most languages. Also note that since we don't have perfect annotations on Wikipedia data for any language, these results can be used to estimate how predictable our "silver-standard" data is, but they are not directly comparable to traditional name tagging results measured against gold-standard data annotated by human.

[10]The mapping to language names can be found at http://nlp.cs.rpi.edu/wikiann/mapping

4.2 Performance on Non-Wikipedia Data

In order to have more direct comparison with state-of-the-art name taggers trained from human annotated gold-standard data, we conduct experiments on non-Wikipedia data in 9 languages for which we have human annotated ground truths from the DARPA LORELEI program. Table 4 shows the data statistics. The documents are from news sources and discussion fora.

For fair comparison, we use the same learning method and feature set as described in Section 2.3 to train the models using gold-standard data. Therefore the results of our models trained from gold-standard data are slightly different from some previous work such as (Tsai et al., 2016), mainly due to different learning algorithms and different features sets. For example, the gazetteers we used are different from those in (Tsai et al., 2016), and we did not use brown clusters as additional features.

The name tagging results on LORELEI data set are presented in Table 5. We can see that our approach advances state-of-the-art language-independent methods (Zhang et al., 2016a; Tsai et al., 2016) on the same data sets for most languages, and achieves 6.5% - 17.6% lower F-scores than the models trained from manually annotated gold-standard documents that include thousands of name mentions. To fill in this gap, we would need to exploit more linguistic resources.

Mayfield et al. (2011) constructed a cross-lingual entity linking collection for 21 languages, which covers ground truth for the largest number of languages to date. Therefore we compare our approach with theirs that uses a supervised name transliteration model (McNamee et al., 2011). The entity linking results on non-NIL mentions are presented in Table 6. We can see that except Romanian, our approach outperforms or achieves comparable accuracy as their method on all languages, without using any additional resources or tools such as name transliteration.

4.3 Analysis

Impact of KB-derived Morphological Features

We measured the impact of our affix lists derived from Wikipedia markups on two morphologically-rich languages: Turkish and Uzbek. The morphol-

[11]McNamee et al. (2011) did not develop a model for Chinese even though Chinese data set was included in the collection.

L	*M*	*F*	*A*	*L*	*M*	*F*	*A*	*L*	*M*	*F*	*A*	*L*	*M*	*F*	*A*
en	12M	91.8	84.3	mr	18K	82.4	89.8	szl	3.0K	82.7	92.2	tet	1.2K	73.5	92.2
ja	1.9M	79.2	86.7	bar	17K	97.1	93.1	tk	2.9K	86.3	90.1	sc	1.2K	78.1	91.6
sv	1.8M	93.6	89.7	cv	15K	95.7	93.2	z-c	2.9K	88.2	87.0	wuu	1.2K	79.7	90.8
de	1.7M	89.0	89.8	ba	15K	93.8	92.6	mn	2.9K	76.4	84.4	ksh	1.2K	56.0	83.6
fr	1.4M	93.3	91.2	mg	14K	98.7	90.1	kv	2.9K	89.7	93.2	pfl	1.1K	42.9	80.4
ru	1.4M	90.1	90.0	hi	14K	86.9	88.0	f-v	2.9K	65.4	88.8	haw	1.1K	88.0	84.6
it	1.2M	96.6	90.2	an	14K	93.0	91.1	gan	2.9K	84.9	90.9	am	1.1K	84.7	83.0
sh	1.1M	97.8	90.9	als	14K	85.0	90.9	fur	2.8K	84.5	89.2	bcl	1.1K	82.3	91.7
es	992K	93.9	90.2	sco	14K	86.8	89.6	kw	2.8K	94.0	93.3	nah	1.1K	89.9	89.6
pl	931K	90.0	91.3	bug	13K	99.9	90.0	ilo	2.8K	90.3	91.1	udm	1.1K	88.9	85.0
nl	801K	93.2	91.5	lb	13K	81.5	88.4	mwl	2.7K	76.1	89.4	su	1.1K	72.7	89.2
zh	718K	82.0	90.0	fy	13K	86.6	91.2	mai	2.7K	99.7	90.0	dsb	1.1K	84.7	82.1
pt	576K	90.7	90.3	new	12K	98.2	91.5	nv	2.7K	90.9	91.6	tpi	1.1K	83.3	90.1
uk	472K	91.5	89.4	ga	12K	85.3	91.3	sd	2.7K	65.8	90.9	lo	1.0K	52.8	88.6
cs	380K	94.6	90.5	ht	12K	98.9	93.4	os	2.7K	87.4	89.4	bpy	1.0K	98.3	89.3
sr	365K	95.3	91.2	war	12K	94.9	89.8	mzn	2.6K	86.4	86.9	ki	1.0K	97.5	90.0
hu	357K	95.9	90.4	te	11K	80.5	86.1	azb	2.6K	88.4	90.6	ty	1.0K	86.7	89.8
fi	341K	93.4	90.6	is	11K	80.2	83.2	bxr	2.6K	75.0	90.3	hif	1.0K	81.1	93.1
no	338K	94.1	90.6	pms	10K	98.0	89.5	vec	2.6K	87.9	91.3	ady	979	92.7	91.2
fa	294K	96.4	86.4	zea	10K	86.8	90.3	bo	2.6K	70.4	88.9	ig	968	74.4	91.8
ko	273K	90.6	89.8	sw	9.3K	93.4	90.8	yi	2.6K	76.9	87.2	tyv	903	91.1	91.0
ca	265K	90.3	90.3	ia	8.9K	75.4	90.5	frp	2.5K	86.2	92.3	tn	902	76.9	90.1
tr	223K	96.9	87.3	qu	8.7K	92.5	88.2	myv	2.5K	88.6	92.2	cu	898	75.5	91.3
ro	197K	90.6	89.2	ast	8.3K	89.2	92.0	se	2.5K	90.3	83.5	sm	888	80.0	85.3
bg	186K	65.8	88.4	rm	8.0K	82.0	91.3	cdo	2.5K	91.0	91.9	to	866	92.3	90.7
ar	185K	88.3	89.7	ay	7.9K	88.5	91.0	nso	2.5K	98.9	90.0	tum	831	93.8	92.9
id	150K	87.8	90.0	ps	7.7K	66.9	89.9	gom	2.4K	88.8	90.0	r-r	750	93.0	85.9
he	145K	79.0	91.0	mi	7.5K	95.9	93.4	ky	2.4K	71.8	88.4	om	709	74.2	81.1
eu	137K	82.5	89.2	gag	7.3K	89.3	84.0	n-n	2.3K	92.6	91.6	glk	688	59.5	80.7
da	133K	87.1	85.8	nds	7.0K	84.5	89.8	ne	2.3K	81.5	91.1	lbe	651	88.9	90.8
vi	125K	89.6	82.0	gd	6.7K	92.8	91.3	sa	2.2K	73.9	91.3	bjn	640	64.7	89.5
th	96K	56.2	87.7	mrj	6.7K	97.0	91.6	mt	2.2K	82.3	90.3	srn	619	76.5	89.3
sk	93K	87.3	90.3	so	6.5K	85.8	91.7	my	2.2K	51.5	91.2	mdf	617	82.2	92.4
uz	92K	98.3	90.3	co	6.0K	85.4	89.9	bh	2.2K	92.6	92.5	tw	572	94.6	90.4
eo	85K	88.7	81.4	pnb	6.0K	90.8	86.2	vls	2.2K	78.2	89.1	pih	555	87.2	89.0
la	81K	90.8	89.4	pcd	5.8K	86.1	90.8	ug	2.1K	79.7	92.4	rmy	551	68.5	86.4
z-m	79K	99.3	89.2	wa	5.8K	81.6	82.0	si	2.1K	87.7	90.5	lg	530	98.8	89.3
lt	79K	86.3	87.2	frr	5.7K	70.1	86.3	kaa	2.1K	55.2	89.5	chr	530	70.6	86.2
el	78K	84.6	88.3	scn	5.6K	93.2	89.2	b-s	2.1K	84.5	88.0	ha	517	75.0	87.9
ce	77K	99.4	93.5	fo	5.4K	83.6	92.2	krc	2.1K	84.9	88.9	ab	506	60.0	92.4
ur	77K	96.4	89.3	ckb	5.3K	88.1	89.3	ie	2.1K	88.8	92.8	got	506	91.7	90.1
hr	76K	82.8	88.5	li	5.2K	89.4	91.3	dv	2.0K	76.2	90.5	bi	490	88.5	88.3
ms	75K	86.8	84.1	nap	4.9K	86.9	89.9	xmf	2.0K	73.4	92.2	st	455	84.4	89.8
et	69K	86.8	89.9	crh	4.9K	90.1	89.9	rue	1.9K	82.7	92.2	chy	450	85.1	89.9
kk	68K	88.3	81.8	gu	4.6K	76.0	90.8	pa	1.8K	74.8	84.3	iu	450	66.7	88.9
ceb	68K	96.3	86.6	km	4.6K	52.2	89.9	eml	1.8K	83.5	88.5	zu	449	82.3	89.9
sl	67K	89.5	90.1	tg	4.5K	88.3	90.6	arc	1.8K	68.5	89.2	pnt	445	61.5	89.6
nn	65K	88.1	89.9	hsb	4.5K	91.5	92.0	pdc	1.8K	78.1	91.1	ik	436	94.1	88.2
sim	59K	85.7	90.7	c-z	4.5K	75.0	86.6	kbd	1.7K	74.9	80.6	lrc	416	65.2	86.9
lv	57K	92.1	89.8	jv	4.4K	82.6	87.8	pap	1.7K	88.8	58.4	bm	386	77.3	89.1
tt	53K	87.7	91.4	lez	4.4K	84.2	82.3	jbo	1.7K	92.4	91.6	za	382	57.1	88.2
gl	52K	87.4	88.2	hak	4.3K	85.5	88.1	diq	1.7K	79.3	80.9	mo	373	69.6	88.2
ka	49K	79.8	89.5	ang	4.2K	84.0	92.0	pag	1.7K	91.2	89.5	ss	362	69.2	91.8
vo	47K	98.5	90.8	r-t	4.2K	88.1	89.0	kg	1.6K	82.1	90.1	ee	297	63.2	90.0
lmo	39K	98.3	89.0	kn	4.1K	60.1	91.7	m-b	1.6K	78.3	80.0	dz	262	50.0	90.0
be	38K	84.1	88.3	csb	4.1K	87.0	92.3	rw	1.6K	95.4	91.5	ak	258	86.8	92.2
mk	35K	93.4	83.3	lij	4.1K	72.3	91.9	or	1.6K	86.4	77.9	sg	245	99.9	86.8
cy	32K	90.7	89.3	nov	4.0K	77.0	92.1	ln	1.6K	82.8	91.4	ts	236	93.3	88.9
bs	31K	84.8	89.8	ace	4.0K	81.6	90.3	kl	1.5K	75.0	90.9	rn	185	40.0	78.6
ta	31K	77.9	88.2	gn	4.0K	71.2	89.3	sn	1.5K	95.0	93.3	ve	183	99.9	88.0
hy	28K	90.4	81.3	koi	4.0K	89.6	92.9	av	1.4K	82.0	83.7	ny	169	56.0	90.2
bn	27K	93.8	87.2	mhr	3.9K	86.7	92.4	as	1.4K	89.6	89.3	ff	168	76.9	88.9
az	26K	85.1	86.0	io	3.8K	87.2	92.3	stq	1.4K	70.0	90.6	ch	159	70.6	90.0
sq	26K	94.1	92.1	min	3.8K	85.8	89.9	gv	1.3K	84.8	89.1	xh	141	35.3	89.5
ml	24K	82.4	88.8	arz	3.8K	77.8	89.3	wo	1.3K	87.7	90.0	fj	126	75.0	91.3
br	22K	87.0	85.5	ext	3.7K	77.8	91.6	xal	1.3K	98.7	90.9	ks	124	75.0	83.3
z-y	22K	87.3	88.4	yo	3.7K	94.0	90.8	nrm	1.3K	96.4	92.7	ti	52	94.2	90.0
af	21K	85.7	91.1	sah	3.6K	91.2	93.0	na	1.2K	87.6	88.7	cr	49	91.8	89.8
b-x	20K	85.1	87.7	vep	3.5K	85.8	89.8	ltg	1.2K	74.3	92.1	pi	41	83.3	86.4
tl	19K	92.7	90.3	ku	3.3K	83.2	85.1	pam	1.2K	87.2	91.0				
oc	18K	92.5	90.0	kab	3.3K	75.7	84.3	lad	1.2K	92.3	92.4				

Table 3: Performance on Wikipedia Data (*L*: language ID [10]; *M*: the number of name mentions; *F*: name tagging F-score (%); *A*: entity linking accuracy (%))

Language	*Gold* Training	*Silver* Training	Test
Bengali	8,760	22,093	3,495
Hungarian	3,414	34,022	1,320
Russian	2,751	35,764	1,213
Tamil	7,033	25,521	4,632
Tagalog	4,648	15,839	3,351
Turkish	3,067	37,058	2,172
Uzbek	3,137	64,242	2,056
Vietnamese	2,261	63,971	987
Yoruba	4,061	9,274	3,395

Table 4: # of Names in Non-Wikipedia Data

Language	Training from *Gold*	Training from *Silver*	(Zhang et al., 2016a)	(Tsai et al., 2016)
Bengali	61.6	**44.0**	34.8	43.3
Hungarian	63.9	47.9	-	-
Russian	61.8	49.4	-	-
Tamil	42.2	**35.7**	26.0	29.6
Tagalog	70.7	58.3	51.3	65.4
Turkish	66.0	**51.5**	43.6	47.1
Uzbek	56.0	44.2	-	-
Vietnamese	54.3	44.5	-	-
Yoruba	55.1	**37.6**	36.0	36.7

Table 5: Name Tagging F-score (%) on Non-Wikipedia Data

Language	# of Non-NIL Mentions	(Mayfield et al., 2011)	Our Approach
Arabic	661	70.6	**80.2**
Bulgarian	2,068	82.1	**84.1**
Chinese	956	- [11]	91.0
Croatian	2,257	88.9	**90.8**
Czech	722	77.2	**85.9**
Danish	1,096	93.8	91.2
Dutch	1,087	92.4	89.2
Finnish	1,049	86.8	85.8
French	657	90.4	**92.1**
German	769	85.7	**89.7**
Greek	2,129	71.4	**79.8**
Italian	1,087	83.3	**85.6**
Macedonian	1,956	70.6	**71.6**
Portuguese	1,096	97.4	95.8
Romanian	2,368	93.5	88.7
Serbian	2,156	65.3	**81.2**
Spanish	743	87.3	**91.5**
Swedish	1,107	93.5	90.3
Turkish	2,169	92.5	92.2
Urdu	1,093	70.7	**73.2**

Table 6: Entity Linking Accuracy (%) on Non-Wikipedia Data

ogy features contributed 11.1% and 7.1% absolute name tagging F-score gains to Turkish and Uzbek LORELEI data sets respectively.

Impact of Self-Training

Using Turkish as a case study, the learning curves of self-training on Wikipedia and non-Wikipedia test sets are shown in Figure 6. We can see that self-training provides significant improvement for both Wikipedia (6% absolute gain) and non-Wikipedia test data (12% absolute gain). As expected the learning curve on Wikipedia data is more smooth and converges more slowly than that of non-Wikipedia data. This indicates that when the training data is incomplete and noisy, the model can benefit from self-training through iterative label correction and propagation.

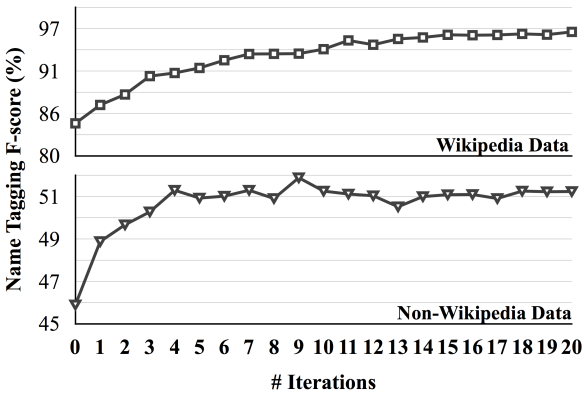


Figure 6: Learning Curve of Self-training

Impact of Topical Relatedness

We also found that the topical relatedness measure proposed in Section 2.5 not only significantly reduces the size of training data and thus speeds up the training process for many languages, but also consistently improves the quality. For example, the Turkish name tagger trained from the entire data set without topic selection yields 49.7% F-score on LORELEI data set, and the performance is improved to 51.5% after topic selection.

5 Related Work

Wikipedia markup based silver standard generation: Our work was mainly inspired from previous work that leveraged Wikipedia markups to train name taggers (Nothman et al., 2008; Dakka and Cucerzan, 2008; Mika et al., 2008; Ringland et al., 2009; Alotaibi and Lee, 2012; Nothman et al., 2013; Althobaiti et al., 2014). Most of these previous methods manually classified many English Wikipedia entries into pre-defined entity types. In contrast, our approach doesn't need any manual annotations or language-specific features, while generates both coarse-grained and fine-grained types.

Many fine-grained entity typing approaches (Fleischman and Hovy, 2002; Giuliano,

2009; Ekbal et al., 2010; Ling and Weld, 2012; Yosef et al., 2012; Nakashole et al., 2013; Gillick et al., 2014; Yogatama et al., 2015; Del Corro et al., 2015) also created annotations based on Wikipedia anchor links. Our framework performs both name identification and typing and takes advantage of richer structures in the KBs. Previous work on Arabic name tagging (Althobaiti et al., 2014) extracted entity titles as a gazetteer for stemming, and thus it cannot handle unknown names. We developed a new method to derive generalizable affixes for morphologically rich language based on Wikipedia markups.

Wikipedia as background features for IE: Wikipedia pages have been used as additional features to improve various Information Extraction (IE) tasks, including name tagging (Kazama and Torisawa, 2007), coreference resolution (Paolo Ponzetto and Strube, 2006), relation extraction (Chan and Roth, 2010) and event extraction (Hogue et al., 2014). Other automatic name annotation generation methods have been proposed, including KB driven distant supervision (An et al., 2003; Mintz et al., 2009; Ren et al., 2015) and cross-lingual projection (Li et al., 2012; Kim et al., 2012; Che et al., 2013; Wang et al., 2013; Wang and Manning, 2014; Zhang et al., 2016b).

Multi-lingual name tagging: Some recent research (Zhang et al., 2016a; Littell et al., 2016; Tsai et al., 2016) under the DARPA LORELEI program focused on developing name tagging techniques for low-resource languages. These approaches require English annotations for projection (Tsai et al., 2016), some input from a native speaker, either through manual annotations (Littell et al., 2016), or a linguistic survey (Zhang et al., 2016a). Without using any manual annotations, our name taggers outperform previous methods on the same data sets for many languages.

Multi-lingual entity linking: NIST TAC-KBP Tri-lingual entity linking (Ji et al., 2016) focused on three languages: English, Chinese and Spanish. (McNamee et al., 2011) extended it to 21 languages. But their methods required labeled data and name transliteration. We share the same goal as (Sil and Florian, 2016) to extend cross-lingual entity linking to all languages in Wikipedia. They exploited Wikipedia links to train a supervised linker. We mine reliable word translations from cross-lingual Wikipedia titles, which enables us to adopt unsupervised English entity linking techniques such as (Pan et al., 2015) to directly link translated English name mentions to English KB.

Efforts to save annotation cost for name tagging: Some previous work including (Ji and Grishman, 2006; Richman and Schone, 2008; Althobaiti et al., 2013) exploited semi-supervised methods to save annotation cost. We observed that self-training can provide further gains when the training data contains certain amount of noise.

6 Conclusions and Future Work

We developed a simple yet effective framework that can extract names from 282 languages and link them to an English KB. This framework follows a fully automatic training and testing pipeline, without the needs of any manual annotations or knowledge from native speakers. We evaluated our framework on both Wikipedia articles and external formal and informal texts and obtained promising results. To the best of our knowledge, our multilingual name tagging and linking framework is applied to the largest number of languages. We release the following resources for each of these 282 languages: "silver-standard" name tagging and linking annotations with multiple levels of granularity, morphology analyzer if it's a morphologically-rich language, and an end-to-end name tagging and linking system. In this work, we treat all languages independently when training their corresponding name taggers. In the future, we will explore the topological structure of related languages and exploit cross-lingual knowledge transfer to enhance the quality of extraction and linking. The general idea of deriving noisy annotations from KB properties can also be extended to other IE tasks such as relation extraction.

Acknowledgments

This work was supported by the U.S. DARPA LORELEI Program No. HR0011-15-C-0115, ARL/ARO MURI W911NF-10-1-0533, DARPA DEFT No. FA8750-13-2-0041 and FA8750-13-2-0045, and NSF CAREER No. IIS-1523198. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1024–1029. https://doi.org/10.3115/v1/N15-1107.

Fahd Alotaibi and Mark Lee. 2012. Mapping arabic wikipedia into the named entities taxonomy. In *Proceedings of COLING 2012: Posters*. The COLING 2012 Organizing Committee, pages 43–52. http://aclweb.org/anthology/C12-2005.

Maha Althobaiti, Udo Kruschwitz, and Massimo Poesio. 2013. A semi-supervised learning approach to arabic named entity recognition. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*. INCOMA Ltd. Shoumen, BULGARIA, pages 32–40. http://aclweb.org/anthology/R13-1005.

Maha Althobaiti, Udo Kruschwitz, and Massimo Poesio. 2014. Automatic creation of arabic named entity annotated corpus using wikipedia. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 106–115. https://doi.org/10.3115/v1/E14-3012.

Joohui An, Seungwoo Lee, and Gary Geunbae Lee. 2003. Automatic acquisition of named entity tagged corpus from world wide web. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P03-2031.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, pages 178–186. http://aclweb.org/anthology/W13-2322.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. ACM, New York, NY, USA, SIGMOD '08, pages 1247–1250. https://doi.org/10.1145/1376616.1376746.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. In *Proceedings of the Biennial GSCL Conference 2009*.

Seng Yee Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, pages 152–160. http://aclweb.org/anthology/C10-1018.

Wanxiang Che, Mengqiu Wang, D. Christopher Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 52–62. http://aclweb.org/anthology/N13-1006.

Wisam Dakka and Silviu Cucerzan. 2008. Augmenting wikipedia with named entity tags. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*. http://aclweb.org/anthology/I08-1071.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 868–878. https://doi.org/10.18653/v1/D15-1103.

Asif Ekbal, Eva Sourjikova, Anette Frank, and Simone Paolo Ponzetto. 2010. Assessing the challenge of fine-grained named entity recognition and classification. In *Proceedings of the 2010 Named Entities Workshop*. Association for Computational Linguistics, pages 93–101. http://aclweb.org/anthology/W10-2415.

Michael Fleischman and Eduard Hovy. 2002. Fine grained classification of named entities. In *COLING 2002: The 19th International Conference on Computational Linguistics*. http://aclweb.org/anthology/C02-1130.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *CoRR* abs/1412.1820. http://arxiv.org/abs/1412.1820.

Claudio Giuliano. 2009. Fine-grained classification of named entities exploiting latent semantic kernels. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*. Association for Computational Linguistics, pages 201–209. http://aclweb.org/anthology/W09-1125.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor flatcat: An hmm-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, pages 1177–1185. http://aclweb.org/anthology/C14-1111.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 110–120. https://doi.org/10.3115/v1/D14-1012.

Alexander Hogue, Joel Nothman, and James R. Curran. 2014. Unsupervised biographical event extraction using wikipedia traffic. In *Proceedings of the Australasian Language Technology Association Workshop 2014*. pages 41–49. http://aclweb.org/anthology/U14-1006.

Heng Ji and Ralph Grishman. 2006. Analysis and repair of name tagger errors. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. Association for Computational Linguistics, pages 420–427. http://aclweb.org/anthology/P06-2055.

Heng Ji, Joel Nothman, and Hoa Trang Dang. 2016. Overview of tac-kbp2016 tri-lingual edl and its impact on end-to-end kbp. In *Proceedings of the Text Analysis Conference*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics, Volume 29, Number 1, March 2003* http://aclweb.org/anthology/J03-1002.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. http://aclweb.org/anthology/D07-1073.

Sungchul Kim, Kristina Toutanova, and Hwanjo Yu. 2012. Multilingual named entity recognition using parallel data and metadata from wikipedia. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 694–702. http://aclweb.org/anthology/P12-1073.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 260–270. https://doi.org/10.18653/v1/N16-1030.

Hao Li, Heng Ji, Hongbo Deng, and Jiawei Han. 2011. Exploiting background information networks to enhance bilingual event extraction through topic modeling. In *Proceedings of International Conference on Advances in Information Mining and Management (IMMM2011)*.

Qi Li, Haibo Li, Heng Ji, Wen Wang, Jing Zheng, and Fei Huang. 2012. Joint bilingual name tagging for parallel corpora. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, USA, CIKM '12, pages 1727–1731. https://doi.org/10.1145/2396761.2398506.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'12, pages 94–100.

Patrick Littell, Kartik Goyal, R. David Mortensen, Alexa Little, Chris Dyer, and Lori Levin. 2016. Named entity recognition for linguistic rapid response in low-resource languages: Sorani kurdish and tajik. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 998–1006. http://aclweb.org/anthology/C16-1095.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. Yago3: A knowledge base from multilingual wikipedias. In *Proceedings of the Conference on Innovative Data Systems Research*.

Alireza Mahmoudi, Mohsen Arabsorkhi, and Heshaam Faili. 2013. Supervised morphology generation using parallel corpus. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*. INCOMA Ltd. Shoumen, BULGARIA, pages 408–414. http://aclweb.org/anthology/R13-1053.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, pages 55–60. https://doi.org/10.3115/v1/P14-5010.

James Mayfield, Dawn Lawrie, Paul McNamee, and Douglas W. Oard. 2011. Building a cross-language entity linking collection in twenty-one languages. In *Multilingual and Multimodal Information Access Evaluation: Second International Conference of the Cross-Language Evaluation Forum*.

Paul McNamee, James Mayfield, Dawn Lawrie, Douglas Oard, and David Doermann. 2011. Cross-language entity linking. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, pages 255–263. http://aclweb.org/anthology/I11-1029.

Peter Mika, Massimiliano Ciaramita, Hugo Zaragoza, and Jordi Atserias. 2008. Learning to tag and tagging to learn: A case study on wikipedia. *IEEE Intelligent Systems* .

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, pages 1003–1011. http://aclweb.org/anthology/P09-1113.

Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained semantic typing of emerging entities. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1488–1497. http://aclweb.org/anthology/P13-1146.

Joel Nothman, R. James Curran, and Tara Murphy. 2008. Transforming wikipedia into named entity training data. In *Proceedings of the Australasian Language Technology Association Workshop 2008*. pages 124–132. http://aclweb.org/anthology/U08-1016.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence* 194:151–175. https://doi.org/10.1016/j.artint.2012.03.006.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1130–1139. https://doi.org/10.3115/v1/N15-1119.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. http://aclweb.org/anthology/N06-1025.

Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R. Voss, and Jiawei Han. 2015. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '15, pages 995–1004. https://doi.org/10.1145/2783258.2783362.

E. Alexander Richman and Patrick Schone. 2008. Mining wiki resources for multilingual named entity recognition. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, pages 1–9. http://aclweb.org/anthology/P08-1001.

Nicky Ringland, Joel Nothman, Tara Murphy, and R. James Curran. 2009. Classifying articles in english and german wikipedia. In *Proceedings of the Australasian Language Technology Association Workshop 2009*. pages 20–28. http://aclweb.org/anthology/U09-1004.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *Proceedings of ACL-08: HLT, Short Papers*. Association for Computational Linguistics, pages 117–120. http://aclweb.org/anthology/P08-2030.

Teemu Ruokolainen, Oskar Kohonen, Kairit Sirts, Stig-Arne Grönroos, Mikko Kurimo, and Sami Virpioja. 2016. A comparative study of minimally supervised morphological segmentation. *Computational Linguistics* .

Avirup Sil and Radu Florian. 2016. One for all: Towards language independent named entity linking. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2255–2264. https://doi.org/10.18653/v1/P16-1213.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 219–228. https://doi.org/10.18653/v1/K16-1022.

Mengqiu Wang, Wanxiang Che, and D. Christopher Manning. 2013. Joint word alignment and bilingual named entity recognition using dual decomposition. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1073–1082. http://aclweb.org/anthology/P13-1106.

Mengqiu Wang and D. Christopher Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association of Computational Linguistics* 2:55–66. http://aclweb.org/anthology/Q14-1005.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms mutual information, and lexicography. *Computational Linguistics, Volume 16, Number 1, March 1990* http://aclweb.org/anthology/J90-1003.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 291–296. https://doi.org/10.3115/v1/P15-2048.

Amir Mohamed Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. In *Proceedings of COLING 2012: Posters*. The COLING 2012 Organizing Committee, pages 1361–1370. http://aclweb.org/anthology/C12-2133.

Boliang Zhang, Xiaoman Pan, Tianlu Wang, Ashish Vaswani, Heng Ji, Kevin Knight, and Daniel Marcu. 2016a. Name tagging for low-resource incident languages based on expectation-driven learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 249–259. https://doi.org/10.18653/v1/N16-1029.

Dongxu Zhang, Boliang Zhang, Xiaoman Pan, Xiaocheng Feng, Heng Ji, and Weiran XU. 2016b. Bitext name tagging for cross-lingual entity annotation projection. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 461–470. http://aclweb.org/anthology/C16-1045.

Adversarial Training for Unsupervised Bilingual Lexicon Induction

Meng Zhang[†‡] **Yang Liu**[†‡*] **Huanbo Luan**[†] **Maosong Sun**[†‡]
[†]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[‡]Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China
zmlarry@foxmail.com, liuyang2011@tsinghua.edu.cn
luanhuanbo@gmail.com, sms@tsinghua.edu.cn

Abstract

Word embeddings are well known to capture linguistic regularities of the language on which they are trained. Researchers also observe that these regularities can transfer across languages. However, previous endeavors to connect separate monolingual word embeddings typically require cross-lingual signals as supervision, either in the form of parallel corpus or seed lexicon. In this work, we show that such cross-lingual connection can actually be established without any form of supervision. We achieve this end by formulating the problem as a natural adversarial game, and investigating techniques that are crucial to successful training. We carry out evaluation on the unsupervised bilingual lexicon induction task. Even though this task appears intrinsically cross-lingual, we are able to demonstrate encouraging performance without any cross-lingual clues.

1 Introduction

As word is the basic unit of a language, the betterment of its representation has significant impact on various natural language processing tasks. Continuous word representations, commonly known as word embeddings, have formed the basis for numerous neural network models since their advent. Their popularity results from the performance boost they bring, which should in turn be attributed to the linguistic regularities they capture (Mikolov et al., 2013b).

Soon following the success on monolingual tasks, the potential of word embeddings for cross-lingual natural language processing has attracted much attention. In their pioneering work, Mikolov et al. (2013a) observe that word embeddings trained separately on monolingual corpora exhibit isomorphic structure across languages, as illustrated in Figure 1. This interesting finding is in line with research on human cognition (Youn et al., 2016). It also means a linear transformation may be established to connect word embedding spaces, allowing word feature transfer. This has far-reaching implication on low-resource scenarios (Daumé III and Jagarlamudi, 2011; Irvine and Callison-Burch, 2013), because word embeddings only require plain text to train, which is the most abundant form of linguistic resource.

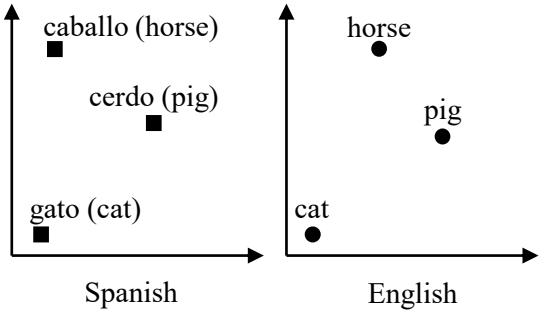


Figure 1: Illustrative monolingual word embeddings of Spanish and English, adapted from (Mikolov et al., 2013a). Although trained independently, the two sets of embeddings exhibit approximate isomorphism.

However, connecting separate word embedding spaces typically requires supervision from cross-lingual signals. For example, Mikolov et al. (2013a) use five thousand seed word translation pairs to train the linear transformation. In a recent study, Vulić and Korhonen (2016) show that at least hundreds of seed word translation pairs are needed for the model to generalize. This is unfortunate for low-resource languages and domains,

*Corresponding author.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1959–1970
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1179

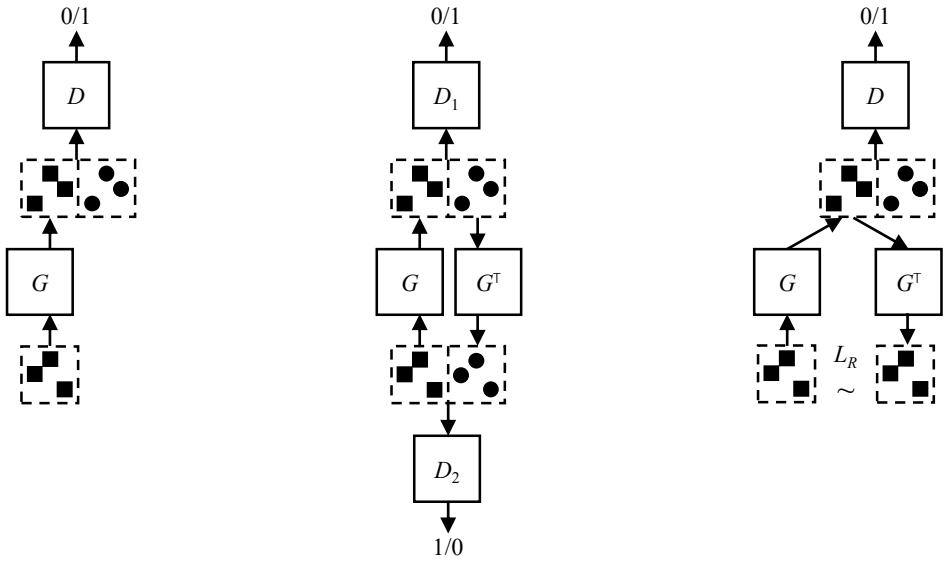


Figure 2: (a) The unidirectional transformation model directly inspired by the adversarial game: The generator G tries to transform source word embeddings (squares) to make them seem like target ones (dots), while the discriminator D tries to classify whether the input embeddings are generated by G or real samples from the target embedding distribution. (b) The bidirectional transformation model. Two generators with tied weights perform transformation between languages. Two separate discriminators are responsible for each language. (c) The adversarial autoencoder model. The generator aims to make the transformed embeddings not only indistinguishable by the discriminator, but also recoverable as measured by the reconstruction loss L_R.

because data encoding cross-lingual equivalence is often expensive to obtain.

In this work, we aim to entirely eliminate the need for cross-lingual supervision. Our approach draws inspiration from recent advances in generative adversarial networks (Goodfellow et al., 2014). We first formulate our task in a fashion that naturally admits an adversarial game. Then we propose three models that implement the game, and explore techniques to ensure the success of training. Finally, our evaluation on the bilingual lexicon induction task reveals encouraging performance, even though this task appears formidable without any cross-lingual supervision.

2 Models

In order to induce a bilingual lexicon, we start from two sets of monolingual word embeddings with dimensionality d. They are trained separately on two languages. Our goal is to learn a mapping function $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ so that for a source word embedding x, $f(x)$ lies close to the embedding of its target language translation y. The learned mapping function can then be used to translate each source word x by finding the nearest target embedding to $f(x)$.

We consider x to be drawn from a distribution p_x, and similarly $y \sim p_y$. The key intuition here is to find the mapping function to make $f(x)$ seem to follow the distribution p_y, for all $x \sim p_x$. From this point of view, we design an adversarial game as illustrated in Figure 2(a): The generator G implements the mapping function f, trying to make $f(x)$ passable as target word embeddings, while the discriminator D is a binary classifier striving to distinguish between fake target word embeddings $f(x) \sim p_{f(x)}$ and real ones $y \sim p_y$. This intuition can be formalized as the minimax game $\min_G \max_D V(D, G)$ with value function

$$\begin{aligned} & V(D,G) \\ =& \mathbb{E}_{y\sim p_y}\left[\log D(y)\right] + \\ & \mathbb{E}_{x\sim p_x}\left[\log\left(1 - D\left(G\left(x\right)\right)\right)\right]. \end{aligned} \tag{1}$$

Theoretical analysis reveals that adversarial training tries to minimize the Jensen-Shannon divergence $\text{JSD}\left(p_y || p_{f(x)}\right)$ (Goodfellow et al., 2014). Importantly, the minimization happens at the distribution level, without requiring word

translation pairs to supervise training.

2.1 Model 1: Unidirectional Transformation

The first model directly implements the adversarial game, as shown in Figure 2(a). As hinted by the isomorphism shown in Figure 1, previous works typically choose the mapping function f to be a linear map (Mikolov et al., 2013a; Dinu et al., 2015; Lazaridou et al., 2015). We therefore parametrize the generator as a transformation matrix $G \in \mathbb{R}^{d \times d}$. We also tried non-linear maps parametrized by neural networks, without success. In fact, if the generator is given sufficient capacity, it can in principle learn a constant mapping function to a target word embedding, which makes the discriminator impossible to distinguish, much like the "mode collapse" problem widely observed in the image domain (Radford et al., 2015; Salimans et al., 2016). We therefore believe it is crucial to grant the generator with suitable capacity.

As a generic binary classifier, a standard feed-forward neural network with one hidden layer is used to parametrize the discriminator D, and its loss function is the usual cross-entropy loss, as in the value function (1):

$$L_D = -\log D(y) - \log(1 - D(Gx)). \quad (2)$$

For simplicity, here we write the loss with a mini-batch size of 1; in our experiments we use 128.

The generator loss is given by

$$L_G = -\log D(Gx). \quad (3)$$

In line with previous work (Goodfellow et al., 2014), we find this loss easier to minimize than the original form $\log(1 - D(Gx))$.

Orthogonal Constraint

The above model is very difficult to train. One possible reason is that the parameter search space $\mathbb{R}^{d \times d}$ for the generator may still be too large. Previous works have attempted to constrain the transformation matrix to be orthogonal (Xing et al., 2015; Zhang et al., 2016b; Artetxe et al., 2016). An orthogonal transformation is also theoretically appealing for its self-consistency (Smith et al., 2017) and numerical stability. However, using constrained optimization for our purpose is cumbersome, so we opt for an orthogonal parametrization (Mhammedi et al., 2016) of the generator instead.

2.2 Model 2: Bidirectional Transformation

The orthogonal parametrization is still quite slow. We can relax the orthogonal constraint and only require the transformation to be self-consistent (Smith et al., 2017): If G transforms the source word embedding space into the target language space, its transpose $G^\top$ should transform the target language space back to the source. This can be implemented by two unidirectional models with a tied generator, as illustrated in Figure 2(b). Two separate discriminators are used, with the same cross-entropy loss as Equation (2) used by Model 1. The generator loss is given by

$$L_G = -\log D_1(Gx) - \log D_2\left(G^\top x\right). \quad (4)$$

2.3 Model 3: Adversarial Autoencoder

As another way to relax the orthogonal constraint, we introduce the adversarial autoencoder (Makhzani et al., 2015), depicted in Figure 2(c). After the generator G transforms a source word embedding x into a target language representation Gx, we should be able to reconstruct the source word embedding x by mapping back with $G^\top$. We therefore introduce the reconstruction loss measured by cosine similarity:

$$L_R = -\cos\left(x, G^\top Gx\right). \quad (5)$$

Note that this loss will be minimized if G is orthogonal. With this term included, the loss function for the generator becomes

$$L_G = -\log D(Gx) - \lambda \cos\left(x, G^\top Gx\right), \quad (6)$$

where λ is a hyperparameter that balances the two terms. $\lambda = 0$ recovers the unidirectional transformation model, while larger λ should enforce a stricter orthogonal constraint.

3 Training Techniques

Generative adversarial networks are notoriously difficult to train, and investigation into stabler training remains a research frontier (Radford et al., 2015; Salimans et al., 2016; Arjovsky and Bottou, 2017). We contribute in this aspect by reporting techniques that are crucial to successful training for our task.

3.1 Regularizing the Discriminator

Recently, it has been suggested to inject noise into the input to the discriminator (Sønderby et al.,

2016; Arjovsky and Bottou, 2017). The noise is typically additive Gaussian. Here we explore more possibilities, with the following types of noise, injected into the input and hidden layer:

- Multiplicative Bernoulli noise (dropout) (Srivastava et al., 2014): $\epsilon \sim \text{Bernoulli}\,(p)$.
- Additive Gaussian noise: $\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$.
- Multiplicative Gaussian noise: $\epsilon \sim \mathcal{N}\left(1, \sigma^2\right)$.

As noise injection is a form of regularization (Bishop, 1995; Van der Maaten et al., 2013; Wager et al., 2013), we also try l_2 regularization, and directly restricting the hidden layer size to combat overfitting. Our findings include:

- Without regularization, it is not impossible for the optimizer to find a satisfactory parameter configuration, but the hidden layer size has to be tuned carefully. This indicates that a balance of capacity between the generator and discriminator is needed.
- All forms of regularization help training by allowing us to liberally set the hidden layer size to a relatively large value.
- Among the types of regularization, multiplicative Gaussian injected into the input is the most effective, and additive Gaussian is similar. On top of input noise, hidden layer noise helps slightly.

In the following experiments, we inject multiplicative Gaussian into the input and hidden layer of the discriminator with $\sigma = 0.5$.

3.2 Model Selection

From a typical training trajectory shown in Figure 3, we observe that training is not convergent. In fact, simply using the model saved at the end of training gives poor performance. Therefore we need a mechanism to select a good model. We observe there are sharp drops of the generator loss L_G, and find they correspond to good models, as the discriminator gets confused at these points with its classification accuracy (D accuracy) dropping simultaneously. Interestingly, the reconstruction loss L_R and the value of $\left\|G^\top G - I\right\|_F$ exhibit synchronous drops, even if we use the unidirectional transformation model ($\lambda = 0$). This means a good transformation matrix is indeed nearly orthogonal, and justifies our encouragement of G towards orthogonality. With this finding, we can train for sufficient steps and save the model with the lowest generator loss.

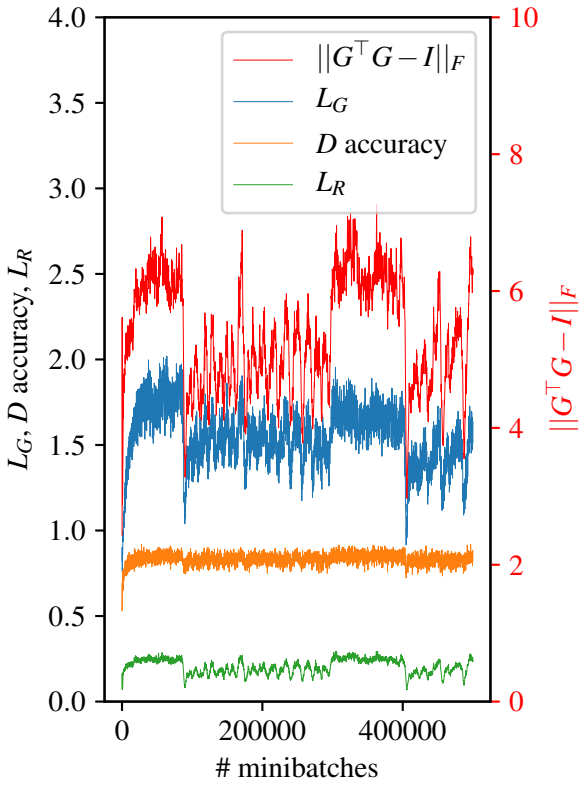


Figure 3: A typical training trajectory of the adversarial autoencoder model with $\lambda = 1$. The values are averages within each minibatch.

As we aim to find the cross-lingual transformation without supervision, it would be ideal to determine hyperparameters without a validation set. The sharp drops can also be indicative in this case. If a hyperparameter configuration is poor, those values will oscillate without a clear drop. Although this criterion is somewhat subjective, we find it to be quite feasible in practice.

3.3 Other Training Details

Our approach takes monolingual word embeddings as input. We train the CBOW model (Mikolov et al., 2013b) with default hyperparameters in `word2vec`.[1] The embedding dimension d is 50 unless stated otherwise. Before feeding them into our system, we normalize the word embeddings to unit length. When sampling words for adversarial training, we penalize frequent words in a way similar to (Mikolov et al., 2013b). G is

[1] https://code.google.com/archive/p/word2vec

initialized with a random orthogonal matrix. The hidden layer size of D is 500. Adversarial training involves alternate gradient update of the generator and discriminator, which we implement with a simpler variant algorithm described in (Nowozin et al., 2016). Adam (Kingma and Ba, 2014) is used as the optimizer, with default hyperparameters. For the adversarial autoencoder model, $\lambda = 1$ generally works well, but $\lambda = 10$ appears stabler for the low-resource Turkish-English setting.

4 Experiments

We evaluate the quality of the cross-lingual embedding transformation on the bilingual lexicon induction task. After a source word embedding is transformed into the target space, its M nearest target embeddings (in terms of cosine similarity) are retrieved, and compared against the entry in a ground truth bilingual lexicon. Performance is measured by top-M accuracy (Vulić and Moens, 2013): If any of the M translations is found in the ground truth bilingual lexicon, the source word is considered to be handled correctly, and the accuracy is calculated as the percentage of correctly translated source words. We generally report the harshest top-1 accuracy, unless when comparing with published figures in Section 4.4.

Baselines

Almost all approaches to bilingual lexicon induction from non-parallel data depend on seed lexica. An exception is decipherment (Dou and Knight, 2012; Dou et al., 2015), and we use it as our baseline. The decipherment approach is not based on distributional semantics, but rather views the source language as a cipher for the target language, and attempts to learn a statistical model to decipher the source language. We run the MonoGiza system as recommended by the toolkit.[2] It can also utilize monolingual embeddings (Dou et al., 2015); in this case, we use the same embeddings as the input to our approach.

Sharing the underlying spirit with our approach, related methods also build upon monolingual word embeddings and find transformation to link different languages. Although they need seed word translation pairs to train and thus not directly comparable, we report their performance with 50 and 100 seeds for reference. These methods are:

		# tokens	vocab. size
Wikipedia comparable corpora			
zh-en	zh	21m	3,349
	en	53m	5,154
es-en	es	61m	4,774
	en	95m	6,637
it-en	it	73m	8,490
	en	93m	6,597
ja-zh	ja	38m	6,043
	zh	16m	2,814
tr-en	tr	6m	7,482
	en	28m	13,220
Large-scale settings			
zh-en Wikipedia	zh	143m	14,686
	en	1,907m	61,899
zh-en Gigaword	zh	2,148m	45,958
	en	5,017m	73,504

Table 1: Statistics of the non-parallel corpora. Language codes: zh = Chinese, en = English, es = Spanish, it = Italian, ja = Japanese, tr = Turkish.

- Translation matrix (`TM`) (Mikolov et al., 2013a): the pioneer of this type of methods mentioned in the introduction, using linear transformation. We use a publicly available implementation.[3]

- Isometric alignment (`IA`) (Zhang et al., 2016b): an extension of `TM` by augmenting its learning objective with the isometric (orthogonal) constraint. Although Zhang et al. (2016b) had subsequent steps for their POS tagging task, it could be used for bilingual lexicon induction as well.

We ensure the same input embeddings for these methods and ours.

The seed word translation pairs are obtained as follows. First, we ask Google Translate[4] to translate the source language vocabulary. Then the target translations are queried again and translated back to the source language, and those that do not match the original source words are discarded. This helps to ensure the translation quality. Finally, the translations are discarded if they fall out of our target language vocabulary.

[2] http://www.isi.edu/natural-language/software/monogiza_release_v1.0.tar.gz

[3] http://clic.cimec.unitn.it/~georgiana.dinu/down

[4] https://translate.google.com

method	# seeds	accuracy (%)
MonoGiza w/o emb.	0	0.05
MonoGiza w/ emb.	0	0.09
TM	50	0.29
	100	21.79
IA	50	18.71
	100	32.29
Model 1	0	39.25
Model 1 + ortho.	0	28.62
Model 2	0	40.28
Model 3	0	43.31

Table 2: Chinese-English top-1 accuracies of the MonoGiza baseline and our models, along with the translation matrix (TM) and isometric alignment (IA) methods that utilize 50 and 100 seeds.

4.1 Experiments on Chinese-English

Data

For this set of experiments, the data for training word embeddings comes from Wikipedia comparable corpora.[5] Following (Vulić and Moens, 2013), we retain only nouns with at least 1,000 occurrences. For the Chinese side, we first use OpenCC[6] to normalize characters to be simplified, and then perform Chinese word segmentation and POS tagging with THULAC.[7] The preprocessing of the English side involves tokenization, POS tagging, lemmatization, and lowercasing, which we carry out with the NLTK toolkit.[8] The statistics of the final training data is given in Table 1, along with the other experimental settings.

As the ground truth bilingual lexicon for evaluation, we use Chinese-English Translation Lexicon Version 3.0 (LDC2002L27).

Overall Performance

Table 2 lists the performance of the MonoGiza baseline and our four variants of adversarial training. MonoGiza obtains low performance, likely due to the harsh evaluation protocol (cf. Section 4.4). Providing it with syntactic information can help (Dou and Knight, 2013), but in a low-resource scenario with zero cross-lingual information, parsers are likely to be inaccurate or even unavailable.

[5] http://linguatools.org/tools/corpora/wikipedia-comparable-corpora

[6] https://github.com/BYVoid/OpenCC

[7] http://thulac.thunlp.org

[8] http://www.nltk.org

城市 *chengshi*	小行星 *xiaoxingxing*	文学 *wenxue*
city	**asteroid**	poetry
town	astronomer	**literature**
suburb	comet	prose
area	constellation	poet
proximity	orbit	writing

Table 3: Top-5 English translation candidates proposed by our approach for some Chinese words. The ground truth is marked in bold.

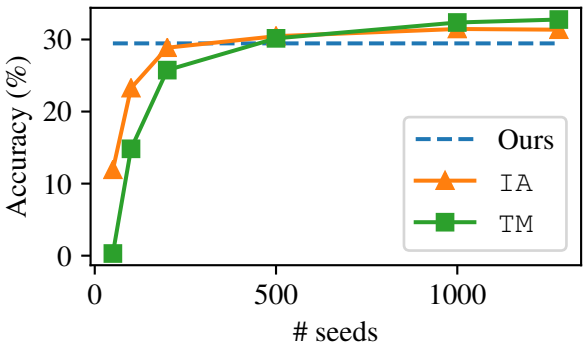


Figure 4: Top-1 accuracies of our approach, isometric alignment (IA), and translation matrix (TM), with the number of seeds varying in {50, 100, 200, 500, 1000, 1280}.

The unidirectional transformation model attains reasonable accuracy if trained successfully, but it is rather sensitive to hyperparameters and initialization. This training difficulty motivates our orthogonal constraint. But imposing a strict orthogonal constraint hurts performance. It is also about 20 times slower even though we utilize orthogonal parametrization instead of constrained optimization. The last two models represent different relaxations of the orthogonal constraint, and the adversarial autoencoder model achieves the best performance. We therefore use it in our following experiments. Table 3 lists some word translation examples given by the adversarial autoencoder model.

Comparison With Seed-Based Methods

In this section, we investigate how many seeds TM and IA require to attain the performance level of our approach. There are a total of 1,280 seed translation pairs for Chinese-English, which are removed from the test set during the evaluation for this experiment. We use the most frequent S pairs for TM and IA.

Figure 4 shows the accuracies with respect to

method	# seeds	es-en	it-en	ja-zh	tr-en
MonoGiza w/o embeddings	0	0.35	0.30	0.04	0.00
MonoGiza w/ embeddings	0	1.19	0.27	0.23	0.09
TM	50	1.24	0.76	0.35	0.09
	100	48.61	37.95	26.67	11.15
IA	50	39.89	27.03	19.04	7.58
	100	60.44	46.52	36.35	17.11
Ours	0	71.97	58.60	43.02	17.18

Table 4: Top-1 accuracies (%) of the MonoGiza baseline and our approach on Spanish-English, Italian-English, Japanese-Chinese, and Turkish-English. The results for translation matrix (TM) and isometric alignment (IA) using 50 and 100 seeds are also listed.

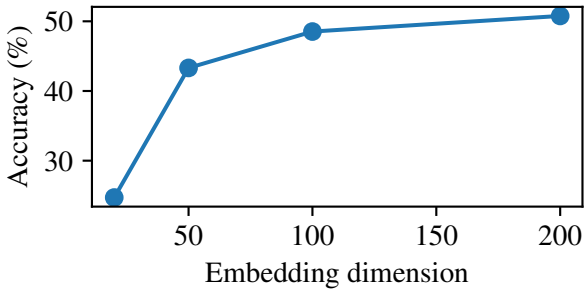


Figure 5: Top-1 accuracies of our approach with respect to the input embedding dimensions in {20, 50, 100, 200}.

S. When the seeds are few, the seed-based methods exhibit clear performance degradation. In this case, we also observe the importance of the orthogonal constraint from the superiority of IA to TM, which supports our introduction of this constraint as we attempt zero supervision. Finally, in line with the finding in (Vulić and Korhonen, 2016), hundreds of seeds are needed for TM to generalize. Only then do seed-based methods catch up with our approach, and the performance difference is marginal even when more seeds are provided.

Effect of Embedding Dimension

As our approach takes monolingual word embeddings as input, it is conceivable that their quality significantly affects how well the two spaces can be connected by a linear map. We look into this aspect by varying the embedding dimension d in Figure 5. As the dimension increases, the accuracy improves and gradually levels off. This indicates that too low a dimension hampers the encoding of linguistic information drawn from the corpus, and it is advisable to use a sufficiently large dimension.

4.2 Experiments on Other Language Pairs

Data

We also induce bilingual lexica from Wikipedia comparable corpora for the following language pairs: Spanish-English, Italian-English, Japanese-Chinese, and Turkish-English. For Spanish-English and Italian-English, we choose to use TreeTagger[9] for preprocessing, as in (Vulić and Moens, 2013). For the Japanese corpus, we use MeCab[10] for word segmentation and POS tagging. For Turkish, we utilize the preprocessing tools (tokenization and POS tagging) provided in LORELEI Language Packs (Strassel and Tracey, 2016), and its English side is preprocessed by NLTK. Unlike the other language pairs, the frequency cutoff threshold for Turkish-English is 100, as the amount of data is relatively small.

The ground truth bilingual lexica for Spanish-English and Italian-English are obtained from Open Multilingual WordNet[11] through NLTK. For Japanese-Chinese, we use an in-house lexicon. For Turkish-English, we build a set of ground truth translation pairs in the same way as how we obtain seed word translation pairs from Google Translate, described above.

Results

As shown in Table 4, the MonoGiza baseline still does not work well on these language pairs, while our approach achieves much better performance. The accuracies are particularly high for Spanish-English and Italian-English, likely because they are closely related languages, and their embedding spaces may exhibit stronger isomorphism. The

[9] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger

[10] http://taku910.github.io/mecab

[11] http://compling.hss.ntu.edu.sg/omw

method	# seeds	Wikipedia	Gigaword
TM	50	0.00	0.01
	100	4.79	2.07
IA	50	3.25	1.68
	100	7.08	4.18
Ours	0	7.92	2.53

Table 5: Top-1 accuracies (%) of our approach to inducing bilingual lexica for Chinese-English from Wikipedia and Gigaword. Also listed are results for translation matrix (TM) and isometric alignment (IA) using 50 and 100 seeds.

performance on Japanese-Chinese is lower, on a comparable level with Chinese-English (cf. Table 2), and these languages are relatively distantly related. Turkish-English represents a low-resource scenario, and therefore the lexical semantic structure may be insufficiently captured by the embeddings. The agglutinative nature of Turkish can also add to the challenge.

4.3 Large-Scale Settings

We experiment with large-scale Chinese-English data from two sources: the whole Wikipedia dump and Gigaword (LDC2011T13 and LDC2011T07). We also simplify preprocessing by removing the noun restriction and the lemmatization step (cf. preprocessing decisions for the above experiments).

Although large-scale data may benefit the training of embeddings, it poses a greater challenge to bilingual lexicon induction. First, the degree of non-parallelism tends to increase. Second, with cruder preprocessing, the noise in the corpora may take its toll. Finally, but probably most importantly, the vocabularies expand dramatically compared to previous settings (see Table 1). This means a word translation has to be retrieved from a much larger pool of candidates.

For these reasons, we consider the performance of our approach presented in Table 5 to be encouraging. The imbalanced sizes of the Chinese and English Wikipedia do not seem to cause a problem for the structural isomorphism needed by our method. MonoGiza does not scale to such large vocabularies, as it already takes days to train in our Italian-English setting. In contrast, our approach is immune from scalability issues by working with embeddings provided by word2vec, which is well known for its fast speed. With the network configuration used in our experiments, the adversarial autoencoder model takes about two hours to train for 500k minibatches on a single CPU.

method	5k	10k
MonoGiza w/o embeddings	13.74	7.80
MonoGiza w/ embeddings	17.98	10.56
(Cao et al., 2016)	23.54	17.82
Ours	68.59	51.86

Table 6: Top-5 accuracies (%) of 5k and 10k most frequent words in the French-English setting. The figures for the baselines are taken from (Cao et al., 2016).

4.4 Comparison With (Cao et al., 2016)

In order to compare with the recent method by Cao et al. (2016), which also uses zero cross-lingual signal to connect monolingual embeddings, we replicate their French-English experiment to test our approach.[12] This experimental setting has important differences from the above ones, mostly in the evaluation protocol. Apart from using top-5 accuracy as the evaluation metric, the ground truth bilingual lexicon is obtained by performing word alignment on a parallel corpus. We find this automatically constructed bilingual lexicon to be noisier than the ones we use for the other language pairs; it often lists tens of translations for a source word. This lenient evaluation protocol should explain MonoGiza's higher numbers in Table 6 than what we report in the other experiments. In this setting, our approach is able to considerably outperform both MonoGiza and the method by Cao et al. (2016).

5 Related Work

5.1 Cross-Lingual Word Embeddings for Bilingual Lexicon Induction

Inducing bilingual lexica from non-parallel data is a long-standing cross-lingual task. Except for the decipherment approach, traditional statistical methods all require cross-lingual signals (Rapp, 1999; Koehn and Knight, 2002; Fung and Cheung, 2004; Gaussier et al., 2004; Haghighi et al., 2008; Vulić et al., 2011; Vulić and Moens, 2013).

Recent advances in cross-lingual word embeddings (Vulić and Korhonen, 2016; Upadhyay et al.,

[12] As a confirmation, we ran MonoGiza in this setting and obtained comparable performance as reported.

2016) have rekindled interest in bilingual lexicon induction. Like their traditional counterparts, these embedding-based methods require cross-lingual signals encoded in parallel data, aligned at document level (Vulić and Moens, 2015), sentence level (Zou et al., 2013; Chandar A P et al., 2014; Hermann and Blunsom, 2014; Kočiský et al., 2014; Gouws et al., 2015; Luong et al., 2015; Coulmance et al., 2015; Oshikiri et al., 2016), or word level (i.e. seed lexicon) (Gouws and Søgaard, 2015; Wick et al., 2016; Duong et al., 2016; Shi et al., 2015; Mikolov et al., 2013a; Dinu et al., 2015; Lazaridou et al., 2015; Faruqui and Dyer, 2014; Lu et al., 2015; Ammar et al., 2016; Zhang et al., 2016a, 2017; Smith et al., 2017). In contrast, our work completely removes the need for cross-lingual signals to connect monolingual word embeddings, trained on non-parallel text corpora.

As one of our baselines, the method by Cao et al. (2016) also does not require cross-lingual signals to train bilingual word embeddings. It modifies the objective for training embeddings, whereas our approach uses monolingual embeddings trained beforehand and held fixed. More importantly, its learning mechanism is substantially different from ours. It encourages word embeddings from different languages to lie in the shared semantic space by matching the mean and variance of the hidden states, assumed to follow a Gaussian distribution, which is hard to justify. Our approach does not make any assumptions and directly matches the mapped source embedding distribution with the target distribution by adversarial training.

A recent work also attempts adversarial training for cross-lingual embedding transformation (Barone, 2016). The model architectures are similar to ours, but the reported results are not positive. We tried the publicly available code on our data, but the results were not positive, either. Therefore, we attribute the outcome to the difference in the loss and training techniques, but not the model architectures or data.

5.2 Adversarial Training

Generative adversarial networks are originally proposed for generating realistic images as an implicit generative model, but the adversarial training technique for matching distributions is generalizable to much more tasks, including natural language processing. For example, Ganin et al. (2016) address domain adaptation by adversarially training features to be domain invariant, and test on sentiment classification. Chen et al. (2016) extend this idea to cross-lingual sentiment classification. Our research deals with unsupervised bilingual lexicon induction based on word embeddings, and therefore works with word embedding distributions, which are more interpretable than the neural feature space of classifiers in the above works.

In the field of neural machine translation, a recent work (He et al., 2016) proposes dual learning, which also involves a two-agent game and therefore bears conceptual resemblance to the adversarial training idea. The framework is carried out with reinforcement learning, and thus differs greatly in implementation from adversarial training.

6 Conclusion

In this work, we demonstrate the feasibility of connecting word embeddings of different languages without any cross-lingual signal. This is achieved by matching the distributions of the transformed source language embeddings and target ones via adversarial training. The success of our approach signifies the existence of universal lexical semantic structure across languages. Our work also opens up opportunities for the processing of extremely low-resource languages and domains that lack parallel data completely.

Our work is likely to benefit from advances in techniques that further stabilize adversarial training. Future work also includes investigating other divergences that adversarial training can minimize (Nowozin et al., 2016), and broader mathematical tools that match distributions (Mohamed and Lakshminarayanan, 2016).

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work is supported by the National Natural Science Foundation of China (No. 61522204), the 973 Program (2014CB340501), and the National Natural Science Foundation of China (No. 61331013). This research is also supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively Multilingual Word Embeddings. *arXiv:1602.01925 [cs]* http://arxiv.org/abs/1602.01925.

Martin Arjovsky and Léon Bottou. 2017. Towards Principled Methods For Training Generative Adversarial Networks. In *ICLR*. http://arxiv.org/abs/1701.04862.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *EMNLP*. http://aclanthology.info/papers/learning-principled-bilingual-mappings-of-word-embeddings-while-preserving-monolingual-invariance.

Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. https://doi.org/10.18653/v1/W16-1614.

Chris M. Bishop. 1995. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Comput.* https://doi.org/10.1162/neco.1995.7.1.108.

Hailong Cao, Tiejun Zhao, Shu Zhang, and Yao Meng. 2016. A Distribution-based Model to Learn Bilingual Word Embeddings. In *COLING*. http://aclanthology.info/papers/a-distribution-based-model-to-learn-bilingual-word-embeddings.

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An Autoencoder Approach to Learning Bilingual Word Representations. In *NIPS*. http://papers.nips.cc/paper/5270-an-autoencoder-approach-to-learning-bilingual-word-representations.pdf.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial Deep Averaging Networks for Cross-Lingual Sentiment Classification. *arXiv:1606.01614 [cs]* http://arxiv.org/abs/1606.01614.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, Fast Cross-lingual Word-embeddings. In *EMNLP*. http://aclanthology.info/papers/trans-gram-fast-cross-lingual-word-embeddings.

Hal Daumé III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *ACL-HLT*. http://aclweb.org/anthology/P11-2071.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving Zero-Shot Learning by Mitigating the Hubness Problem. In *ICLR Workshop*. http://arxiv.org/abs/1412.6568.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *EMNLP-CoNLL*. http://aclweb.org/anthology/D12-1025.

Qing Dou and Kevin Knight. 2013. Dependency-Based Decipherment for Resource-Limited Machine Translation. In *EMNLP*. http://aclanthology.info/papers/dependency-based-decipherment-for-resource-limited-machine-translation.

Qing Dou, Ashish Vaswani, Kevin Knight, and Chris Dyer. 2015. Unifying Bayesian Inference and Vector Space Models for Improved Decipherment. In *ACL-IJCNLP*. http://www.aclweb.org/anthology/P15-1081.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning Crosslingual Word Embeddings without Bilingual Corpora. In *EMNLP*. http://aclanthology.info/papers/learning-crosslingual-word-embeddings-without-bilingual-corpora.

Manaal Faruqui and Chris Dyer. 2014. Improving Vector Space Word Representations Using Multilingual Correlation. In *EACL*. http://aclanthology.info/papers/improving-vector-space-word-representations-using-multilingual-correlation.

Pascale Fung and Percy Cheung. 2004. Mining Very-Non-Parallel Corpora: Parallel Sentence and Lexicon Extraction via Bootstrapping and EM. In *EMNLP*. http://aclweb.org/anthology/W04-3208.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research* http://jmlr.org/papers/v17/15-239.html.

Eric Gaussier, J.M. Renders, I. Matveeva, C. Goutte, and H. Dejean. 2004. A Geometric View on Bilingual Lexicon Extraction from Comparable Corpora. In *ACL*. https://doi.org/10.3115/1218955.1219022.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. In *ICML*. http://jmlr.org/proceedings/papers/v37/gouws15.html.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *NAACL-HLT*. http://www.aclweb.org/anthology/N15-1157.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning Bilingual Lexicons from Monolingual Corpora. In *ACL-HLT*. http://aclanthology.info/papers/learning-bilingual-lexicons-from-monolingual-corpora.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. 2016. Dual Learning for Machine Translation. In *NIPS*. http://papers.nips.cc/paper/6469-dual-learning-for-machine-translation.pdf.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Distributed Representations without Word Alignment. In *ICLR*. http://arxiv.org/abs/1312.6173.

Ann Irvine and Chris Callison-Burch. 2013. Combining bilingual and comparable corpora for low resource machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*. http://aclweb.org/anthology/W13-2233.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* http://arxiv.org/abs/1412.6980.

Philipp Koehn and Kevin Knight. 2002. Learning a Translation Lexicon from Monolingual Corpora. In *ACL Workshop on Unsupervised Lexical Acquisition*. https://doi.org/10.3115/1118627.1118629.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning Bilingual Word Representations by Marginalizing Alignments. In *ACL*. http://aclanthology.info/papers/learning-bilingual-word-representations-by-marginalizing-alignments.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and Pollution: Delving into Cross-Space Mapping for Zero-Shot Learning. In *ACL-IJCNLP*. https://doi.org/10.3115/v1/P15-1027.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep Multilingual Correlation for Improved Word Embeddings. In *NAACL-HLT*. http://aclanthology.info/papers/deep-multilingual-correlation-for-improved-word-embeddings.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual Word Representations with Monolingual Quality in Mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. http://aclanthology.info/papers/bilingual-word-representations-with-monolingual-quality-in-mind.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial Autoencoders. *arXiv:1511.05644 [cs]* http://arxiv.org/abs/1511.05644.

Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. 2016. Efficient Orthogonal Parametrisation of Recurrent Neural Networks Using Householder Reflections. *arXiv:1612.00188 [cs]* http://arxiv.org/abs/1612.00188.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting Similarities among Languages for Machine Translation. *arXiv:1309.4168 [cs]* http://arxiv.org/abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Shakir Mohamed and Balaji Lakshminarayanan. 2016. Learning in Implicit Generative Models. *arXiv:1610.03483 [cs, stat]* http://arxiv.org/abs/1610.03483.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. *arXiv:1606.00709 [cs, stat]* http://arxiv.org/abs/1606.00709.

Takamasa Oshikiri, Kazuki Fukui, and Hidetoshi Shimodaira. 2016. Cross-Lingual Word Representations via Spectral Graph Embeddings. In *ACL*. https://doi.org/10.18653/v1/P16-2080.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434 [cs]* http://arxiv.org/abs/1511.06434.

Reinhard Rapp. 1999. Automatic Identification of Word Translations from Unrelated English and German Corpora. In *ACL*. https://doi.org/10.3115/1034678.1034756.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved Techniques for Training GANs. In *NIPS*. http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning Cross-lingual Word Embeddings via Matrix Co-factorization. In *ACL-IJCNLP*. http://aclanthology.info/papers/learning-cross-lingual-word-embeddings-via-matrix-co-factorization.

Samuel Smith, David Turban, Steven Hamblin, and Nils Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *ICLR*. http://arxiv.org/abs/1702.03859.

Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. 2016. Amortised MAP Inference for Image Super-resolution. *arXiv:1610.04490 [cs, stat]* http://arxiv.org/abs/1610.04490.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* http://www.jmlr.org/papers/v15/srivastava14a.html.

Stephanie Strassel and Jennifer Tracey. 2016. LORELEI Language Packs: Data, Tools, and Resources for Technology Development in Low Resource Languages. In *LREC*. http://www.lrec-conf.org/proceedings/lrec2016/pdf/1138_Paper.pdf.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual Models of Word Embeddings: An Empirical Comparison. In *ACL*. http://aclanthology.info/papers/cross-lingual-models-of-word-embeddings-an-empirical-comparison.

Laurens Van der Maaten, Minmin Chen, Stephen Tyree, and Kilian Weinberger. 2013. Learning with Marginalized Corrupted Features. In *ICML*. http://www.jmlr.org/proceedings/papers/v28/vandermaaten13.html.

Ivan Vulić and Anna Korhonen. 2016. On the Role of Seed Lexicons in Learning Bilingual Word Embeddings. In *ACL*. http://aclanthology.info/papers/on-the-role-of-seed-lexicons-in-learning-bilingual-word-embeddings.

Ivan Vulić and Marie-Francine Moens. 2013. Cross-Lingual Semantic Similarity of Words as the Similarity of Their Semantic Word Responses. In *NAACL-HLT*. http://aclanthology.info/papers/cross-lingual-semantic-similarity-of-words-as-the-similarity-of-their-semantic-word-responses.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual Word Embeddings from Non-Parallel Document-Aligned Data Applied to Bilingual Lexicon Induction. In *ACL-IJCNLP*. http://aclanthology.info/papers/bilingual-word-embeddings-from-non-parallel-document-aligned-data-applied-to-bilingual-lexicon-induction.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2011. Identifying Word Translations from Comparable Corpora Using Latent Topic Models. In *ACL-HLT*. http://aclanthology.info/papers/identifying-word-translations-from-comparable-corpora-using-latent-topic-models.

Stefan Wager, Sida Wang, and Percy S Liang. 2013. Dropout Training as Adaptive Regularization. In *NIPS*. http://papers.nips.cc/paper/4882-dropout-training-as-adaptive-regularization.pdf.

Michael Wick, Pallika Kanani, and Adam Pocock. 2016. Minimally-Constrained Multilingual Embeddings via Artificial Code-Switching. In *AAAI*. http://www.aaai.org/Conferences/AAAI/2016/Papers/15Wick12464.pdf.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. In *NAACL-HLT*. http://aclanthology.info/papers/normalized-word-embedding-and-orthogonal-transform-for-bilingual-word-translation.

Hyejin Youn, Logan Sutton, Eric Smith, Cristopher Moore, Jon F. Wilkins, Ian Maddieson, William Croft, and Tanmoy Bhattacharya. 2016. On the universal structure of human lexical semantics. *Proceedings of the National Academy of Sciences* https://doi.org/10.1073/pnas.1520752113.

Meng Zhang, Yang Liu, Huanbo Luan, Yiqun Liu, and Maosong Sun. 2016a. Inducing Bilingual Lexica From Non-Parallel Data With Earth Mover's Distance Regularization. In *COLING*. http://aclanthology.info/papers/inducing-bilingual-lexica-from-non-parallel-data-with-earth-mover-s-distance-regularization.

Meng Zhang, Haoruo Peng, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Bilingual Lexicon Induction From Non-Parallel Data With Minimal Supervision. In *AAAI*. http://thunlp.org/˜zm/publications/aaai2017.pdf.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016b. Ten Pairs to Tag – Multilingual POS Tagging via Coarse Mapping between Embeddings. In *NAACL-HLT*. http://aclanthology.info/papers/ten-pairs-to-tag-multilingual-pos-tagging-via-coarse-mapping-between-embeddings.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *EMNLP*. http://aclanthology.info/papers/bilingual-word-embeddings-for-phrase-based-machine-translation.

Estimating Code-Switching on Twitter with a Novel Generalized Word-Level Language Detection Technique

Shruti Rijhwani*
Language Technologies Institute
Carnegie Mellon University
srijhwan@cs.cmu.edu

Royal Sequiera*
University of Waterloo
Waterloo, Canada
rdsequie@uwaterloo.ca

Monojit Choudhury **Kalika Bali** **Chandra Sekhar Maddila**
Microsoft Research
Bangalore, India
{monojitc,kalikab,chmaddil}@microsoft.com

Abstract

Word-level language detection is necessary for analyzing code-switched text, where multiple languages could be mixed within a sentence. Existing models are restricted to code-switching between two specific languages and fail in real-world scenarios as text input rarely has a priori information on the languages used. We present a novel unsupervised word-level language detection technique for code-switched text for an arbitrarily large number of languages, which does not require any manually annotated training data. Our experiments with tweets in seven languages show a 74% relative error reduction in word-level labeling with respect to competitive baselines. We then use this system to conduct a large-scale quantitative analysis of code-switching patterns on Twitter, both global as well as region-specific, with 58M tweets.

1 Introduction

In stable multilingual societies, communication often features fluid alteration between two or more languages – a phenomenon known as *code-switching*[1] (Gumperz, 1982; Myers-Scotton, 1993). It has been studied extensively in linguistics, primarily as a speech phenomenon (Poplack, 1980; Gumperz, 1982; Myers-Scotton, 1993; Milroy and Muysken, 1995; Auer, 2013). However, the growing popularity of computer mediated communication, particularly social media, has resulted in language data in the text form which exhibits code-switching, among other speech-like characteristics (Crystal, 2001; Herring, 2003; Danet and Herring, 2007; Cardenas-Claros and Isharyanti, 2009). With the large amount of online content generated by multilingual users around the globe, it becomes necessary to design techniques to analyze mixed language, which can help not only in developing end-user applications, but also in conducting fundamental sociolinguistic studies.

Language detection (LD) is a prerequisite to several NLP techniques. Most state-of-the-art LD systems detect a single language for an entire document or sentence. Such methods often fail to detect code-switching, which can occur within a sentence. In recent times, there has been some effort to build word-level LD for code-switching between a specific pair of languages (Nguyen and Dogruöz, 2013; Elfardy et al., 2013; Solorio et al., 2014; Barman et al., 2014). However, usually user-generated text (e.g., on social media) has no prior information of the languages being used. Further, as several previous social-media based studies on multilingualism have pointed out (Kim et al., 2014; Manley, 2012), lack of general word-level LD has been a bottleneck in studying code-switching patterns in multilingual societies.

This paper proposes a novel technique for word-level LD that generalizes to an arbitrarily large set of languages. The method does not require a priori information on the specific languages (potentially more than two) being mixed in an input text as long as the languages are from a fixed (arbitrarily large) set. Training is done without any manually annotated data, while achieving accuracies comparable to language-restricted systems trained

* This work was done when the authors were affiliated with Microsoft Research.

[1] This paper uses the terms 'code-switching' and 'code-mixing' interchangeably.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1971–1982
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1180

with large amounts of labeled data. With a word-level LD accuracy of 96.3% on seven languages, this technique enabled us to analyze patterns of code-switching on Twitter, which is the second key contribution of this paper. To the best of our knowledge, this is the first quantitative study of its kind, particularly at such a large-scale.

2 Related Work

In this section, we will briefly survey the language detection techniques (see Hughes et al. (2006) and Garg et al. (2014) for comprehensive surveys), and sociolinguistic studies on multilingualism (see Nguyen et al. (2016) for a detailed survey) that were enabled by these techniques.

Early work on LD (Cavnar and Trenkle, 1994; Dunning, 1994) focused on detecting a single language for an entire document. These obtained high accuracies on well-formed text (e.g., news articles), which led to LD being considered solved (McNamee, 2005). However, there has been renewed interest with the amount of user-generated content on the web. Such text poses unique challenges such as short length, misspelling, idiomatic expressions and acronyms (Carter et al., 2013; Goldszmidt et al., 2013). Xia et al. (2009), Tromp and Pechenizkiy (2011) and Lui and Baldwin (2012) created LD systems for monolingual sentences, web pages and tweets. Zhang et al. (2016) built an unsupervised model to detect the majority language in a document. There has also been document-level LD that assigns multiple language to each document (Prager, 1999; Lui et al., 2014). However, documents were synthetically generated, restricted to inter-sentential language mixing. Also, these models do not fragment the document based on language, making language-specific analysis impossible.

Document-level or sentence-level LD does not identify code-switching accurately, which can occur within a sentence. Word-level LD systems attempt to remedy this problem. Most work has been restricted to cases where two languages, known a priori, is to be detected in the input i.e, *binary LD at the word-level*. There has been work on Dutch-Turkish (Nguyen and Dogruöz, 2013), English-Bengali (Das and Gambäck, 2014) and Standard and dialectal Arabic (Elfardy et al., 2013). King and Abney (2013) address word-level LD for bilingual documents in 30 language pairs, where the language pair is known a priori. The features for word-level LD proposed by Al-Badrashiny and Diab (2016) are language-independent, however, at any given time, the model is only trained to tag a specific language pair. There have also been two shared task series on word-level LD: FIRE (Roy et al., 2013; Choudhury et al., 2014; Sequiera et al., 2015) focused on Indian languages and the EMNLP Code-Switching Workshop (Solorio et al., 2014; Molina et al., 2016). These pairwise LD methods vary from dictionary-based to completely supervised and semi-supervised. None tackle the imminent lack of annotated data required for scaling to more than one language pair.

There has been little research on word-level LD that is not restricted to two languages. Hammarström (2007) proposed a model for multilingual LD for short texts like queries. Gella et al. (2014) designed an algorithm for word-level LD across 28 languages. Jurgens et al. (2017) use an encoder-decoder architecture for word-level LD that supports dialectal variation and code-switching. However, these studies experiment with synthetically created multilingual data, constrained either by the number of language switches permitted or to phrase-level code-switching, and are not equipped to handle the challenges posed by real-world code-switching.

Using tweet-level LD systems like the CompactLanguageDetector[2], there have been studies on multilingualism in specific cities like London (Manley, 2012) and Manchester (Bailey et al., 2013). These studies, as well as Bergsma et al. (2012), observe that existing LD systems fail on code-switched text. Kim et al. (2014) studied the linguistic behavior of bilingual Twitter users from Qatar, Switzerland and Québec, and also acknowledge that code-switching could not be studied due to the absence of appropriate LD tools.

Using word-level LD for English-Hindi (Gella et al., 2013), Bali et al. 2014 observed that as much as 17% of Indian Facebook posts had code-switching, and Rudra et al. (2016) showed that the native language is strongly preferred for expressing negative sentiment by English-Hindi bilinguals on Twitter. However, without accurate multilingual word-level LD, there have been no large-scale studies on the extent and distribution of code-switching across various communities.

[2] https://www.npmjs.com/package/cld

3 Generalized Word-level LD

We present *Generalized Word-Level Language Detection*, or GWLD, where:

- The number of supported languages can be arbitrarily large
- Any number of the supported languages can be mixed within a single input
- The languages in the input do not need to be known a priori
- Any number of language switches are allowed in the input.
- No manual annotation is required for training

Formalizing our model, let $\mathbf{w} = w_{i=1...n}$ be a natural language text consisting of a sequence of words, w_1 to w_n. For our current work, we define words to be whitespace-separated tokens (details in Sec 5). Let $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$ be a set of k natural languages. We assume that each w_i can be assigned to a unique language $l_j \in \mathcal{L}$.

We also define *universal tokens* like numbers, emoticons, URLs, emails and punctuation, which do not belong to any specific natural language. Certain strings of alphabetic characters representing generic interjections or sounds, such as `oh`, `awww`, `zzz` also fall in this category. For labeling these tokens, we use an auxiliary set of labels, $\mathcal{X}_\mathcal{L} = \{xl_1, xl_2, \ldots, xl_k\}$. Labeling each universal token with a specific language l_i (using xl_i) instead of generically labeling all such tokens xl allows preserving linguistic context when a memoryless model like Hidden Markov Models (HMM) are used for tagging. Further, various NLP tasks on might require the input text, including these universal tokens, to be split by language.

For input $\mathbf{w}$, let the output from the LD system be $\mathbf{y} = y_{i=1...n}$, a sequence of labels, where $y_i \in \mathcal{L} \cup \mathcal{X}_\mathcal{L}$. $y_i = l_j$ if and only if, in the context of $\mathbf{w}$, w_i is a word from l_j. If w_i is a *universal token*, $y_i = xl_j$, when $y_{i-1} = l_j$ or $y_{i-1} = xl_j$. If w_1 is a *universal token*, $y_1 = xl_j$, where l_j is the label of the first token $\in \mathcal{L}$ in the input.

Fig. 1 shows a few examples of labeled code-switched tweets. Named entities (NE) are assigned labels according to the convention used by King and Abney (2013).

4 Method

Word-level LD is essentially a sequence labeling task. We use a Hidden Markov Model (HMM), though any other sequence labeling technique, e.g., CRFs, can be used as well.

The intuition behind the model architecture is simple – a person who is familiar with k languages can easily recognize (and also understand) the words when any of those languages are code-switched, even if s/he has never seen any mixed language text before. Analogously, *is it possible that monolingual language models, when combined, can identify code-switched text accurately?*

Imagine we have k HMMs, where the ith HMM has two states l_i and xl_i. Each state can label a word. The HMMs are independent, but they are tied to a common start state s and end state e, forming a word-level LD model for monolingual text in one of the k languages. Now, we make transitions from $l_i \rightarrow l_j$ possible, where $i \neq j$. This HMM, shown in Fig. 2, is capable of generating and consequently, labeling code-switched text between any of the k languages. The solid and dotted lines show monolingual transitions and the added code-switching transitions respectively. Fig. 2 depicts three languages, however, the number of languages can be arbitrarily large.

Obtaining word-level annotated monolingual and code-switched data is expensive and nearly infeasible for a large number of languages. Instead, we automatically create weakly-labeled monolingual text (set $\mathcal{W}$) and use it to initialize the HMM parameters. We then use Baum-Welch reestimation on unlabeled data (set $\mathcal{U}$) that has monolingual and code-switched text in their natural distribution. Sec. 5 discusses creation of $\mathcal{W}$ and $\mathcal{U}$.

4.1 Structure, Initialization and Learning

The structure of the HMM shown in Fig. 2 can be formally described using:

- Set of states, $\mathcal{S} = s \cup \mathcal{L} \cup \mathcal{X}_\mathcal{L} \cup e$
- Set of observations, $\mathcal{O}$
- Emission matrix ($|\mathcal{S}| \times |\mathcal{O}|$)
- Transition matrix ($|\mathcal{S}| \times |\mathcal{S}|$)

$\mathcal{O}$ consists of all seen events in the data, and a special symbol unk for all unseen events. We define an event as a token n-gram and we experimented with $n = 1$ to 3. It is important to mention that the n-grams do not spread over language states. We also use special start and end symbols, which are observed at states s and e respectively. Elements of $\mathcal{O}$ are effectively what the states of the HMM 'emit' or generate during decoding.

Ex(1): no\$l_2$ me\$l_2$ lebante\$l_2$ ahorita\$l_2$ cuz\$l_1$ I\$l_1$ felt\$l_1$ como\$l_2$ si\$l_2$ me\$l_2$ kemara\$l_2$ por\$l_2$ dentro\$l_2$!\$xl_2$:o\$xl_2$ Then\$l_1$ I\$l_1$ started\$l_1$ getting\$l_1$ all\$l_1$ red\$l_1$,\$xl_1$ I\$l_1$ think\$l_1$ im\$l_1$ allergic\$l_1$ a\$l_2$ algo\$l_2$

Ex(2): @XXXXX\$xl_3$ @XXXXX\$xl_3$:)\$xl_3$:)\$xl_3$:)\$xl_3$:)\$xl_3$ hahahahah\$xl_3$ alles\$l_3$ is\$l_3$ 3D\$xl_3$ voor\$l_3$ mama\$l_4$ hatta\$l_4$ 4D\$xl_4$:P\$xl_4$:P\$xl_4$:P\$xl_4$:P\$xl_4$ Havva\$l_4$ &\$xl_4$ Yusuf\$l_4$ olunca\$l_4$ misafir\$l_4$ fln\$l_4$ dinlemez\$l_4$!!\$xl_4$

Figure 1: Examples of code-switched tweets and the corresponding language labels. l_1 = English, l_2 = Spanish, l_3 = Dutch, l_4 = Turkish. Usernames have been anonymized.

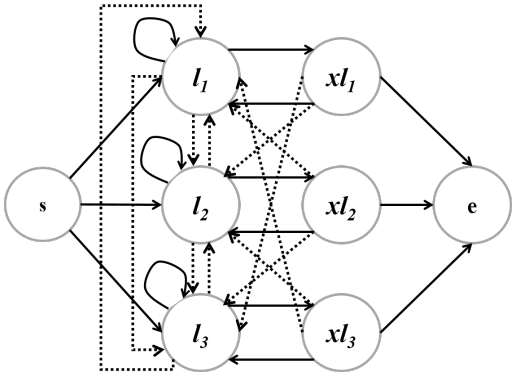


Figure 2: GWLD Hidden Markov Model. $s \rightarrow xl_i$ and $l_i \rightarrow e$ transitions omitted for clarity.

For any input, the HMM always starts in the state s. The parameters to be learned are the transition and emission matrices.

We initialize these matrices using $\mathcal{W}$. The trigram, bigram and unigram word counts from the data for each language in $\mathcal{W}$ are used to create language models (LM) with modified Kneser-Ney smoothing (Chen and Goodman, 1999). The emission values for state l_i are initialized with the respective LM probabilities for all seen n-grams. We also assign a small probability to unk. The emissions for the xl_i state are initialized using the counts of *universal tokens* for the language l_i in $\mathcal{W}$. These are identified using the preprocessing techniques discussed in Sec. 5.1.

Possible transitions for each monolingual HMM are $l_i \rightarrow l_i$, $l_i \rightarrow xl_i$ and $xl_i \rightarrow l_i$. We do not have the $xl_i \rightarrow xl_i$ transition, because preprocessing (Sec. 5.1) concatenates successive *universal tokens* into a single token. This does not change the output as the tokens can easily be separated after LD, but is a useful simplification for the model. The transition values for l_i are initialized by the probability of transitions between words and *universal tokens* in the text from $\mathcal{W}$.

As stated earlier, the model supports code-switching by the addition of transitions $l_i \rightarrow l_j$, and $xl_i \rightarrow l_j$, for all $i \neq j$. For each state l_i, there are $2k - 2$ new transitions (Fig. 2). We initialize these news edges with a small probability π, before normalizing transitions for each state. π, which we call the code-switch probability, is a hyperparameter tuned on a validation set.

Starting with the initialized matrices, we reestimate the transition and emission matrices using the EM-like *Baum-Welch* algorithm (Welch, 2003) over the large set of unlabeled text $\mathcal{U}$.

4.2 Decoding

The input to the trained model is first preprocessed as described in Sec. 5.1 (tokenization and identification of *universal tokens*). The Viterbi algorithm is then used with the HMM parameters to perform word-level LD. When an unknown n-gram, is encountered, its emission probability is estimated by recursively backing off to $(n-1)$-gram, until we find a known n-gram. If the unigram, i.e., the token, is also unknown, then the observation of the symbol unk is used instead.

5 Dataset Creation

The data for both training and testing comes primarily from Twitter because of its public API, and studies have shown the presence of code-switching in social media (Crystal, 2001; Herring, 2003; Danet and Herring, 2007; Cardenas-Claros and Isharyanti, 2009; Bali et al., 2014).

Our experiments use monolingual and code-switched tweets in seven languages – Dutch (*nl*), English (*en*), French (*fr*), German (*de*), Portuguese (*pt*), Spanish (*es*) and Turkish (*tr*). These form the set $\mathcal{L}$. The choice of languages is motivated by several factors. First, LD is non-trivial as all these languages use the Latin script. Second, a large volume of tweets are generated in these languages.

Third, there is annotated code-switched data available in *nl-tr* and *en-es*, which can be used for validation and testing. Lastly, we know that certain pairs of these languages are code-switched often.

5.1 Collection and Preprocessing

Using the Twitter API (Twitter, 2013), we collected tweets over May-July 2015. We selected tweets identified by Twitter LD API (Twitter, 2015) as one of the languages in $\mathcal{L}$. We also removed non-Latin script tweets.

As preprocessing, each tweet is first tokenized using *ark-twitter* (Gimpel et al., 2011) and URLs, hashtags and user mentions are identified using regular expressions. We also identify emoticons, punctuation, digits, special characters, and some universal interjections and abbreviations (such as `RT`, `aww`) as *universal tokens*. We use an existing dictionary (Chittaranjan et al., 2014) for the latter. Let the set of tweets after preprocessing be $\mathcal{T}$.

5.2 Sets $\mathcal{W}$ and $\mathcal{U}$

We use the CoverSet algorithm (Gella et al., 2014) on each tweet in $\mathcal{T}$. It obtains a confidence score for a word w_i belonging to a language l_j using a Naive Bayes classifier trained on Wikipedia. These scores are used to find the minimal set of languages are required to label all the input words. If CoverSet detects the tweet as monolingual (i.e., one language can label all words) and the identified language is the same as the Twitter LD label, the tweet is added to the weakly-labeled set $\mathcal{W}$. These tweets are almost certainly monolingual, as CoverSet has very high recall (and low precision) for detecting code-switching. As these are not manually labeled, we call them *weakly-labeled*. $\mathcal{W}$ contains 100K tweets in each language (700K in total).

From $\mathcal{T}$, we randomly select 100K tweets in each of the seven languages based on the Twitter LD API labels. These tweets do not have word-level language labels and may be code-switched or have an incorrect Twitter language label. We use these as unlabeled data, the set $\mathcal{U}$.

5.3 Validation and Test Sets

We curate two word-level gold-standard datasets for validation and testing. These sets contain monolingual tweets in each of the seven languages as well as code-switched tweets from certain language pairs, based on the availability of real-world data. However, it must be noted that GWLD can detect code-switching between more than two languages. The language-wise distribution is shown in Table 1. Including *universal tokens*, the validation and test set contain 33981 and 58221 tokens respectively. The annotated tweets will be made available for public use.

L1-L2	Tweets	L1 Tokens	L2 Tokens
nl	100 (100)	965 (1099)	–
fr	100 (102)	1085 (1045)	–
pt	100 (100)	1080 (967)	–
de	101 (100)	1078 (890)	–
tr	100 (100)	939 (879)	–
es	100 (100)	1067 (1119)	–
en	100 (100)	1161 (1006)	–
nl-en	65 (50)	498 (436)	243 (174)
fr-en	50 (48)	428 (370)	224 (227)
pt-en	53 (53)	463 (513)	278 (242)
de-en	49 (50)	417 (459)	293 (292)
tr-en	50 (50)	347 (336)	238 (209)
es-en	3013 (52)	8510 (355)	16356 (395)
nl-tr	735 (728)	5895 (8590)	5293 (8140)

Table 1: Test Set Statistics (Validation Set in parentheses). Rows in gray show existing datasets.

For *es-en*, we use the word-level annotated test set from the code-switching shared task on language detection (Solorio et al., 2014). We ignore the tokens labeled NE, Ambiguous and Mixed during our system evaluation (Sec. 6), as they do not fall in the scope of this work. The words labeled 'Other' were marked as xl_i where l_i is *en* or *es*, based on the context. We also use existing *nl-tr* validation and test sets (Nguyen and Dogruöz, 2013), which contain posts from a web forum.

For the other language pairs, we created our own validation and test sets, as none already exist. We randomly selected tweets for which CoverSet identified code-switching with high confidence. We gave 215 of these to six annotators for word-level annotation. It is difficult to find annotators who know all seven languages; elaborate guidelines were provided on using online machine translation, dictionaries and search engines for the task. Four out of the six annotators had high inter-annotator agreement – the agreement on $L1$ (language that the majority of the words in the tweet belong to) was 0.93, $L2$ (the other language, whenever present) was 0.8 and whether the tweet is code-switched was 0.84. We did not find any instances of code-switching between more than two

Systems	*Acc*	L_1L_2Acc	*IsMix*
Dictionary-based Baselines			
MAXFREQ	0.824	0.752	0.600
MINCOVER	0.853	0.818	0.733
Existing Systems			
LINGUINI	NA	0.529	0.783
LANGID	NA	0.830	0.783
POLYGLOT	NA	0.521	0.692
GWLD: The Proposed Method			
Initial	0.838	0.825	0.837
Reestimated	**0.963**	**0.914**	**0.88**

Table 2: Performance of LD Systems on Test Set

languages, which is rare in general. We distributed 3000 tweets between the four annotators (monolingual and code-switched tweets from COVERSET). Disagreements were settled between the annotators and a linguist. A subset of the annotated tweets form the validation and test sets (Table 1), and were removed from $\mathcal{W}$ and $\mathcal{U}$.

6 Experiments and Results

We compare GWLD with three existing systems: LINGUINI (Prager, 1999), LANGID (Lui and Baldwin, 2012), and POLYGLOT (Lui et al., 2014). None of these perform word-level LD, however, LANGID and POLYGLOT return a list of languages with confidence scores for the input. Since code-switching with more than two languages is absent in our dataset, we consider up to two language labels. We define the tweet to be monolingual if the difference between the confidence values for the top two languages is greater than a parameter δ. Otherwise, it is assumed to be code-switched with the top two languages. δ is tuned independently for the two LD systems on the validation set by maximizing the metric L_1L_2 *Accuracy* (Sec. 6.2). Inspired by Gella et al. (2013), we also compare with dictionary-based word-level LD baselines.

6.1 Dictionary-based Baselines

For each language, we build a lexicon of all the words and their frequencies found in $\mathcal{W}$ for that language. Let the lexicon for language $l_i \in \mathcal{L}$ be lex_i. Let $f(lex_i, w_j)$ be the frequency of w_j in lex_i. We define the following baselines:

MAXFREQ: For each w_j in $\mathbf{w}$, MAXFREQ returns lex_i that has the maximum frequency for that token. Therefore, the language label for w_j is $y_j = l_{[\arg\max_i f(lex_i, w_j)]}$. If the token is not found in any lexicon, y_j is assigned the value of y_{j-1}.

MINCOVER: We find the smallest subset $mincov(\mathbf{w}) \subset \mathcal{L}$, such that for all w_j in input $\mathbf{w}$, we have at least one language $l_i \in mincov(\mathbf{w})$ with $f(lex_i, w_j) > 0$. If there is no such language, then w_j is not considered while computing $mincov(\mathbf{w})$. Once $mincov(\mathbf{w})$ is obtained, labels y_i are computed using the MAXFREQ strategy, where the set of languages is restricted to $mincov(\mathbf{w})$ instead of $\mathcal{L}$. Note that $mincov(\mathbf{w})$ need not be unique for $\mathbf{w}$; in such cases, we choose the $mincov(\mathbf{w})$ which maximizes the sum of lexical frequencies based on MAXFREQ labels.

6.2 Metrics

We define the *Accuracy* (*Acc*) of an LD system as the fraction of words in the test set that are labeled correctly. Since the existing LD systems do not label languages at word-level, we also define:

IsMix is the fraction of tweets that are correctly identified as either monolingual or code-mixed.

L_1L_2 *Accuracy* (L_1L_2Acc) is the mean accuracy of detecting language(s) at tweet-level. For monolingual tweets, this accuracy is 1 if the gold standard label is detected by the LD system, else 0. For code-switched tweets, the accuracy is 1 if both languages are detected, 0.5 if one language is detected, and 0 otherwise. L_1L_2Acc is the average over all test set tweets.

6.3 Results

We use these metrics to assess performance on the test set for the baselines, existing LD systems and GWLD (Table 2). *Initial* refers to the HMM model estimated from $\mathcal{W}$ and *Reestimated* refers to the final model after Baum-Welch reestimation. The parameter π is tuned on the validation set using grid search. *Reestimated* GWLD has the best accuracy of 0.963 and performs significantly better than all the other systems for all metrics. Reesimatation improves the word-level *Acc* for L1 from 0.89 to 0.97 and for L2 from 0.43 to 0.82. LINGUINI and POLYGLOT likely have low L_1L_2Acc because they are trained on synthetically-created documents with no word-level code-switching.

Since our test set contains pre-existing annotations for *en-es* (Solorio et al., 2014) and *nl-tr* (Nguyen and Dogruöz, 2013), we compare with state-of-the-art results on those datasets. On *en-es* tokens, Al-Badrashiny and Diab (2016) reports an $F1$-score of 0.964; GWLD obtains 0.978. Nguyen and Dogruöz (2013) report 0.976 *Acc* on the *nl-tr*

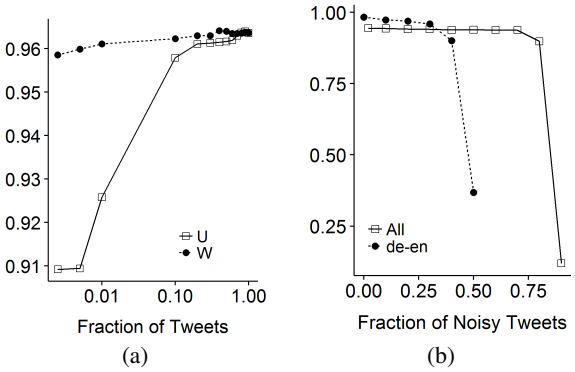


Figure 3: Acc versus Dataset Parameters

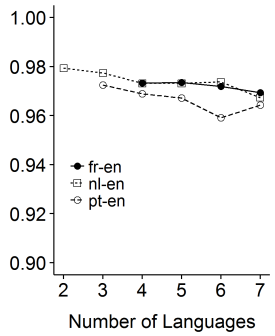


Figure 4: Acc versus Number of Languages

test set. We obtain a less competitive 0.936. However, when errors between *nl-en* are ignored as most of these are *en* words with *nl* gold-standard labels (convention followed by the dataset creators), the revised Acc is 0.963. Notably, unlike GWLD, both these models use large amounts of annotated data for training and are restricted to detecting only two languages.

Error Analysis: GWLD sometimes detects languages that are not present in the tweet, which account for a sizable fraction (39%) of all word-level errors. Not detecting a language switch causes 8% of the errors. Most other errors are caused by named entities, single-letter tokens, unseen words and the *nl-en* annotation convention in the test set from Nguyen and Dogruöz (2013).

6.4 Robustness of GWLD

We test the robustness of GWLD by varying the size of the weakly-labeled set, the unlabeled dataset and the number of languages the model is trained to support.

6.4.1 Size of $\mathcal{W}$ and $\mathcal{U}$

The variation of Acc with the size of $\mathcal{W}$ is shown in Figure 3a. Even with 0.25% of the set (250 tweets for each $l_i \in \mathcal{L}$), the model has accuracy of nearly 0.96. A slow rise in accuracy is observed as the number of tweets in $\mathcal{W}$ is increased. We also experiment with varying the size of $\mathcal{U}$. In Figure 3a, we see that with 0.25% of U (around 1,400 randomly sampled tweets), the accuracy on the test set is lower than 0.91. This quickly increases with 10% of $\mathcal{U}$. Thus, GWLD achieves Acc comparable to existing systems with very little weakly-labeled data (just 250 tweets per language, which are easily procurable for most languages) and around 50,000 unlabeled tweets.

L1-L2	Acc	$IsCM$	GWLD-Acc
nl-en	0.979	0.943	0.967
fr-en	0.982	0.948	0.969
pt-en	0.977	0.952	0.964
de-en	0.984	0.956	0.975
tr-en	0.985	0.984	0.983
es-en	0.954	0.929	0.978
nl-tr	0.975	0.907	0.936

Table 3: Statistics for Pairwise (col. 2 and 3) and GWLD Systems

6.4.2 Noise in $\mathcal{W}$

Since a small, but pure, $\mathcal{W}$ gives high accuracy (Sec. 6.4.1), we evaluate how artificially introduced noise affects Acc. The noise introduced into the $\mathcal{W}$ of each language comes uniformly from the other six languages. Figure 3b shows how increasing fractions of noise slowly degrades accuracy, with a steep drop to 0.11 accuracy at 90% noise, where the tweets from each incorrect language outnumber the correct language tweets. We test this with a pairwise model as well, as noise from a single language might have greater effect. The accuracy falls to 0.36 at 50% noise (Fig. 3b). At this point, $\mathcal{W}$ has an equal number of tweets from each language and is essentially useless.

6.4.3 Number of languages

Pairwise Models: Table 3 details two performance metrics (defined in Sec. 5.2) for our model trained on only two languages and the corresponding 7-language GWLD Acc for that language pair.

Incremental Addition of Languages: We test Acc while incrementally adding languages to the model in a random order (*nl-en-pt-fr-de-es-tr*). Figure 4 shows the variation in Acc for *nl-en*, *pt-en* and *fr-en* as more languages are added to the

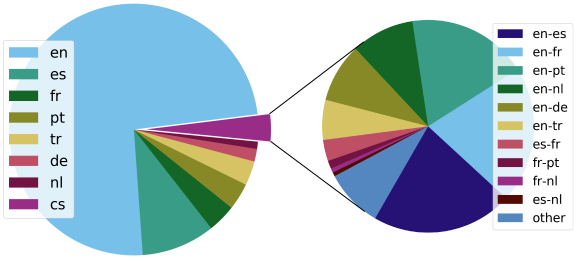


Figure 5: Worldwide distribution of monolingual and CS tweets (left and right charts respectively)

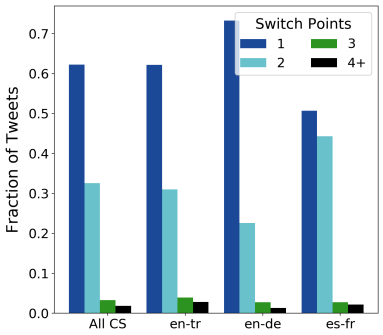


Figure 6: Worldwide CS point distribution

model. Although there is a slight degradation, in absolute terms, the accuracy remains very high.

7 Code-Switching on Twitter

The high accuracy and fast processing speed (the current multithreaded implementation labels 2.5M tweets per hour) of GWLD enables us to conduct large-scale and reliable studies of CS patterns on Twitter for the 7 languages. In this paper, we conduct two such studies. The first study analyzes 50M tweets from across the world to understand the extent and broad patterns of switching among these languages. In the second study, we analyze 8M tweets from 24 cities to gain insights into geography-specific CS patterns.

7.1 Worldwide Code-Switching Trends

We collected 50 million unique tweets that were identified by the Twitter LD API as one of the 7 languages. We place this constraint to avoid tweets from unsupported languages during analysis. Figure 5 shows the overall language distribution, including the CS language-pair distribution. Approximately 96.5% of the tweets are monolingual, a majority of which are *en* (74%).

Around 3.5% of all tweets are code-switched. Globally, *en-es*, *en-fr* and *en-pt* are the three most commonly mixed pairs accounting for 21.5%, 20.8% and 18.4% of all CS tweets in our data respectively. Interestingly, 85.4% of the CS tweets have *en* as one of the languages; *fr* is the next most popularly mixed language, with *fr-es* (3.2%), *fr-pt* (1.2%) and *fr-nl* (0.6%) as the top three observed pairs. Although around 1% of CS tweets were detected as containing more than two languages, these likely have low precision because of language overdetection as discussed in Sec. 6.3.

Figure 6 shows the fraction of *code-switch points*, i.e., how many times the language changes in a CS tweet, for all the languages, as well as for three language pairs with to highlight different trends. Most CS tweets have one CS-point, which implies that the tweet begins with one language, and then ends with another. Such tweets are very frequent for *en-de* where we observe that usually the tweets state the same fact in both *en* and *de*. This so-called *translation function* (Begum et al., 2016) of CS is probably adopted for reaching out to a wider and global audience. In contrast, *es-fr* tweets have fewer tweets with single and far more with two CS-point than average. Tweets with two CS-points typically imply the inclusion of a short phrase or chunk from another language. *en-tr* tweets have the highest number of CS-points, implying rampant and fluid switching between the two languages at all structural levels.

7.2 City-Specific Code-Switching Trends

Cosmopolitan cities are melting pots of cultures, which make them excellent locations for studying multilingualism and language interaction, including CS (Bailey et al., 2013). We collected tweets from 24 populous and highly cosmopolitan cities from Europe, North America and South America, where the primarily spoken language is one of the 7 languages detectable by GWLD. Around 8M tweets were collected from these cities.

Table 4 shows the top and bottom 6 cities, ranked by the fraction of CS tweets from that city. The total number of tweets analyzed and the top two CS pairs, along with their fractions (of CS tweets from that city) are also reported. More details can be found in the supplementary material. It is interesting to note that the 6 cities with lowest CS tweet fractions have *en* as the major language, whereas the 6 cities with highest CS fractions are from non-English (Turkish, Spanish and French) speaking geographies. In fact, the Pearson's cor-

Cities with highest fraction of CS tweet			Cities with lowest fraction of CS tweets		
City	Tweets	CS-fraction (CS pairs)	City	Tweets	CS-fraction (CS pairs)
Istanbul	351K	.12 (*en-tr* .53, *nl-tr* .13)	Houston	588K	.01 (*en-es* .22, *en-fr* .21)
Québec City	108K	.08 (*en-fr* .45, *es-fr* .23)	San Francisco	532K	.02 (*en-es* .26, *en-fr* .19)
Paris	158K	.07 (*en-fr* .43, *fr-pt* .21)	NYC	690K	.02 (*en-es* .21, *en-fr* .19)
Mexico City	332K	.07 (*en-es* .54, *es-fr* .14)	Miami	290K	.02 (*en-es* .33, *en-pt* .20)
Brussels	100K	.06 (*en-fr* .37, *es-fr* .15)	London	492K	.02 (*en-fr* .26, *en-pt* .17)
Madrid	147K	.06 (*en-es* .43, *es-fr* .32)	San Diego	432K	.02 (*en-es* .29, *en-fr* .14)

Table 4: Top (left) and bottom (right) six cities according to the fraction of CS tweets.

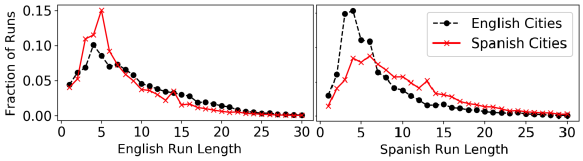


Figure 7: *en-es* Run Length

relation between the fraction of monolingual English tweets and CS tweets for these 24 cities is -0.85. Further, from Table 4 one can also observe that for non-English speaking geographies, the majority language is most commonly mixed with English, followed by French (Spanish, if French is the majority language). Istanbul is an exception, where Dutch is the second most commonly mixed language with Turkish, presumably because of the large Turkish immigrant population in Netherlands resulting in a sizeable Turkish-Dutch bilingual diaspora (Doğruöz and Backus, 2009; Nguyen and Dogruöz, 2013).

Is there a difference in the way speakers mix a pair of languages, say *en* and *es*, in *en*-speaking goegraphies like San Diego, Miami, Houston and New York City, and *es*-speaking geographies like Madrid, Barcelona, Buenos Aires and Mexico City? Indeed, as shown in Fig. 7, the distribution of the lengths of *en* and *es* runs (contiguous sequence of words in a single language beginning and ending with either a CS-point or beginning/end of a tweet) in *en-es* CS tweets is significantly different in *en*-speaking and *es*-speaking geographies. *en* runs are longer in *en*-speaking cities and vice versa, showing that the second language is likely used in short phrases.

8 Conclusion and Future Work

We present GWLD, a system for word-level language detection for an arbitrarily large set of languages that is completely unsupervised. Our results on monolingual and code-switched tweets in seven Latin script languages show a high 0.963 accuracy, significantly out-performing existing systems. Using GWLD, we conducted a large-scale study of CS trends among these languages, both globally and in specific cities.

One of the primary observations of this study is that while code-switching on Twitter is common worldwide (3.5%), it is much more common in non-English speaking cities like Istanbul (12%) where 90% of the population speak Turkish. On the other hand, while a third of the population of Houston speaks Spanish and almost everybody English, only 1% of the tweets from the city are code-switched. All the trends indicate a global dominance of English, which might be because Twitter is primarily a medium for broadcast, and English tweets have a wider audience. Bergsma et al. (2012) show that "[On Twitter] bilinguals bridge between monolinguals with English as a hub, while monolinguals tend not to directly follow each other." Androutsopoulos (2006) argues that due to linguistic non-homogenity of online public spaces, languages like *en*, *fr* and *de* are typically preferred for communication, even though in private spaces, "bilingual talk" differs considerably in terms of distribution and CS patterns.

As future directions, we plan to extend GWLD to several other languages and conduct similar sociolinguistic studies on CS patterns including not only more languages and geographies, but also other aspects like topic and sentiment.

Acknowledgments

We would like to thank Prof. Shambavi Pradeep and her students from BMS College of Engineering for assisting with data annotation. We are also grateful to Ashutosh Baheti and Silvana Hartmann from Microsoft Research (Bangalore, India) for help with data organization and error analysis.

References

Mohamed Al-Badrashiny and Mona Diab. 2016. Lili: A simple language independent approach for language identification. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING). Osaka, Japan*.

Jannis Androutsopoulos. 2006. Multilingualism, diaspora, and the internet: Codes and identities on german-based diaspora websites. *Journal of Sociolinguistics* 10(4):520–547.

Peter Auer. 2013. *Code-switching in conversation: Language, interaction and identity*. Routledge.

George Bailey, Joseph Goggins, and Thomas Ingham. 2013. What can Twitter tell us about the language diversity of Greater Manchester? In *Report by Multilingual Manchester*. School of Languages, Linguistics and Cultures at the University of Manchester. http://bit.ly/2kG42Qf.

Kalika Bali, Yogarshi Vyas, Jatin Sharma, and Monojit Choudhury. 2014. "I am borrowing ya mixing?" an analysis of English-Hindi code mixing in Facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Rafiya Begum, Kalika Bali, Monojit Choudhury, Koustav Rudra, and Niloy Ganguly. 2016. Functions of code-switching in tweets: An annotation framework and some initial experiments. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*.

Shane Bergsma, Paul McNamee, Mossaab Bagdouri, Clayton Fink, and Theresa Wilson. 2012. Language identification for creating language-specific twitter collections. In *Proceedings of the second workshop on language in social media*. Association for Computational Linguistics.

Mónica Stella Cardenas-Claros and Neny Isharyanti. 2009. Code-switching and code-mixing in internet chatting: Between yes, ya, and si a case study. In *The JALT CALL Journal, 5*.

Simon Carter, Wouter Weerkamp, and Manos Tsagkias. 2013. Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation Journal* 47:195–215.

William B Cavnar and John M Trenkle. 1994. N-gram-based text categorization .

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language* 13(4):359–393.

Gokul Chittaranjan, Yogrshi Vyas, Kalika Bali, and Monojit Choudhury. 2014. Word-level language identication using crf : Code-switching shared task report of msr india system. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Monojit Choudhury, Gokul Chittaranjan, Parth Gupta, and Amitava Das. 2014. Overview of FIRE 2014 track on transliterated search .

David Crystal. 2001. *Language and the Internet*. Cambridge University Press.

Brenda Danet and Susan Herring. 2007. *The Multilingual Internet: Language, Culture, and Communication Online*. Oxford University Press., New York.

Amitava Das and Bjorn Gambäck. 2014. Identifying languages at the word level in code-mixed indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*. Goa, India, pages 169–178.

A Seza Doğruöz and Ad Backus. 2009. Innovative constructions in dutch turkish: An assessment of ongoing contact-induced change. *Bilingualism: language and cognition* 12(01):41–63.

Ted Dunning. 1994. *Statistical identification of language*. Computing Research Laboratory, New Mexico State University.

Heba Elfardy, Mohamed Al-Badrashiny, and Mona Diab. 2013. Code switch point detection in arabic. In *Natural Language Processing and Information Systems*, Springer, pages 412–416.

Archana Garg, Vishal Gupta, and Manish Jindal. 2014. A survey of language identification techniques and applications. *Journal of Emerging Technologies in Web Intelligence* 6(4):388–400.

Spandana Gella, Kalika Bali, and Monojit Choudhury. 2014. "ye word kis lang ka hai bhai?" testing the limits of word level language identification. In *NLPAI*.

Spandana Gella, Jatin Sharma, and Kalika Bali. 2013. Query word labeling and back transliteration for indian languages: Shared task system description .

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and A. Noah Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Moises Goldszmidt, Marc Najork, and Stelios Paparizos. 2013. Boot-strapping language identifiers for short colloquial postings. In *Machine Learning and Knowledge Discovery in Databases*, volume 8189 of *Lecture Notes in Computer Science*, pages 95–111.

John. J. Gumperz. 1982. *Discourse strategies*. Cambridge University Press, Cambridge.

Harald Hammarström. 2007. A fine-grained model for language identification. In *In Workshop of Improving Non English Web Searching. Proceedings of iNEWS 2007 Workshop at SIGIR*.

Susan Herring, editor. 2003. *Media and Language Change*. Special issue of Journal of Historical Pragmatics 4:1.

Baden Hughes, Timothy Baldwin, SG Bird, Jeremy Nicholson, and Andrew MacKinlay. 2006. Reconsidering language identification for written language resources .

David Jurgens, Yulia Tsvetkov, and Dan Jurafsky. 2017. Incorporating dialectal variability for socially equitable language identification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*. Vancouver, Canada.

Suin Kim, Ingmar Weber, Li Wei, and Alice Oh. 2014. Sociolinguistic analysis of twitter in multilingual societies. In *Proceedings of the 25th ACM conference on Hypertext and social media*.

Ben King and Steven Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proceedings of NAACL-HLT*. pages 1110–1119.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *In Proceedings of the ACL 2012 System Demonstrations*. pages 25–30.

Marco Lui, Jey Han Lau, and Timothy Baldwin. 2014. Automatic detection and language identification of multilingual documents. In *Transactions of the Association for Computational Linguistics*.

Ed Manley. 2012. Detecting languages in Londons Twittersphere. In *Blog post: Urban Movements*. http://bit.ly/2kBytHm.

P. McNamee. 2005. Language identification: A solved problem suitable for undergraduate instruction. *Journal of Computing Sciences in Colleges* 20.

Lesley Milroy and Pieter Muysken. 1995. *One speaker, two languages: Cross-disciplinary perspectives on code-switching*. Cambridge University Press.

Giovanni Molina, Nicolas Rey-Villamizar, Thamar Solorio, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, and Mona Diab. 2016. Overview for the second shared task on language identification in code-switched data. *EMNLP 2016* page 40.

Carol Myers-Scotton. 1993. *Dueling Languages: Grammatical Structure in Code-Switching*. Claredon, Oxford.

Dong Nguyen and A. Seza Dogruöz. 2013. Word level language identification in online multilingual communication. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Dong Nguyen, A Seza Doğruöz, Carolyn P Rosé, and Franciska de Jong. 2016. Computational sociolinguistics: A survey. *Computational Linguistics* .

Shana Poplack. 1980. Sometimes Ill start a sentence in Spanish y termino en espaol. *Linguistics* 18:581–618.

John M Prager. 1999. Language identification for multilingual documents. In *Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference*.

Rishiraj Saha Roy, Monojit Choudhury, Prasenjit Majumder, and Komal Agarwal. 2013. Overview and datasets of FIRE 2013 track on transliterated search. In *Working Notes of FIRE*.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do Hindi-English speakers do on Twitter? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Royal Sequiera, Monojit Choudhury, Parth Gupta, Paolo Rosso, Shubham Kumar, Somnath Banerjee, Sudip Kumar Naskar, Sivaji Bandyopadhyay, Gokul Chittaranjan, Amitava Das, and Kunal Chakma. 2015. Overview of fire-2015 shared task on mixed script information retrieval. In *Working Notes of FIRE*.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Gohneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, et al. 2014. Overview for the first shared task on language identification in code-switched data. *Proceedings of The First Workshop on Computational Approaches to Code Switching* .

Erik Tromp and Mykola Pechenizkiy. 2011. Graph-based n-gram language identification on short texts. In *In Proc. 20th Machine Learning conference of Belgium and The Netherlands*. pages 27–34.

Twitter. 2013. GET statuses/sample — Twitter Developers. https://dev.twitter.com/docs/api/1/get/statuses/sample.

Twitter. 2015. GET help/languages — Twitter Developers. https://dev.twitter.com/rest/reference/get/help/languages.

Lloyd R Welch. 2003. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter* 53(4):10–13.

Fei Xia, William D Lewis, and Hoifung Poon. 2009. Language id in the context of harvesting language data off the web. In *In Proceedings of the 12th EACL*. pages 870–878.

Wei Zhang, Robert AJ Clark, Yongyuan Wang, and Wen Li. 2016. Unsupervised language identification based on latent dirichlet allocation. *Computer Speech & Language* 39:47–66.

Using Global Constraints and Reranking to Improve Cognates Detection

Michael Bloodgood
Department of Computer Science
The College of New Jersey
Ewing, NJ 08628
mbloodgood@tcnj.edu

Benjamin Strauss
Computer Science and Engineering Dept.
The Ohio State University
Columbus, OH 43210
strauss.105@osu.edu

Abstract

Global constraints and reranking have not been used in cognates detection research to date. We propose methods for using global constraints by performing rescoring of the score matrices produced by state of the art cognates detection systems. Using global constraints to perform rescoring is complementary to state of the art methods for performing cognates detection and results in significant performance improvements beyond current state of the art performance on publicly available datasets with different language pairs and various conditions such as different levels of baseline state of the art performance and different data size conditions, including with more realistic large data size conditions than have been evaluated with in the past.

1 Introduction

This paper presents an effective method for using global constraints to improve performance for cognates detection. Cognates detection is the task of identifying words across languages that have a common origin. Automatic cognates detection is important to linguists because cognates are needed to determine how languages evolved. Cognates are used for protolanguage reconstruction (Hall and Klein, 2011; Bouchard-Côté et al., 2013). Cognates are important for cross-language dictionary look-up and can also improve the quality of machine translation, word alignment, and bilingual lexicon induction (Simard et al., 1993; Kondrak et al., 2003).

A word is traditionally only considered cognate with another if both words proceed from the same ancestor. Nonetheless, in line with the conventions of previous research in computational linguistics, we set a broader definition. We use the word 'cognate' to denote, as in (Kondrak, 2001): "...words in different languages that are similar in form and meaning, without making a distinction between borrowed and genetically related words; for example, English 'sprint' and the Japanese borrowing 'supurinto' are considered cognate, even though these two languages are unrelated." These broader criteria are motivated by the ways scientists develop and use cognate identification algorithms in natural language processing (NLP) systems. For cross-lingual applications, the advantage of such technology is the ability to identify words for which similarity in meaning can be accurately inferred from similarity in form; it does not matter if the similarity in form is from strict genetic relationship or later borrowing (Mericli and Bloodgood, 2012).

Cognates detection has received a lot of attention in the literature. The research of the use of statistical learning methods to build systems that can automatically perform cognates detection has yielded many interesting and creative approaches for gaining traction on this challenging task. Currently, the highest-performing state of the art systems detect cognates based on the combination of multiple sources of information. Some of the most indicative sources of information discovered to date are word context information, phonetic information, word frequency information, temporal information in the form of word frequency distributions across parallel time periods, and word burstiness information. See section 3 for fuller explanations of each of these sources of information that state of the art systems currently use. Scores for all pairs of words from language L1 x language L2 are generated by generating component scores based on these sources of information and then combining them in an appropriate manner. Simple methods of combination are giving equal weight-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1983–1992
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1181

ing for each score, while state of the art performance is obtained by learning an optimal set of weights from a small seed set of known cognates. Once the full matrix of scores is generated, the word pairs with the highest scores are predicted as being cognates.

The methods we propose in the current paper consume as input the final score matrix that state of the art methods create. We test if our methods can improve performance by generating new rescored matrices by rescoring all of the pairs of words by taking into account global constraints that apply to cognates detection. Thus, our methods are complementary to previous methods for creating cognates detection systems. Using global constraints and performing rescoring to improve cognates detection has not been explored yet. We find that rescoring based on global constraints improves performance significantly beyond current state of the art levels.

The cognates detection task is an interesting task to apply our methods to for a few reasons:

- It's a challenging unsolved task where ongoing research is frequently reported in the literature trying to improve performance;
- There is significant room for improvement in performance;
- It has a global structure in its output classifications since if a word lemma[1] w_i from language L1 is cognate with a word lemma w_j from language L2, then w_i is not cognate with any other word lemma from L2 different from w_j and w_j is not cognate with any other word lemma w_k from L1.
- There are multiple standard datasets freely and publicly available that have been worked on with which to compare results.
- Different datasets and language pairs yield initial score matrices with very different qualities. Some of the score matrices built using the existing state of the art best approaches yield performance that is quite low (11-point interpolated average precision of only approximately 16%) while some of these score matrices for other language pairs and data sets have state of the art score matrices that are already able to achieve 11-point interpolated average precision of 57%.

Although we are not aware of work using global constraints to perform rescoring to improve cognates detection, there are related methodologies for reranking in different settings. Methodologically related work includes past work in structured prediction and reranking (Collins, 2002; Collins and Roark, 2004; Collins and Koo, 2005; Taskar et al., 2005a,b). Note that in these past works, there are many instances with structured outputs that can be used as training data to learn a structured prediction model. For example, a seminal application in the past was using online training with structured perceptrons to learn improved systems for performing various syntactic analyses and tagging of sentences such as POS tagging and base noun phrase chunking (Collins, 2002). Note that in those settings the unit at which there are structural constraints is a sentence. Also note that there are many sentences available so that online training methods such as discriminative training of structured perceptrons can be used to learn structured predictors effectively in those settings. In contrast, for the cognates setting the unit at which there are structural constraints is the entire set of cognates for a language pair and there is only one such unit in existence (for a given language pair). We call this a single overarching global structure to make the distinction clear. The method we present in this paper deals with a single overarching global structure on the predictions of all instances in the *entire* problem space for a task. For this type of setting, there is only a *single* global structure in existence, contrasted with the situation of there being many sentences each imposing a global structure on the tagging decisions for that individual sentence. Hence, previous structured prediction methods that require numerous instances each having a structured output on which to train parameters via methods such as perceptron training are inapplicable to the cognates setting. In this paper we present methods for rescoring effectively in settings with a single overarching global structure and show their applicability to improving the performance of cognates detection. Still, we note that philosophically our method builds on previous structured prediction methods since in both cases there is a similar intuition in that we're

[1] A lemma is a base form of a word. For example, in English the words 'baked' and 'baking' would both map to the lemma 'bake'. Lemmatizing software exists for many languages and lemmatization is a standard preprocessing task conducted before cognates detection.

using higher-level structural properties to inform and accordingly alter our system's predictions of values for subitems within a structure.

In section 2 we present our methods for performing rescoring of matrices based on global constraints such as those that apply for cognates detection. The key intuition behind our approach is that the scoring of word pairs for cognateness ought not be made independently as is currently done, but rather that global constraints ought to be taken into account to inform and potentially alter system scores for word pairs based on the scores of other word pairs. In section 3 we provide results of experiments testing the proposed methods on the cognates detection task on multiple datasets with multiple language pairs under multiple conditions. We show that the new methods complement and effectively improve performance over state of the art performance achieved by combining the major research breakthroughs that have taken place in cognates detection research to date. Complete precision-recall curves are provided that show the full range of performance improvements over the current state of the art that are achieved. Summary measurements of performance improvements, depending on the language pair and dataset, range from 6.73 absolute MaxF1 percentage points to 16.75 absolute MaxF1 percentage points and from 5.58 absolute 11-point interpolated average precision percentage points to 17.19 absolute 11-point interpolated average precision percentage points. Section 4 discusses the results and possible extensions of the method. Section 5 wraps up with the main conclusions.

2 Algorithm

While our focus in this paper is on using global constraints to improve cognates detection, we believe that our method is useful more generally. We therefore abstract out some of the specifics of cognates detection and present our algorithm more generally in this section, with the hope that it will be able to be used in the future for other applications in addition to cognates detection. None of our abstraction harms understanding of our method's applicability to cognates detection and the fact that the method may be more widely beneficial does not in any way detract from the utility we show it has for improving cognates detection.

A common setting is where one has a set $X = \{x_1, x_2, ..., x_n\}$ and a set $Y = \{y_1, y_2, ..., y_n\}$ where the task is to extract (x, y) pairs such that (x, y) are in some relation R. Here are examples:

- X might be a set of states and Y might be a set of cities and the relation R might be "is the capital of";
- X might be a set of images and Y might be a set of people's names and the relation R might be "is a picture of";
- X might be a set of English words and Y might be a set of French words and the relation R might be "is cognate with".

A common way these problems are approached is that a model is trained that can score each pair (x, y) and those pairs with scores above a threshold are extracted. We propose that often the relation will have a tendency, or a hard constraint, to satisfy particular properties and that this ought to be utilized to improve the quality of the extracted pairs.

The approach we put forward is to re-score each (x, y) pair by utilizing scores generated for other pairs and our knowledge of properties of the relation being extracted. In this paper, we present and evaluate methods for improving the scores of each (x, y) pair for the case when the relation is known to be one-to-one and discuss extensions to other situations.

The current approach is to generate a matrix of scores for each candidate pair as follows:

$$Score_{X,Y} = \begin{bmatrix} s_{x_1,y_1} & \cdots & s_{x_1,y_n} \\ \vdots & \ddots & \vdots \\ s_{x_n,y_1} & \cdots & s_{x_n,y_n} \end{bmatrix}. \quad (1)$$

Then those pairs with scores above a threshold are predicted as being in the relation. We now describe methods for sharpening the scores in the matrix by utilizing the fact that there is an overarching global structure on the predictions.

2.1 Reverse Rank

We know that if $(x_i, y_j) \in R$, then $(x_k, y_j) \notin R$ for $k \neq i$ when R is 1-to-1. We define $reverse\_rank(x_i, y_j) = |\{x_k \in X | s_{x_k,y_j} \geq s_{x_i,y_j}\}|$. Intuitively, a high reverse rank means that there are lots of other elements of X that score better to y_j than x_i does; this could be evidence that (x_i, y_j) is not in R and ought to have a lower score. Alternatively, if there are very few or no other elements of X that score better to y_j than x_i does this

could be evidence that (x_i, y_j) is in R and ought to have a higher score. In accord with this intuition, we use reverse rank as the basis for rescaling our scores as follows:

$$score_{RR}(x_i, y_j) = \frac{s_{x_i,y_j}}{reverse\_rank(x_i, y_j)}. \quad (2)$$

2.2 Forward Rank

Analogous to reverse rank, another basis we can use for adjusting scores is the forward rank. We define $forward\_rank(x_i, y_j) = |\{y_k \in Y | s_{x_i,y_k} \geq s_{x_i,y_j}\}|$. We then scale the scores analogously to how we did with reverse ranks via an inverse linear function.[2]

2.3 Combining Reverse Rank and Forward Rank

For combining reverse rank and forward rank, we present results of experiments doing it two ways. The first is a 1-step approach:

$$score_{RR\_FR\_1step}(x_i, y_j) = \frac{s_{x_i,y_j}}{product}, \quad (3)$$

where

$$\begin{aligned} product = & reverse\_rank(x_i, y_j) \times \\ & forward\_rank(x_i, y_j). \end{aligned} \quad (4)$$

The second combination method involves first computing the reverse rank and re-adjusting every score based on the reverse ranks. Then in a second step the new scores are used to compute forward ranks and then those scores are adjusted based on the forward ranks. We refer to this method as RR_FR_2step.

2.4 Maximum Assignment

If one makes the assumption that all elements in X and Y are present and have their partner element in the other set present with no extra elements and the sets are not too large, then it is interesting to compute what the 'maximal assignment' would be using the Hungarian Algorithm to optimize:

$$\begin{aligned} \max_{Z \in X \times Y} \quad & \sum_{(x,y) \in Z} score(x, y) \\ \text{s.t.} \quad & (x_i, y_j) \in Z \Rightarrow (x_k, y_j) \notin Z, \forall k \neq i \\ & (x_i, y_j) \in Z \Rightarrow (x_i, y_k) \notin Z, \forall k \neq j. \end{aligned} \quad (5)$$

We do this on datasets where the assumptions hold and see how close our methods get to the Hungarian maximal assignment at similar points of the precision-recall curves. For our larger datasets where the assumptions don't hold, the Hungarian either can't complete due to limited computational resources or it functioned poorly in comparison with the performance of our reverse rank and forward rank combination methods.

3 Experiments

Our goal is to test whether using the global structure algorithms we described in section 2 can significantly boost performance for cognates detection. To test this hypothesis, our first step is to implement a system that uses state of the art research results to generate the initial score matrices as a current state of the art system would currently do for this task. To that end, we implemented a baseline state of the art system that uses the information sources that previous research has found to be helpful for this task such as phonetic information, word context information, temporal context information, word frequency information, and word burstiness information (Kondrak, 2001; Mann and Yarowsky, 2001; Schafer and Yarowsky, 2002; Klementiev and Roth, 2006; Irvine and Callison-Burch, 2013). Consistent with past work (Irvine and Callison-Burch, 2013), we use supervised training to learn the weights for combining the various information sources. The system combines the sources of information by using weights learned by an SVM (Support Vector Machine) on a small seed training set of cognates[3] to optimize performance. This baseline system obtains state of the art performance on cognates detection. Using this state of the art system as our baseline, we investigated how much we could improve performance beyond current state of the art levels by applying the rescoring algorithm we described in section 2. We performed experiments on three language pairs: French-English, German-English, and Spanish-English, with different text corpora used as training and test data. The different language pairs and datasets have different levels of performance in terms of their baseline current state of the art score matrices. In the

[2] For both reverse rank and forward rank we also experimented with exponential decay and step functions, but found that simple division by the ranks worked as well or better than any of those more complicated methods.

[3] The small seed set was randomly selected and less than 20% in all cases. It was not used for testing. Note that using this data to optimize performance of the baseline system makes our baseline even stronger and makes it even harder for our new rescoring method to achieve larger improvements.

next few subsections, we describe our experimental details.

3.1 Lemmatization

We used morphological analyzers to convert the words in text corpora to lemma form. For English, we used the NLTK WordNetLemmatizer (Bird et al., 2009). For French, German, and Spanish we used the TreeTagger (Schmid, 1994).

3.2 Word Context Information

We used the Google N-Gram corpus (Michel et al., 2010). For English we used the English 2012 Google 5-gram corpus, for French we used the French 2012 Google 5-gram corpus, for German we used the German 2012 Google 5-gram corpus, and for Spanish we used the Spanish 2012 Google 5-gram corpus. From these corpora we compute word context similarity scores across languages using Rapp's method (Rapp, 1995, 1999). The intuition behind this method is that cognates are more likely to occur in correlating context windows and this statistic inferred from large amounts of data captures this correlation.

3.3 Frequency Information

The intuition is that over large amounts of data cognates should have similar relative frequencies. We compute our relative frequencies by using the same corpora mentioned in the previous subsection.

3.4 Temporal Information

The intuition is that cognates will have similar temporal distributions (Klementiev and Roth, 2006). To compute the temporal similarity we use newspaper data and convert it to simple daily word counts. For each word in the corpora the word counts create a time series vector. The Fourier transform is computed on the time series vectors. Spearman rank correlation is computed on the transform vectors. For English we used the English Gigaword Fifth Edition[4]. For French we used French Gigaword Third Edition[5]. For Spanish we used Spanish Gigaword First Edition[6]. The German news corpora were obtained by web crawling `http://www.tagesspiegel.de/` and extracting the news articles.

[4]Linguistic Data Consortium Catalog No. LDC2011T07
[5]Linguistic Data Consortium Catalog No. LDC2011T10
[6]Linguistic Data Consortium Catalog No. LDC2006T12

3.5 Word Burstiness

The intuition is that cognates will have similar burstiness measures (Church and Gale, 1995). For word burstiness we used the same corpora as for the temporal corpora.

3.6 Phonetic Information

The intuition is that cognates will have correspondences in how they are pronounced. For this, we compute a measurement based on Normalized Edit Distance (NED).

3.7 Combining Information Sources

We combine the information sources by using a linear Support Vector Machine to learn weights for each of the information sources from a small seed training set of cognates. So our final score assigned to a candidate cognate pair (x, y) is:

$$score(x, y) = \sum_{m \in metrics} w_m score_m(x, y), \quad (6)$$

where $metrics$ is the set of measurements such as phonetic similarity measurements, word burstiness similarity, relative frequency similarity, etc. that were explained in subsections 3.2 through 3.6; w_m is the learned weight for metric m; and $score_m(x, y)$ is the score assigned to the pair (x, y) by metric m.

The scores such assigned represent a state of the art approach for filling in the matrix identified in equation 1. At this point the matrix of scores would be used to predict cognates. We now turn to evaluation of the use of the global constraint rescoring methods from section 2 for improving performance beyond the state of the art levels.

3.8 Using Global Constraints to Rescore

For our cognates data we used the French-English pairs from (Bergsma and Kondrak, 2007) and the German-English and Spanish-English pairs from (Beinborn et al., 2013). Figure 1 shows the precision-recall[7] curves for

[7]Precision and recall are the standard measures used for systems that perform search. Precision is the percentage of predicted cognates that are indeed cognate. Recall is the percentage of cognates that are predicted as cognate. We vary the threshold that determines cognateness to generate all points along the Precision-Recall curve. We start with a very high threshold enabling precision of 100% and lower the threshold until recall of 100% is reached. In particular, we sort the test examples by score in descending order and then go down the list of scores in order to complete the entire precision-recall curve.

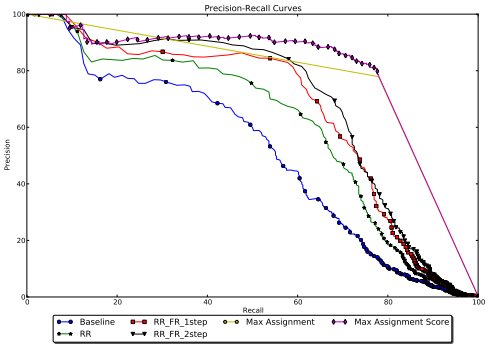


Figure 1: Precision-Recall Curves for French-English. Baseline denotes state of the art performance.

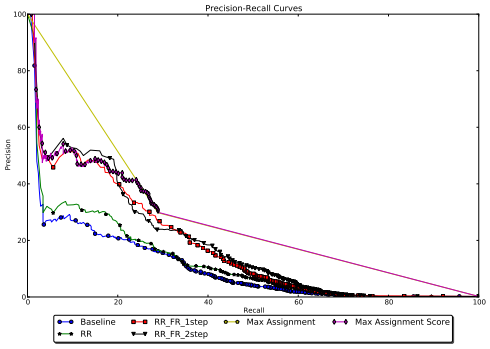


Figure 2: Precision-Recall Curves for German-English. Baseline denotes state of the art performance.

French-English, Figure 2 shows the performance for German-English, and Figure 3 shows the performance for Spanish-English. Note that state of the art performance (denoted in the figures as Baseline) has very different performance across the three datasets, but in all cases the systems from section 2 that incorporate global constraints and perform rescoring greatly exceed current state of the art performance levels. The Max Assignment is really just the single point that the Hungarian finds. We drew lines connecting it, but keep in mind those lines are just connecting the single point to the endpoints. Max Assignment Score traces the precision-recall curve back from the Max Assignment by steadily increasing the threshold so that only points in the maximum assignment set with scores above the increasing threshold are predicted as cognate.

For the non-max assignment curves, it is sometimes helpful to compute a single metric summarizing important aspects of the full curve. For this purpose, maxF1 and 11-point interpolated average precision are often used. MaxF1 is the F1 measure (i.e., harmonic mean of precision and recall) at the point on the precision-recall curve where F1 is highest. The interpolated precision p_{interp} at a given recall level r is defined as the highest precision level found for any recall level $r' \geq r$:

$$p_{interp}(r) = max_{r' \geq r} p(r'). \quad (7)$$

The 11-point interpolated average precision (11-point IAP) is then the average of the p_{interp} at $r = 0.0, 0.1, ..., 1.0$. Table 1 shows these performance measures for French-English, Table 2

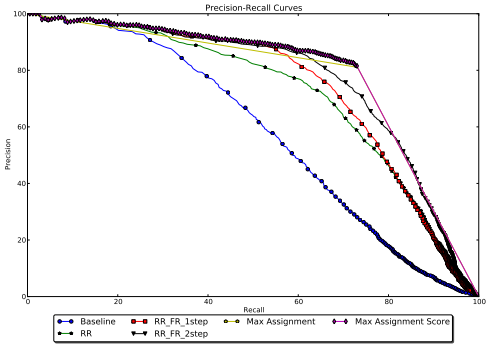


Figure 3: Precision-Recall Curves for Spanish-English. Baseline denotes state of the art performance.

shows the results for German-English, and Table 3 show the results for Spanish-English. In all cases, using global structure greatly improves upon the state of the art baseline performance. In (Bergsma and Kondrak, 2007), for French-English data a result of 66.5 11-point IAP is reported for a situation where word alignments from a bitext are available and a result of 77.7 11-point IAP is reported for a situation where translation pairs are available in large quantities. The setting considered in the current paper is much more challenging since it does not use bilingual dictionaries or word alignments from bitexts. The setting in the current paper is the one mentioned as future work on page 663 of (Bergsma and Kondrak, 2007): "In particular, we plan to investigate approaches that do not re-

METHOD	MAX F1	11-POINT IAP
BASELINE	54.92	50.99
RR	62.94	59.62
RR_FR_1STEP	68.35	64.42
RR_FR_2STEP	69.72	67.29

Table 1: French-English Performance. BASELINE indicates current state of the art performance.

METHOD	MAX F1	11-POINT IAP
BASELINE	21.38	16.25
RR	22.71	17.80
RR_FR_1STEP	28.68	22.37
RR_FR_2STEP	28.11	21.83

Table 2: German-English Performance. BASELINE indicates current state of the art performance.

quire the bilingual dictionaries or bitexts to generate training data."

Note that the evaluation thus far is a bit artificial for real cognates detection because in a real setting you wouldn't only be selecting matches for relatively small subsets of words that are guaranteed to have a cognate on the other side. Such was the case for our evaluation where the French-English set had approx. 600 cognate pairs, the German-English set had approx. 1000 pairs, and the Spanish-English set had approx. 3000 pairs. In a real setting, the system would have to consider words that don't have a cognate match in the other language and not only words that were hand-selected and guaranteed to have cognates. We are not aware of others evaluating according to this much more difficult condition, but we think it is important to consider especially given the potential impacts it could have on the global structure methods we've put forward. Therefore, we run a second set of evaluations where we take the ten thousand most common words in our corpora for each of our languages, which contain many of the cognates from the standard test sets and we add in any remaining words from the standard test sets that didn't make it into the top ten thousand. We then repeat each of the experiments under this much more challenging condition. With approx. ten thousand squared candidates, i.e., approx. 100 million candidates, to consider for cognateness, this is a large data condition. The Hungarian didn't run to completion on two of the datasets due to limited computational resources. On French-English it completed, but achieved poorer performance than any of the other methods. This makes sense as it is designed when there really is a bipartite matching to be found like in the artificial yet standard cognates evaluation that was just presented. When confronted with large amounts of words that create a much denser space and have no match at all on the other side the all or nothing assignments of the Hungarian are not ideal. The reverse rank and forward rank rescoring methods are still quite effective in improving performance although not by as much as they did in the small data results from above.

METHOD	MAX F1	11-POINT IAP
BASELINE	56.26	57.03
RR	68.52	69.33
RR_FR_1STEP	70.66	71.47
RR_FR_2STEP	73.01	74.22

Table 3: Spanish-English Performance. BASELINE indicates current state of the art performance.

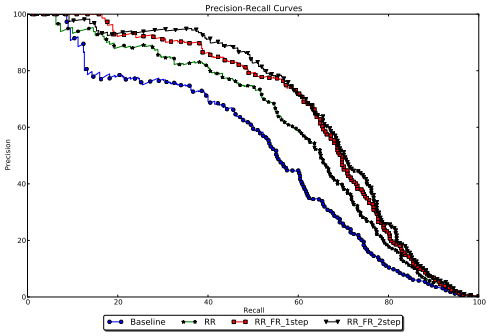


Figure 4: Precision-Recall Curves for French-English (large data). Note that Baseline denotes state of the art performance.

Figure 4 shows the full precision-recall curves for French-English for the large data condition, Figure 5 shows the curves for German-English for the large data condition, and Figure 6 shows the results for Spanish-English for the large data condition.

Tables 4 through 6 show the summary metrics for the three language pairs for the large data experiments. We can see that the reverse rank and forward rank methods of taking into account the global structure of interactions among predictions is still helpful, providing large improvements in performance even in this challenging large data condition over strong state of the art baselines that

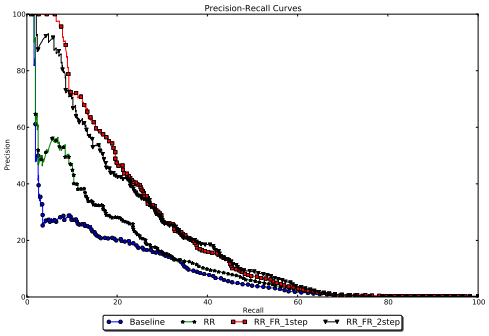


Figure 5: Precision-Recall Curves for German-English (large data). Note that Baseline denotes state of the art performance.

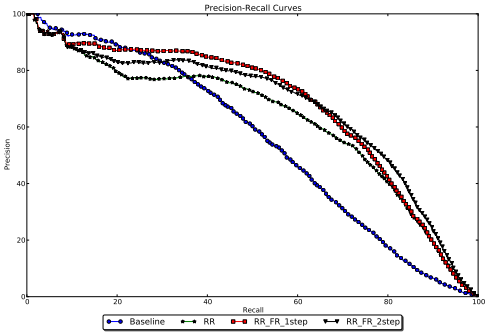


Figure 6: Precision-Recall Curves for Spanish-English (large data). Note that Baseline denotes state of the art performance.

make cognate predictions independently of each other and don't do any rescoring based on global constraints.

4 Discussion

We believe that this work opens up new avenues for further exploration. A few of these include the following:

- investigating the utility of applying and extending the method to other applications such as Information Extraction applications, many of which have similar global constraints as cognates detection;
- investigating how to handle other forms of global structure including tendencies that are not necessarily hard constraints;
- developing more theory to more precisely understand some of the nuances of using global structure when it's applicable and making connections with other areas of machine learning such as semi-supervised learning, active learning, etc.; and
- investigating how to have a machine *learn* that global structure exists and *learn* what form of global structure exists.

METHOD	MAX F1	11-POINT IAP
BASELINE	55.08	51.35
RR	60.88	58.79
RR_FR_1STEP	65.87	63.55
RR_FR_2STEP	65.76	65.26

Table 4: French-English Performance (large data). BASELINE indicates state of the art performance.

METHOD	MAX F1	11-POINT IAP
BASELINE	21.25	16.17
RR	24.78	19.13
RR_FR_1STEP	30.72	24.97
RR_FR_2STEP	30.34	24.86

Table 5: German-English Performance (large data). BASELINE indicates state of the art performance.

METHOD	MAX F1	11-POINT IAP
BASELINE	54.75	54.55
RR	62.52	61.42
RR_FR_1STEP	66.45	65.89
RR_FR_2STEP	66.38	65.5

Table 6: Spanish-English Performance (large data). BASELINE indicates state of the art performance.

5 Conclusions

Cognates detection is an interesting and challenging task. Previous work has yielded state of the art approaches that create a matrix of scores for all word pairs based on optimized weighted combinations of component scores computed on the basis of various helpful sources of information such as phonetic information, word context information, temporal context information, word frequency information, and word burstiness information. However, when assigning a score to a word pair, the current state of the art methods do not take into account scores assigned to other word pairs. We proposed a method for rescoring the matrix that cur-

rent state of the art methods produce by taking into account the scores assigned to other word pairs. The methods presented in this paper are complementary to existing state of the art methods, easy to implement, computationally efficient, and practically effective in improving performance by large amounts. Experimental results reveal that the new methods significantly improve state of the art performance in multiple cognates detection experiments conducted on standard freely and publicly available datasets with different language pairs and various conditions such as different levels of baseline performance and different data size conditions, including with more realistic large data size conditions than have been evaluated with in the past.

References

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2013. Cognate production using character-based machine translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Nagoya, Japan, pages 883–891. http://www.aclweb.org/anthology/I13-1112.

Shane Bergsma and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 656–663. http://www.aclweb.org/anthology/P07-1083.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated Reconstruction of Ancient Languages using Probabilistic Models of Sound Change. *Proceedings of the National Academy of Sciences* 110:4224–4229. https://doi.org/10.1073/pnas.1204678110.

Kenneth W. Church and William A. Gale. 1995. Poisson mixtures. *Natural Language Engineering* 1:163–190.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1–8. https://doi.org/10.3115/1118693.1118694.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics* 31(1):25–70. https://doi.org/10.1162/0891201053630273.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*. Barcelona, Spain, pages 111–118. https://doi.org/10.3115/1218955.1218970.

David Hall and Dan Klein. 2011. Large-scale cognate recovery. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., pages 344–354. http://www.aclweb.org/anthology/D11-1032.

Ann Irvine and Chris Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 518–523. http://www.aclweb.org/anthology/N13-1056.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sydney, Australia, pages 817–824. https://doi.org/10.3115/1220175.1220278.

Grzegorz Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '01, pages 1–8. http://www.aclweb.org/anthology/N/N01/N01-1014.pdf.

Grzegorz Kondrak, Daniel Marcu, and Kevin Knight. 2003. Cognates can improve statistical translation models. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL-Short '03, pages 46–48. http://www.aclweb.org/anthology/N/N03/N03-2016.pdf.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '01, pages

1–8. http://www.aclweb.org/anthology/N/N01/N01-1020.pdf.

Benjamin S. Mericli and Michael Bloodgood. 2012. Annotating cognates and etymological origin in Turkic languages. In *Proceedings of the First Workshop on Language Resources and Technologies for Turkic Languages at the Eighth International Conference on Languange Resources and Evaluation (LREC'12)*. European Language Resources Association, Istanbul, Turkey, pages 47–51. http://arxiv.org/abs/1501.03191.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2010. Quantitative analysis of culture using millions of digitized books. *Science* 331(6014):176–182. https://doi.org/10.1126/science.1199644.

Reinhard Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Cambridge, Massachusetts, USA, pages 320–322. https://doi.org/10.3115/981658.981709.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, College Park, Maryland, USA, pages 519–526. https://doi.org/10.3115/1034678.1034756.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *Proceedings of the 6th Conference on Natural language Learning - Volume 20*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 1–7. http://www.aclweb.org/anthology/W/W02/W02-2026.pdf.

Helmut Schmid. 1994. Part-of-speech tagging with neural networks. In *COLING*. pages 172–176. http://www.aclweb.org/anthology/C/C94/C94-1027.pdf.

Michel Simard, George F. Foster, and Pierre Isabelle. 1993. Using cognates to align sentences in bilingual corpora. In *Proceedings of the 1993 Conference of the Centre for Advanced Studies on Collaborative Research: Distributed Computing - Volume 2*. IBM Press, CASCON '93, pages 1071–1082.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005a. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine learning*. ACM, pages 896–903.

Ben Taskar, Lacoste-Julien Simon, and Klein Dan. 2005b. A discriminative matching approach to word alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Vancouver, British Columbia, Canada, pages 73–80. http://www.aclweb.org/anthology/H/H05/H05-1010.pdf.

One-Shot Neural Cross-Lingual Transfer for Paradigm Completion

Katharina Kann
CIS
LMU Munich, Germany
kann@cis.lmu.de

Ryan Cotterell
Department of Computer Science
Johns Hopkins University, USA
ryan.cotterell@jhu.edu

Hinrich Schütze
CIS
LMU Munich, Germany
inquiries@cislmu.org

Abstract

We present a novel cross-lingual transfer method for paradigm completion, the task of mapping a lemma to its inflected forms, using a neural encoder-decoder model, the state of the art for the monolingual task. We use labeled data from a high-resource language to increase performance on a low-resource language. In experiments on 21 language pairs from four different language families, we obtain up to 58% higher accuracy than without transfer and show that even zero-shot and one-shot learning are possible. We further find that the degree of language relatedness strongly influences the ability to transfer morphological knowledge.

1 Introduction

Low-resource natural language processing (NLP) remains an open problem for many tasks of interest. Furthermore, for most languages in the world, high-cost linguistic annotation and resource creation are unlikely to be undertaken in the near future. In the case of morphology, out of the 7000 currently spoken (Lewis, 2009) languages, only about 200 have computer-readable annotations (Sylak-Glassman et al., 2015) – although morphology is easy to annotate compared to syntax and semantics. *Transfer learning* is one solution to this problem: it exploits annotations in a high-resource language to train a system for a low-resource language. In this work, we present a method for cross-lingual transfer of inflectional morphology using an encoder-decoder recurrent neural network (RNN). This allows for the development of tools for computational morphology with limited annotated data.

In many languages, individual lexical entries may be realized as distinct inflections of a single lemma depending on the syntactic context. For example, the 3SgPresInd of the English verbal lemma *to bring* is *brings*. In morphologically rich languages, a lemma can have hundreds of individual forms. Thus, both generation and analysis of such morphological inflections are active areas of research in NLP and morphological processing has been shown to be a boon to several other down-stream applications, e.g., machine translation (Dyer et al., 2008), speech recognition (Creutz et al., 2007), parsing (Seeker and Çetinoğlu, 2015), keyword spotting (Narasimhan et al., 2014) and word embeddings (Cotterell et al., 2016b), *inter alia*. In this work, we focus on paradigm completion, a form of morphological generation that maps a given lemma to a target inflection, e.g., (*bring*, Past) $\mapsto$ *brought* (with Past being the target tag).

	Present Indicative		Past Indicative	
	Sg	Pl	Sg	Pl
1	*sueño*	*soñamos*	*soñé*	*soñamos*
2	*sueñas*	*soñáis*	*soñaste*	*soñasteis*
3	*sueña*	*sueñan*	*soñó*	*soñaron*

Table 1: Partial inflection table for the Spanish verb *soñar*.

RNN sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2015) are the state of the art for paradigm completion (Faruqui et al., 2016; Kann and Schütze, 2016a; Cotterell et al., 2016a). However, these models require a large amount of data to achieve competitive performance; this makes them unsuitable for out-of-the-box application to paradigm completion in the low-resource scenario. To mitigate this, we consider transfer learning: we train an end-to-end neural system jointly with limited data from a low-resource language and a larger amount of data from a high-resource language. This technique allows

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1993–2003
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1182

the model to apply knowledge distilled from the high-resource training data to the low-resource language as needed.

We conduct experiments on 21 language pairs from four language families, emulating a low-resource setting. Our results demonstrate successful transfer of morphological knowledge. We show improvements in accuracy and edit distance of up to 58% (accuracy) and 4.62 (edit distance) over the same model with only in-domain language data on the paradigm completion task. We further obtain up to 44% (resp. 14%) improvement in accuracy for the one-shot (resp. zero-shot) setting, i.e., one (resp. zero) in-domain language sample per target tag. We also show that the effectiveness of morphological transfer depends on language relatedness, measured by lexical similarity.

2 Inflectional Morphology and Paradigm Completion

Many languages exhibit inflectional morphology, i.e., the form of an individual lexical entry mutates to show properties such as person, number or case. The citation form of a lexical entry is referred to as the **lemma** and the collection of its possible inflections as its **paradigm**. Tab. 1 shows an example of a partial paradigm; we display several forms for the Spanish verbal lemma *soñar*. We may index the entries of a paradigm by a **morphological tag**, e.g., the 2SgPresInd form *sueñas* in Tab. 1. In generation, the speaker must select an entry of the paradigm given the form's context. In general, the presence of rich inflectional morphology is problematic for NLP systems as it greatly increases the token-type ratio and, thus, word form sparsity.

An important task in inflectional morphology is **paradigm completion** (Durrett and DeNero, 2013; Ahlberg et al., 2014; Nicolai et al., 2015; Cotterell et al., 2015; Faruqui et al., 2016). Its goal is to map a lemma to all individual inflections, e.g., $(\textit{soñar}, \textsf{1SgPresInd}) \mapsto \textit{sueño}$. There are good solutions for paradigm completion when a large amount of annotated training data is available (Cotterell et al., 2016a).[1] In this work, we address the low-resource setting, a yet unsolved challenge.

[1]The SIGMORPHON 2016 shared task (Cotterell et al., 2016a) on morphological reinflection, a harder generalization of paradigm completion, found that $\geq 98\%$ accuracy can be achieved in many languages with neural sequence-to-sequence models, improving the state of the art by 10%.

2.1 Transferring Inflectional Morphology

In comparison to other NLP annotations, e.g., part-of-speech (POS) and named entities, morphological inflection is especially challenging for transfer learning: we can define a universal set of POS tags (Petrov et al., 2012) or of entity types (e.g., coarse-grained types like *person* and *location* or fine-grained types (Yaghoobzadeh and Schütze, 2015)), but inflection is much more language-specific. It is infeasible to transfer morphological knowledge from Chinese to Portuguese as Chinese does not use inflected word forms. Transferring named entity recognition, however, among Chinese and European languages works well (Wang and Manning, 2014a). But even transferring inflectional paradigms from morphologically rich Arabic to Portuguese seems difficult as the inflections often mark dissimilar subcategories. In contrast, transferring morphological knowledge from Spanish to Portuguese, two languages with similar conjugations and 89% lexical similarity, appears promising. Thus, we conjecture that transfer of inflectional morphology is only viable among *related languages*.

2.2 Formalization of the Task

We now offer a formal treatment of the cross-lingual paradigm completion task and develop our notation. Let Σ_ℓ be a discrete alphabet for language ℓ and let $\mathcal{T}_\ell$ be a set of morphological tags for ℓ. Given a lemma w_ℓ in ℓ, the morphological paradigm (inflectional table) π can be formalized as a set of pairs

$$\pi(w_\ell) = \Big\{ \big(f_k[w_\ell], t_k\big) \Big\}_{k \in T(w_\ell)} \quad (1)$$

where $f_k[w_\ell] \in \Sigma_\ell^+$ is an inflected form, $t_k \in \mathcal{T}_\ell$ is its morphological tag and $T(w_\ell)$ is the set of slots in the paradigm; e.g., a Spanish paradigm is:

$$\pi(\text{soñar}) = \Big\{ \big(\textit{sueño}, \textsf{1SgPresInd}\big), \ldots, \big(\textit{soñaran}, \textsf{3PlPastSbj}\big) \Big\}$$

Paradigm completion consists of predicting missing slots in the paradigm $\pi(w_\ell)$ of a given lemma w_ℓ.

In cross-lingual paradigm completion, we consider a *high-resource source language* ℓ_s (lots of training data available) and a *low-resource target language* ℓ_t (little training data available). We denote the source training examples as $\mathcal{D}_s$ (with $|\mathcal{D}_s| = n_s$) and the target training examples as

$\mathcal{D}_t$ (with $|\mathcal{D}_t| = n_t$). The goal of cross-lingual paradigm completion is to populate paradigms in the low-resource target language with the help of data from the high-resource source language, using only few in-domain examples.

3 Cross-Lingual Transfer as Multi-Task Learning

We describe our probability model for morphological transfer using terminology from multi-task learning (Caruana, 1997; Collobert et al., 2011). We consider two tasks, training a paradigm completor (i) for a high-resource language and (ii) for a low-resource language. We want to train jointly, so we reap the benefits of having related languages. Thus, we define the log-likelihood as

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(k,w_{\ell_t})\in\mathcal{D}_t} \log p_{\boldsymbol{\theta}}\left(f_k[w_{\ell_t}] \mid w_{\ell_t}, t_k, \lambda_{\ell_t}\right) \quad (2)$$
$$+ \sum_{(k,w_{\ell_s})\in\mathcal{D}_s} \log p_{\boldsymbol{\theta}}(f_k[w_{\ell_s}] \mid w_{\ell_s}, t_k, \lambda_{\ell_s})$$

where we tie parameters $\boldsymbol{\theta}$ for the two languages together to allow the transfer of morphological knowledge between languages. The λs are special language tags, cf. Sec. 3.2. Each probability distribution $p_{\boldsymbol{\theta}}$ defines a distribution over all possible realizations of an inflected form, i.e., a distribution over Σ^*. For example, consider the related Romance languages Spanish and French; focusing on one term from each of the summands in Eq. (2) (the past participle of the translation of *to visit* in each language), we arrive at

$$\mathcal{L}_{\text{visit}}(\boldsymbol{\theta}) = \log p_{\boldsymbol{\theta}}(\textit{visitado} \mid \textit{visitar}, \textsf{PastPart}, \textsf{ES})$$
$$+ \log p_{\boldsymbol{\theta}}(\textit{visité} \mid \textit{visiter}, \textsf{PastPart}, \textsf{FR}) \quad (3)$$

Our *cross-lingual* setting forces both transductions to share part of the parameter vector $\boldsymbol{\theta}$, to represent morphological regularities between the two languages in a common embedding space and, thus, to enable morphological transfer. This is no different from *monolingual* multi-task settings, e.g., jointly training a parser and tagger for transfer of syntax.

Based on recent advances in neural transducers, we parameterize each distribution as an encoder-decoder RNN, as in (Kann and Schütze, 2016b). In their setup, the RNN encodes the input and predicts the forms in a *single* language. In contrast, we force the network to predict *two or more* languages.

3.1 Encoder-Decoder RNN

We parameterize the distribution $p_{\boldsymbol{\theta}}$ as an encoder-decoder gated RNN (GRU) with attention (Bahdanau et al., 2015), the state-of-the-art solution for the monolingual case (Kann and Schütze, 2016b). A bidirectional gated RNN encodes the input sequence (Cho et al., 2014) – the concatenation of (i) the language tag, (ii) the morphological tag of the form to be generated and (iii) the characters of the input word – represented by embeddings. The input to the decoder consists of concatenations of $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$, the forward and backward hidden states of the encoder. The decoder, a unidirectional RNN, uses attention: it computes a weight α_i for each h_i. Each weight reflects the importance given to that input position. Using the attention weights, the probability of the output sequence given the input sequence is:

$$p(y \mid x_1, \ldots, x_{|X|}) = \prod_{t=1}^{|Y|} g(y_{t-1}, s_t, c_t) \quad (4)$$

where $y = (y_1, \ldots, y_{|Y|})$ is the output sequence (a sequence of $|Y|$ characters), $x = (x_1, \ldots x_{|X|})$ is the input sequence (a sequence of $|X|$ characters), g is a non-linear function, s_t is the hidden state of the decoder and c_t is the sum of the encoder states h_i, weighted by attention weights $\alpha_i(s_{t-1})$ which depend on the decoder state:

$$c_t = \sum_{i=1}^{|X|} \alpha_i(s_{t-1}) h_i \quad (5)$$

Fig. 1 shows the encoder-decoder. See Bahdanau et al. (2015) for further details.

3.2 Input Format

Each source form is represented as a sequence of characters; each character is represented as an embedding. In the same way, each source tag is represented as a sequence of subtags, and each subtag is represented as an embedding. More formally, we define the alphabet $\Sigma = \cup_{\ell\in L}\Sigma_\ell$ as the set of characters in the languages in L, with L being the set of languages in the given experiment. Next, we define $\mathcal{S}$ as the set of subtags that occur as part of the set of morphological tags $\mathcal{T} = \cup_{\ell\in L}\mathcal{T}_\ell$; e.g., if 1SgPresInd $\in \mathcal{T}$, then 1, Sg, Pres, Ind $\in \mathcal{S}$. Note that the set of subtags $\mathcal{S}$ is defined as attributes from the UniMorph schema (Sylak-Glassman, 2016) and, thus, is universal across languages; the schema is

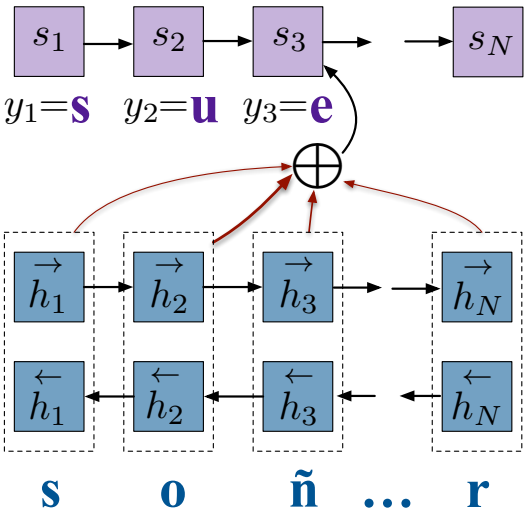


Figure 1: Encoder-decoder RNN for paradigm completion. The lemma *soñar* is mapped to a target form (e.g., *sueña*). For brevity, language and target tags are omitted from the input. Thickness of red arrows symbolizes the degree to which the model attends to the corresponding hidden state of the encoder.

derived from research in linguistic typology.[2] The format of the input to our system is $\mathcal{S}^+\Sigma^+$. The output format is Σ^+. Both input and output are padded with distinguished BOW and EOW symbols.

What we have described is the representation of Kann and Schütze (2016b). In addition, we preprend a symbol $\lambda \in L$ to the input string (e.g., λ = Es, also represented by an embedding), so the RNN can handle multiple languages simultaneously and generalize over them. Thus, our final input is of the form $\lambda\mathcal{S}^+\Sigma^+$.

4 Languages and Language Families

To verify the applicability of our method to a wide range of languages, we perform experiments on example languages from several different families.

Romance languages, a subfamily of Indo-European, are widely spoken, e.g., in Europe and Latin America. Derived from the common ancestor Vulgar Latin (Harris and Vincent, 2003), they share large parts of their lexicon and inflectional morphology; we expect knowledge among them to be easily transferable.

[2]Note that while the subtag set is universal, *which* subtags a language actually uses is language-specific; e.g., Spanish does not mark animacy as Russian does. We contrast this with the universal POS set (Petrov et al., 2012), where it is more likely that we see all 17 tags in most languages.

	PT	CA	IT	FR
similarity to ES	89%	85%	82%	75%

Table 2: Lexical similarities for Romance (Lewis, 2009).

We experiment on Catalan, French, Italian, Portuguese and Spanish. Tab. 2 shows that Spanish – which takes the role of the low-resource language in our experiments – is closely related with the other four, with Portuguese being most similar. We hypothesize that the transferability of morphological knowledge between source and target corresponds to the degree of lexical similarity; thus, we expect Portuguese and Catalan to be more beneficial for Spanish than Italian and French.

The Indo-European **Slavic language family** has its origin in eastern-central Europe (Corbett and Comrie, 2003). We experiment on Bulgarian, Macedonian, Russian and Ukrainian (Cyrillic script) and on Czech, Polish and Slovene (Latin script). Macedonian and Ukranian are low-resource languages, so we assign them the low-resource role. For Romance and for Uralic, we experiment with groups containing three or four source languages. To arrive at a comparable experimental setup for Slavic, we run two experiments, each with three source and one target language: (i) from Russian, Bulgarian and Czech to Macedonian; and (ii) from Russian, Polish and Slovene to Ukrainian.

We hope that the paradigm completor learns similar embeddings for, say, the characters "e" in Polish and "є" in Ukrainian. Thus, the use of two scripts in Slavic allows us to explore transfer across different alphabets.

We further consider a non-Indo-European language family, the **Uralic languages**. We experiment on the three most commonly spoken languages – Finnish, Estonian and Hungarian (Abondolo, 2015) – as well as Northern Sami, a language used in Northern Scandinavia. While Finnish and Estonian are closely related (both are members of the Finnic subfamily), Hungarian is a more distant cousin. Estonian and Northern Sami are low-resource languages, so we assign them the low-resource role, resulting in two groups of experiments: (i) Finnish, Hungarian and Estonian to Northern Sami; (ii) Finnish, Hungarian and Northern Sami to Estonian.

Arabic (baseline) is a Semitic language (part of the Afro-Asiatic family (Hetzron, 2013)) that is

spoken in North Africa, the Arabian Peninsula and other parts of the Middle East. It is unrelated to all other languages used in this work. Both in terms of form (new words are mainly built using a templatic system) and categories (it has tags such as construct state), Arabic is very different. Thus, we do not expect it to support morphological knowledge transfer and use it as a baseline for all target languages.

5 Experiments

We run four experiments on 21 distinct pairings of languages to show the feasibility of morphological transfer and analyze our method. We first discuss details common to all experiments.

We keep **hyperparameters** during all experiments (and for all languages) fixed to the following values. Encoder and decoder RNNs each have 100 hidden units and the size of all subtag, character and language embeddings is 300. For training we use ADADELTA (Zeiler, 2012) with minibatch size 20. All models are trained for 300 epochs. Following Le et al. (2015), we initialize all weights in the encoder, decoder and the embeddings except for the GRU weights in the decoder to the identity matrix. Biases are initialized to zero.

Evaluation metrics: (i) 1-best accuracy: the percentage of predictions that match the true answer exactly; (ii) average edit distance between prediction and true answer. The two metrics differ in that accuracy gives no partial credit and incorrect answers may be drastically different from the annotated form without incurring additional penalty. In contrast, edit distance gives partial credit for forms that are closer to the true answer.

5.1 Exp. 1: Transfer Learning for Paradigm Completion

In this experiment, we investigate to what extent our model transfers morphological knowledge from a high-resource source language to a low-resource target language. We experimentally answer three questions. (i) Is transfer learning possible for morphology? (ii) How much annotated data do we need in the low-resource target language? (iii) How closely related must the two languages be to achieve good results?

Data. Based on complete inflection tables from unimorph.org (Kirov et al., 2016), we create datasets as follows. Each training set consists of 12,000 samples in the high-resource source language and $n_t \in \{50, 200\}$ samples in the low-resource target language. We create target language dev and test sets of sizes 1600 and 10,000, respectively.[3] For Romance and Arabic, we create learning curves for $n_t \in \{100, 400, 800, 1600, 3200, 6400, 12000\}$. Due to the data available to us, we use *only* verbs for the Romance and Uralic language families, but nouns, verbs and adjectives for the Slavic language family and Arabic. Lemmata and inflections are randomly selected from all available paradigms.

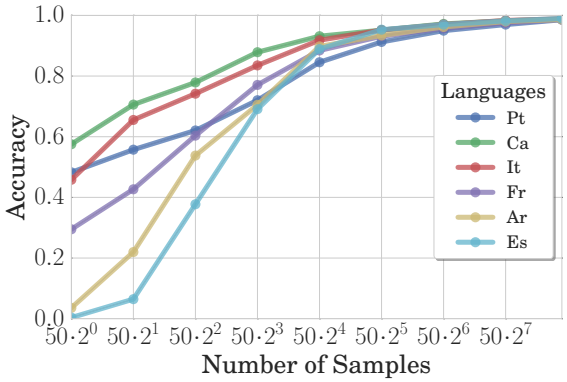


Figure 2: Learning curves showing the accuracy on Spanish test when training on language $\lambda \in \{$PT, CA, IT, FR, AR, ES$\}$. Except for λ=ES, each model is trained on 12,000 samples from λ and "Number of Samples" (x-axis) of Spanish.

Results and Discussion. Tab. 3 shows the effectiveness of transfer learning. There are *two* baselines. (i) "0": no transfer, i.e., we consider only in-domain data; (ii) "AR": Arabic, which is unrelated to all target languages.

With the exception of the 200 sample case of ET→SME, cross-lingual transfer is always better than the two baselines; the maximum improvement is 0.58 (0.58 vs. 0.00) in accuracy for the 50 sample case of CA→ES. More closely related source languages improve performance more than distant ones. French, the Romance language least similar to Spanish, performs worst for →ES. For the target language Macedonian, Bulgarian provides most benefit. This can again be explained by similarity: Bulgarian is closer to Macedonian than the other languages in this group. The best result for Ukrainian is RU→UK. Unlike Polish and Slowenian, Russian is the only language in this group that uses the same script as Ukrainian, showing

[3] For Estonian, we use 7094 (not 12,000) train and 5000 (not 10,000) test samples as more data is unavailable.

		Romance						Slavic I					Slavic II					Uralic I					Uralic II				
source		0	AR	PT	CA	IT	FR	0	AR	RU	BG	CS	0	AR	RU	PL	SL	0	AR	FI	HU	ET	0	AR	FI	HU	SME
target		→ES						→MK					→UK					→SME					→ET				
50	acc ↑	0.00	0.04	0.48	**0.58**	0.46	0.29	0.00	0.00	0.23	**0.47**	0.13	0.01	0.01	**0.47**	0.16	0.07	0.00	0.01	**0.07**	0.05	0.03	0.02	0.01	**0.35**	0.21	0.17
	ED ↓	5.42	4.06	0.85	**0.80**	1.15	1.82	5.71	5.59	1.61	**0.87**	2.32	5.23	4.80	**0.77**	2.14	3.12	6.21	5.47	**2.88**	3.46	3.71	4.50	4.51	**1.55**	2.19	2.60
200	acc ↑	0.38	0.54	0.62	**0.78**	0.74	0.60	0.21	0.40	0.62	**0.77**	0.57	0.16	0.21	**0.64**	0.55	0.50	0.13	0.24	0.26	**0.28**	0.13	0.34	0.53	**0.74**	0.71	0.66
	ED ↓	1.37	0.87	0.57	**0.39**	0.44	0.82	1.93	1.12	0.68	**0.36**	0.72	2.09	1.60	**0.49**	0.73	0.82	2.94	1.89	1.78	**1.61**	2.46	1.47	0.98	**0.41**	0.48	0.62

Table 3: Accuracy (acc; the higher the better; indicated by ↑) and edit distance (ED; the lower the better; indicated by ↓) of cross-lingual transfer learning for paradigm completion. The target language is indicated by "→", e.g., it is Spanish for "→ES". Sources are indicated in the row "source"; "0" is the monolingual case. Except for Estonian, we train on $n_s = 12{,}000$ source samples and $n_t \in \{50, 200\}$ target samples (as indicated by the row). There are *two* baselines in the table. (i) "0": no transfer, i.e., we consider only in-domain data; (ii) "AR": the Semitic language Arabic is unrelated to all target languages and functions as a dummy language that is unlikely to provide relevant information. All languages are denoted using the official codes (SME=Northern Sami).

the importance of the alphabet for transfer. Still, the results also demonstrate that transfer works across alphabets (although not as well); this suggests that similar embeddings for similar characters have been learned. Finnish is the language that is closest to Estonian and it again performs best as a source language for Estonian. For Northern Sami, transfer works least well, probably because the distance between sources and target is largest in this case. The distance of the Sami languages from the Finnic (Estonian, Finnish) and Ugric (Hungarian) languages is much larger than the distances within Romance and within Slavic. However, even for Northern Sami, the worst performing language, adding an additional language is still always beneficial compared to the monolingual baseline.

Learning curves for Romance and Arabic further support our finding that language similarity is important. In Fig. 2, knowledge is transferred to Spanish, and a baseline – a model trained *only* on Spanish data – shows the accuracy obtained without any transfer learning. Here, Catalan and Italian help the most, followed by Portuguese, French and, finally, Arabic. This corresponds to the order of lexical similarity with Spanish, except for the performance of Portuguese (cf. Tab. 2). A possible explanation is the potentially confusing overlap of lemmata between the two languages – cf. discussion in the next subsection. That the transfer learning setup improves performance for the unrelated language Arabic as source is at first surprising. However, adding new samples to a small training set helps prevent overfitting (e.g., rote memorization) even if the source is a morphologically unrelated language; effectively acting as a regularizer.

Following (Kann and Schütze, 2016b) we did not use standard regularizers. To verify that the effect of Arabic is mainly a regularization effect, we ran a small monolingual experiment on ES (200 setting) with dropout 0.5 (Srivastava et al., 2014). The resulting accuracy is 0.57, very similar to the comparable Arabic number of 0.54 in the table. The accuracy for dropout and 50 ES samples stays at 0.00, showing that in extreme low-resource settings an unrelated language might be preferable to a standard regularizer.

Error Analysis for Romance. Even for only 50 Spanish instances, many inflections are correctly produced in transfer. For, e.g., (*criar*, 3PlFutSbj) ↦ *criaren*, model outputs are: fr: *criaren*, ca: *criaren*, es: *crntaron*, it: *criaren*, ar: *ecriren*, pt: *criaren* (all correct except for the two baselines). Many errors involve accents, e.g., (*contrastar*, 2PlFutInd) ↦ *contrastaréis*; model outputs are: fr: *contrastareis*, ca: *contrastareis*, es: *conterarían*, it: *contrastareis*, ar: *contastarías*, pt: *contrastareis*. Some inflected forms are produced incorrectly by all systems, mainly because they apply the inflectional rules of the source language directly to the target. Finally, the output of the model trained on Portuguese contains a class of errors that are unlike those of other systems. Example: (*contraatacar*, 1SgCond) ↦ *contraatacaría* with the following solutions: fr: *contratacaríam*, ca: *contraatacaría*, es: *concarnar*, it: *contratacé*, ar: *cuntataría* and pt: *contra-atacaría*. The Portuguese model inserts "-" because Portuguese train data contains *contraatacar* and "-" appears in its inflected form. Thus, it seems that shared lemmata between the high-resource source language and the low-resource target language hurt our model's performance.[4] An

[4]To investigate this in more detail we retrain the Portuguese model with 50 Spanish samples, but exclude all lemmata that appear in Spanish train/dev/test, resulting in only 3695

		PT	CA	IT	CA&PT	CA&IT
		→ES				
50	acc ↑	0.48	**0.58**	0.46	0.56	**0.58**
	ED ↓	0.85	0.80	1.15	**0.67**	0.82
200	acc ↑	0.62	0.78	0.74	0.77	**0.79**
	ED ↓	0.47	0.39	0.44	0.34	**0.31**

Table 4: Results for transfer from pairs of source languages to ES. Results from single languages are repeated for comparison.

		0	PT	CA	IT	FR	AR
		→ES					
one shot	acc ↑	0.00	**0.44**	0.39	0.23	0.13	0.00
	ED ↓	6.26	**1.01**	1.27	1.83	2.87	7.00
zero shot	acc ↑	0.00	**0.14**	0.08	0.01	0.02	0.00
	ED ↓	7.18	**1.95**	1.99	3.12	4.27	7.50

Table 5: Results for one-shot and zero-shot transfer learning. Formatting is the same as for Tab. 3. We still use $n_s = 12000$ source samples. In the one-shot (resp. zero-shot) case, we observe *exactly one form* (resp. *zero forms*) for each tag in the target language at training time.

example for the generally improved performance across languages for 200 Spanish training samples is (*contrastar*, 2PIIndFut) ↦ *contrastaréis*: all models now produce the correct form.

5.2 Exp. 2: Multiple Source Languages

We now want to investigate the effect of multiple source languages.

Data. Our experimental setup is similar to §5.1: we use the same dev, test and low-resource train sets as before. However, we limit this experiment to the Romance language family and the high-resource train data consists of samples from *two different* source languages at once. Choosing those which have the highest accuracies on their own, we experiment with the following pairs: CA&PT, as well as CA&IT. In order to keep all experiments easily comparable, we use *half* of each source language's data, again ending up with a total of 12,000 high-resource samples.

Results and Discussion. Results are shown in Tab. 4. Training on two source languages improves over training on a single one. Increases in accuracy are minor, but edit distance is reduced by up to 0.13 (50 low-resource samples) and 0.08 (200 low-resource samples). That using data from multiple languages is beneficial might be due to a weaker tendency of the final model to adapt wrong rules from the source language, since different alternatives are presented during training.

5.3 Exp. 3: Zero-Shot/One-Shot Transfer

In §5.1, we investigated the relationship between in-domain (target) training set size and performance. Here, we look at the extreme case of training set sizes 1 (one-shot) and 0 (zero-shot) for a tag. We train our model on *a single* sample for *half* of the tags appearing in the low-resource language, i.e., if $\mathcal{T}_\ell$ is the set of morphological tags for the target language, train set size is $|\mathcal{T}_\ell|/2$. As before, we add 12,000 source samples.

We report *one-shot accuracy* (resp. *zero-shot accuracy*), i.e., the accuracy for samples with a tag that has been seen once (resp. never) during training. Note that the model has seen the *individual subtags* each tag is composed of.[5]

Data. Now, we use the same dev, test and high-resource train sets as in §5.1. However, the low-resource data is created in the way specified above. To remove a potentially confounding variable, we impose the condition that no two training samples belong to the same lemma.

Results and Discussion. Tab. 5 shows that the Spanish and Arabic systems do not learn anything useful for either half of the tags. This is not surprising as there is not enough Spanish data for the system to generalize well and Arabic does not contribute exploitable information. The systems trained on French and Italian, in contrast, get a non-zero accuracy for the zero-shot case as well as 0.13 and 0.23, respectively, in the one-shot case. This shows that a single training example is sometimes sufficient for successful generation although generalization to tags never observed is rarely possible. Catalan and Portuguese show the best performance in both settings; this is intuitive since they are the languages closest to the target (cf. Tab. 2). In fact, adding Portuguese to the training data yields an absolute increase in accuracy of 0.44 (0.44 vs. 0.00) for one-shot and 0.14 (0.14 vs. 0.00) for zero-shot with corresponding improvements in edit distance.

Overall, this experiment shows that with transfer learning from a closely related language the per-

training samples. Accuracy on test *increases* by 0.09 despite the reduced size of the training set.

[5] It is very unlikely that due to random selection a subtag will not be in train; this case did not occur in our experiments.

formance of zero-shot morphological generation improves over the monolingual approach, and, in the one-shot setting, it is possible to generate the right form nearly half the time.

5.4 Exp. 4: True Transfer vs. Other Effects

We would like to separate the effects of regularization that we saw for Arabic from true transfer.

To this end, we generate a random cipher (i.e., a function $\gamma : \Sigma \cup \mathcal{S} \mapsto \Sigma \cup \mathcal{S}$) and apply it to all word forms and morphological tags of the high-resource train set; target language data are not changed. Ciphering makes it harder to learn true "linguistic" transfer of morphology. Consider the simplest case of transfer: an identical mapping in two languages, e.g., (*visitar*, 1SgPresInd) $\mapsto$ *visito* in both Portuguese and Spanish. If we transform Portuguese using the cipher γ(iostv...) = kltqa..., then *visito* becomes *aktkql* in Portuguese and its tag becomes similarly unrecognizable as being identical to the Spanish tag 1SgPresInd. Our intuition is that ciphering will disrupt transfer of morphology.[6] On the other hand, the regularization effect we observed with Arabic should still be effective.

Data. We use the Portuguese-Spanish and Arabic-Spanish data from §5.1. We generate a random cipher and apply it to morphological tags and word forms for Portuguese and Arabic. The language tags are kept unchanged. Spanish is also not changed. For comparability with Tab. 3, we use the same dev and test sets as before.

Results and Discussion. Tab. 6 shows that performance of PT→ES drops a lot: from 0.48 to 0.09 for 50 samples and from 0.62 to 0.54 for 200 samples. This is because there are no overt similarities between the two languages left after applying the cipher, e.g., the two previously identical forms *visito* are now different.

The impact of ciphering on AR→ES varies: slightly improved in one case (0.54 vs. 0.56), slightly worse in three cases. We also apply the cipher to the tags and Arabic and Spanish share subtags, e.g., Sg. Just the knowledge that something is a subtag is helpful because subtags must not be generated as part of the output. We can explain the tendency of ciphering to decrease performance on AR→ES by the "masking" of common subtags.

[6]Note that ciphered input is much harder than transfer between two alphabets (Latin/Cyrillic) because it creates ambiguous input. In the example, Spanish "i" is totally different from Portuguese "i" (which is really "k"), but the model must use the same representation.

		0→ES	PT→ES		AR→ES	
			orig	ciph	orig	ciph
50	acc ↑	0.00	0.48	0.09	0.04	0.02
	ED ↓	5.42	0.85	3.25	4.06	4.62
200	acc ↑	0.38	0.62	0.54	0.54	0.56
	ED ↓	1.37	0.57	0.95	0.87	0.93

Table 6: Results for ciphering. "0→ES" and "orig" are original results, copied from Tab. 3; "ciph" is the result after the cipher has been applied.

For 200 samples and ciphering, there is no clear difference in performance between Portuguese and Arabic. However, for 50 samples and ciphering, Portuguese (0.09) seems to perform better than Arabic (0.02) in accuracy. Portuguese uses suffixation for inflection whereas Arabic is templatic and inflectional changes are not limited to the end of the word. This difference is not affected by ciphering. Perhaps even ciphered Portugese lets the model learn better that the beginnings of words just need to be copied. For 200 samples, the Spanish dataset may be large enough, so that ciphered Portuguese no longer helps in this regard.

Comparing no transfer with transfer from a ciphered language to Spanish, we see large performance gains, at least for the 200 sample case: 0.38 (0→ES) vs. 0.54 (PT→ES) and 0.56 (AR→ES). This is evidence that our conjecture is correct that the baseline Arabic mainly acts as a regularizer that prevents the model from memorizing the training set and therefore improves performance. So performance improves even though no true transfer of morphological knowledge takes place.

6 Related Work

Cross-lingual transfer learning has been used for many tasks, e.g., automatic speech recognition (Huang et al., 2013), parsing (Cohen et al., 2011; Søgaard, 2011; Naseem et al., 2012; Ammar et al., 2016), language modeling (Tsvetkov et al., 2016), entity recognition (Wang and Manning, 2014b) and machine translation (Johnson et al., 2016; Ha et al., 2016).

One straightforward method is to translate datasets and then train a monolingual model (Fortuna and Shawe-Taylor, 2005; Olsson et al., 2005). Also, aligned corpora have been used to project information from annotations in one language to another (Yarowsky et al., 2001; Padó and Lapata, 2005). The drawback is that machine translation

errors cause errors in the target. Therefore, alternative methods have been proposed, e.g., to port a model trained on the source language to the target language (Shi et al., 2010).

In the realm of morphology, Buys and Botha (2016) recently adapted methods for the training of POS taggers to learn weakly supervised morphological taggers with the help of parallel text. Snyder and Barzilay (2008a, 2008b) developed a non-parametric Bayesian model for morphological segmentation. They performed identification of cross-lingual abstract morphemes and segmentation simultaneously and reported, similar to us, best results for related languages.

Work on **paradigm completion** has recently been encouraged by the SIGMORPHON 2016 shared task on morphological reinflection (Cotterell et al., 2016a). Some work first applies an unsupervised alignment model to source and target string pairs and then learns a string-to-string mapping (Durrett and DeNero, 2013; Nicolai et al., 2015), using, e.g., a semi-Markov conditional random field (Sarawagi and Cohen, 2004). Encoder-decoder RNNs (Aharoni et al., 2016; Faruqui et al., 2016; Kann and Schütze, 2016b), a method which our work further develops for the cross-lingual scenario, define the current state of the art.

Encoder-decoder RNNs were developed in parallel by Cho et al. (2014) and Sutskever et al. (2014) for machine translation and extended by Bahdanau et al. (2015) with an attention mechanism, supporting better generalization. They have been applied to NLP tasks like speech recognition (Graves and Schmidhuber, 2005; Graves et al., 2013), parsing (Vinyals et al., 2015) and segmentation (Kann et al., 2016).

More recently, a number of papers have used encoder-decoder RNNs in *multitask and transfer learning settings*; this is mainly work in machine translation: (Dong et al., 2015; Zoph and Knight, 2016; Chu et al., 2017; Johnson et al., 2016; Luong et al., 2016; Firat et al., 2016; Ha et al., 2016), *inter alia*. Each of these papers has both similarities and differences with our approach. (i) Most train several distinct models whereas we train a *single model* on input augmented with an explicit encoding of the language (similar to (Johnson et al., 2016)). (ii) Let k and m be the number of different input and output languages. We address the case $k \in \{1, 2, 3\}$ and $m = k$. Other work has addressed cases with $k > 3$ or $m > 3$; this would be an interesting avenue of future research for paradigm completion. (iii) Whereas training RNNs in machine translation is hard, we only experienced one difficult issue in our experiments (due to the low-resource setting): regularization. (iv) Some work is word- or subword-based, our work is character-based. The same way that similar word embeddings are learned for the inputs *cow* and *vache* (French for "cow") in machine translation, we expect similar embeddings to be learned for similar Cyrillic/Latin characters. (v) Similar to work in machine translation, we show that zero-shot (and, by extension, one-shot) learning is possible.

(Ha et al., 2016) (which was developed in parallel to our transfer model although we did not prepublish our paper on arxiv) is most similar to our work. Whereas Ha et al. (2016) address machine translation, we focus on the task of paradigm completion in low-resource settings and establish the state of the art for this problem.

7 Conclusion

We presented a cross-lingual transfer learning method for paradigm completion, based on an RNN encoder-decoder model. Our experiments showed that information from a high-resource language can be leveraged for paradigm completion in a related low-resource language. Our analysis indicated that the degree to which the source language data helps for a certain target language depends on their relatedness. Our method led to significant improvements in settings with limited training data – up to 58% absolute improvement in accuracy – and, thus, enables the use of state-of-the-art models for paradigm completion in low-resource languages.

8 Future Work

In the future, we want to develop methods to make better use of languages with different alphabets or morphosyntactic features, in order to increase the applicability of our knowledge transfer method.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. We are grateful to Siemens and Volkswagenstiftung for their generous support. This research would not have been possible without the organizers of the SIGMORPHON shared task, especially John Sylak-Glassman and Christo Kirov, who created the resources we use.

References

Daniel Abondolo. 2015. *The Uralic Languages*. Routledge.

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *SIGMORPHON*.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *EACL*.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Jan Buys and Jan A Botha. 2016. Cross-lingual morphological tagging for low-resource languages. In *ACL*.

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

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint 1409.1259* .

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint 1701.03214* .

Shay B Cohen, Dipanjan Das, and Noah A Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR* 12(Aug):2493–2537.

Greville Corbett and Bernard Comrie. 2003. *The Slavonic Languages*. Routledge.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016a. The SIGMORPHON 2016 shared task—morphological reinflection. In *SIGMORPHON*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015. Modeling word forms using latent underlying morphs and phonology. *TACL* 3:433–447.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016b. Morphological smoothing and extrapolation of word embeddings. In *ACL*.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Analysis of morph-based speech recognition and the modeling of out-of-vocabulary words across languages. In *NAACL-HLT*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL-IJCNLP*.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *NAACL*.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *ACL*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *NAACL*.

Orhan Firat, KyungHyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *CoRR* abs/1601.01073.

Blaz Fortuna and John Shawe-Taylor. 2005. The use of machine translation tools for cross-lingual text mining. In *ICML Workshop on Learning with Multiple Views*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *IEEE*.

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

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint 1611.04798* .

Martin Harris and Nigel Vincent. 2003. *The Romance languages*. Routledge.

Robert Hetzron. 2013. *The Semitic Languages*. Routledge.

Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and n Gong. 2013. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *IEEE*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR* abs/1611.04558.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *EMNLP*.

Katharina Kann and Hinrich Schütze. 2016a. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*.

Katharina Kann and Hinrich Schütze. 2016b. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *ACL*.

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large scale parsing and normalization of wiktionary morphological paradigms. In *LREC*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR* abs/1504.00941.

M Paul Lewis, editor. 2009. *Ethnologue: Languages of the World*. SIL International, Dallas, Texas, 16 edition.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Karthik Narasimhan, Damianos Karakos, Richard Schwartz, Stavros Tsakalidis, and Regina Barzilay. 2014. Morphological segmentation for keyword spotting. In *EMNLP*.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *ACL*.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *NAACL*.

J Scott Olsson, Douglas W Oard, and Jan Hajič. 2005. Cross-language text classification. In *ACM SIGIR*.

Sebastian Padó and Mirella Lapata. 2005. Cross-linguistic projection of role-semantic information. In *HLT/EMNLP*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC*.

Sunita Sarawagi and William W Cohen. 2004. Semi-markov conditional random fields for information extraction. In *NIPS*.

Wolfgang Seeker and Özlem Çetinoğlu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *TACL* 3:359–373.

Lei Shi, Rada Mihalcea, and Mingjun Tian. 2010. Cross language text classification by model translation and semi-supervised learning. In *EMNLP*.

Benjamin Snyder and Regina Barzilay. 2008a. Cross-lingual propagation for morphological analysis. In *AAAI*.

Benjamin Snyder and Regina Barzilay. 2008b. Unsupervised multilingual learning for morphological segmentation. In *ACL-HLT*.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *ACL-HLT*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

John Sylak-Glassman. 2016. The composition and use of the universal morphological feature schema (unimorph schema). Technical report, Department of Computer Science, Johns Hopkins University.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *ACL-IJCNLP*.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *NAACL-HLT*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*.

Mengqiu Wang and Christopher D Manning. 2014a. Cross-lingual projected expectation regularization for weakly supervised learning. *TACL* 2:55–66.

Mengqiu Wang and Christopher D Manning. 2014b. Cross-lingual pseudo-projected expectation regularization for weakly supervised learning. *TACL* 2:55–66.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *EMNLP*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *HLT*.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *NAACL-HLT*.

Morphological Inflection Generation with Hard Monotonic Attention

Roee Aharoni & Yoav Goldberg
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
{roee.aharoni,yoav.goldberg}@gmail.com

Abstract

We present a neural model for morphological inflection generation which employs a hard attention mechanism, inspired by the nearly-monotonic alignment commonly found between the characters in a word and the characters in its inflection. We evaluate the model on three previously studied morphological inflection generation datasets and show that it provides state of the art results in various setups compared to previous neural and non-neural approaches. Finally we present an analysis of the continuous representations learned by both the hard and soft attention (Bahdanau et al., 2015) models for the task, shedding some light on the features such models extract.

1 Introduction

Morphological inflection generation involves generating a target word (e.g. "härtestem", the German word for "hardest"), given a source word (e.g. "hart", the German word for "hard") and the morpho-syntactic attributes of the target (POS=adjective, gender=masculine, type=superlative, etc.).

The task is important for many down-stream NLP tasks such as machine translation, especially for dealing with data sparsity in morphologically rich languages where a lemma can be inflected into many different word forms. Several studies have shown that translating into lemmas in the target language and then applying inflection generation as a post-processing step is beneficial for phrase-based machine translation (Minkov et al., 2007; Toutanova et al., 2008; Clifton and Sarkar, 2011; Fraser et al., 2012; Chahuneau et al., 2013) and more recently for neural machine translation (García-Martínez et al., 2016).

The task was traditionally tackled with hand engineered finite state transducers (FST) (Koskenniemi, 1983; Kaplan and Kay, 1994) which rely on expert knowledge, or using trainable weighted finite state transducers (Mohri et al., 1997; Eisner, 2002) which combine expert knowledge with data-driven parameter tuning. Many other machine-learning based methods (Yarowsky and Wicentowski, 2000; Dreyer and Eisner, 2011; Durrett and DeNero, 2013; Hulden et al., 2014; Ahlberg et al., 2015; Nicolai et al., 2015) were proposed for the task, although with specific assumptions about the set of possible processes that are needed to create the output sequence.

More recently, the task was modeled as neural sequence-to-sequence learning over character sequences with impressive results (Faruqui et al., 2016). The vanilla encoder-decoder models as used by Faruqui et al. compress the input sequence to a single, fixed-sized continuous representation. Instead, the soft-attention based sequence to sequence learning paradigm (Bahdanau et al., 2015) allows directly conditioning on the entire input sequence representation, and was utilized for morphological inflection generation with great success (Kann and Schütze, 2016b,a).

However, the neural sequence-to-sequence models require large training sets in order to perform well: their performance on the relatively small CELEX dataset is inferior to the latent variable WFST model of Dreyer et al. (2008). Interestingly, the neural WFST model by Rastogi et al. (2016) also suffered from the same issue on the CELEX dataset, and surpassed the latent variable model only when given twice as much data to train on.

We propose a model which handles the above issues by directly modeling an almost monotonic

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2004–2015
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1183

alignment between the input and output character sequences, which is commonly found in the morphological inflection generation task (e.g. in languages with concatenative morphology). The model consists of an encoder-decoder neural network with a dedicated control mechanism: in each step, the model attends to a *single* input state and either writes a symbol to the output sequence or advances the attention pointer to the next state from the bi-directionally encoded sequence, as described visually in Figure 1.

This modeling suits the natural monotonic alignment between the input and output, as the network learns to attend to the relevant inputs before writing the output which they are aligned to. The encoder is a bi-directional RNN, where each character in the input word is represented using a concatenation of a forward RNN and a backward RNN states over the word's characters. The combination of the bi-directional encoder and the controllable hard attention mechanism enables to condition the output on the entire input sequence. Moreover, since each character representation is aware of the neighboring characters, non-monotone relations are also captured, which is important in cases where segments in the output word are a result of long range dependencies in the input word. The recurrent nature of the decoder, together with a dedicated feedback connection that passes the last prediction to the next decoder step explicitly, enables the model to also condition the current output on all the previous outputs at each prediction step.

The hard attention mechanism allows the network to jointly align and transduce while using a focused representation at each step, rather then the weighted sum of representations used in the soft attention model. This makes our model Resolution Preserving (Kalchbrenner et al., 2016) while also keeping decoding time linear in the output sequence length rather than multiplicative in the input and output lengths as in the soft-attention model. In contrast to previous sequence-to-sequence work, we do not require the training procedure to also learn the alignment. Instead, we use a simple training procedure which relies on independently learned character-level alignments, from which we derive gold transduction+control sequences. The network can then be trained using straightforward cross-entropy loss.

To evaluate our model, we perform extensive experiments on three previously studied morphological inflection generation datasets: the CELEX dataset (Baayen et al., 1993), the Wiktionary dataset (Durrett and DeNero, 2013) and the SIGMORPHON2016 dataset (Cotterell et al., 2016). We show that while our model is on par with or better than the previous neural and non-neural state-of-the-art approaches, it also performs significantly better with very small training sets, being the first neural model to surpass the performance of the weighted FST model with latent variables which was specifically tailored for the task by Dreyer et al. (2008). Finally, we analyze and compare our model and the soft attention model, showing how they function very similarly with respect to the alignments and representations they learn, in spite of our model being much simpler. This analysis also sheds light on the representations such models learn for the morphological inflection generation task, showing how they encode specific features like a symbol's type and the symbol's location in a sequence.

To summarize, our contributions in this paper are three-fold:

1. We present a hard attention model for nearly-monotonic sequence to sequence learning, as common in the morphological inflection setting.

2. We evaluate the model on the task of morphological inflection generation, establishing a new state of the art on three previously-studied datasets for the task.

3. We perform an analysis and comparison of our model and the soft-attention model, shedding light on the features such models extract for the inflection generation task.

2 The Hard Attention Model

2.1 Motivation

We would like to transduce an input sequence, $x_{1:n} \in \Sigma_x^*$ into an output sequence, $y_{1:m} \in \Sigma_y^*$, where Σ_x and Σ_y are the input and output vocabularies, respectively. Imagine a machine with read-only random access to the encoding of the input sequence, and a single pointer that determines the current read location. We can then model sequence transduction as a series of pointer movement and write operations. If we assume the alignment is monotone, the machine can be simpli-

fied: the memory can be read in sequential order, where the pointer movement is controlled by a single "move forward" operation (step) which we add to the output vocabulary. We implement this behavior using an encoder-decoder neural network, with a control mechanism which determines in each step of the decoder whether to write an output symbol or promote the attention pointer the next element of the encoded input.

2.2 Model Definition

In prediction time, we seek the output sequence $y_{1:m} \in \Sigma_y^*$, for which:

$$y_{1:m} = \arg\max_{y' \in \Sigma_y^*} p(y'|x_{1:n}, f) \quad (1)$$

Where $x \in \Sigma_x^*$ is the input sequence and $f = \{f_1, \ldots, f_l\}$ is a set of attributes influencing the transduction task (in the inflection generation task these would be the desired morpho-syntactic attributes of the output sequence). Given a nearly-monotonic alignment between the input and the output, we replace the search for a sequence of letters with a sequence of actions $s_{1:q} \in \Sigma_s^*$, where $\Sigma_s = \Sigma_y \cup \{step\}$. This sequence is a series of step and write actions required to go from $x_{1:n}$ to $y_{1:m}$ according to the monotonic alignment between them (we will elaborate on the deterministic process of getting $s_{1:q}$ from a monotonic alignment between $x_{1:n}$ to $y_{1:m}$ in section 2.4). In this case we define: [1]

$$\begin{aligned} s_{1:q} &= \arg\max_{s' \in \Sigma_s^*} p(s'|x_{1:n}, f) \\ &= \arg\max_{s' \in \Sigma_s^*} \prod_{s'_i \in s'} p(s'_i|s'_1 \ldots s'_{i-1}, x_{1:n}, f) \end{aligned} \quad (2)$$

which we can estimate using a neural network:

$$s_{1:q} = \arg\max_{s' \in \Sigma_s^*} \text{NN}(x_{1:n}, f, \Theta) \quad (3)$$

where the network's parameters Θ are learned using a set of training examples. We will now describe the network architecture.

[1] We note that our model (Eq. 2) solves a different objective than (Eq 1), as it searches for the *best derivation* and not *the best sequence*. In order to accurately solve (1) we would need to marginalize over the different derivations leading to the same sequence, which is computationally challenging. However, as we see in the experiments section, the best-derivation approximation is effective in practice.

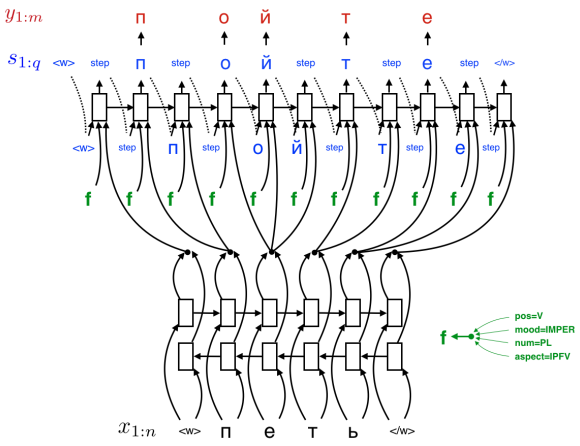


Figure 1: The hard attention network architecture. A round tip expresses concatenation of the inputs it receives. The attention is promoted to the next input element once a step action is predicted.

2.3 Network Architecture

Notation We use bold letters for vectors and matrices. We treat LSTM as a parameterized function $\mathbf{LSTM}_\theta(\mathbf{x}_1 \ldots \mathbf{x}_n)$ mapping a sequence of input vectors $\mathbf{x}_1 \ldots \mathbf{x}_n$ to a an output vector $\mathbf{h}_n$. The equations for the LSTM variant we use are detailed in the supplementary material of this paper.

Encoder For every element in the input sequence: $x_{1:n} = x_1 \ldots x_n$, we take the corresponding embedding: $\mathbf{e}_{x_1} \ldots \mathbf{e}_{x_n}$, where: $\mathbf{e}_{x_i} \in \mathbb{R}^E$. These embeddings are parameters of the model which will be learned during training. We then feed the embeddings into a bi-directional LSTM encoder (Graves and Schmidhuber, 2005) which results in a sequence of vectors: $\mathbf{x}_{1:n} = \mathbf{x}_1 \ldots \mathbf{x}_n$, where each vector $\mathbf{x}_i \in \mathbb{R}^{2H}$ is a concatenation of: $\mathbf{LSTM}_{\mathbf{forward}}(\mathbf{e}_{x_1}, \mathbf{e}_{x_2}, \ldots \mathbf{e}_{x_i})$ and $\mathbf{LSTM}_{\mathbf{backward}}(\mathbf{e}_{x_n}, \mathbf{e}_{x_{n-1}} \ldots \mathbf{e}_{x_i})$, the forward LSTM and the backward LSTM outputs when fed with $\mathbf{e}_{x_i}$.

Decoder Once the input sequence is encoded, we feed the decoder RNN, $\mathbf{LSTM}_{\mathbf{dec}}$, with three inputs at each step:

1. The current attended input, $\mathbf{x}_a \in \mathbb{R}^{2H}$, initialized with the first element of the encoded sequence, $\mathbf{x}_1$.
2. A set of embeddings for the attributes that influence the generation process, concatenated to a single vector: $\mathbf{f} = [\mathbf{f}_1 \ldots \mathbf{f}_l] \in \mathbb{R}^{F \cdot l}$.
3. $\mathbf{s}_{i-1} \in \mathbb{R}^E$, which is an embedding for the

predicted output symbol in the previous decoder step.

Those three inputs are concatenated into a single vector $\mathbf{z}_i = [\mathbf{x}_a, \mathbf{f}, \mathbf{s}_{i-1}] \in \mathbb{R}^{2H+F\cdot l+E}$, which is fed into the decoder, providing the decoder output vector: $\mathbf{LSTM_{dec}}(\mathbf{z}_1 \dots \mathbf{z}_i) \in \mathbb{R}^H$. Finally, to model the distribution over the possible actions, we project the decoder output to a vector of $|\Sigma_s|$ elements, followed by a softmax layer:

$$p(s_i = c) = softmax_c(\mathbf{W} \cdot \mathbf{LSTM_{dec}}(\mathbf{z}_1 \dots \mathbf{z}_i) + \mathbf{b}) \quad (4)$$

Control Mechanism When the most probable action is *step*, the attention is promoted so $\mathbf{x}_a$ contains the next encoded input representation to be used in the next step of the decoder. The process is demonstrated visually in Figure 1.

2.4 Training the Model

For every example: $(x_{1:n}, y_{1:m}, f)$ in the training data, we should produce a sequence of step and write actions $s_{1:q}$ to be predicted by the decoder. The sequence is dependent on the alignment between the input and the output: ideally, the network will attend to all the input characters aligned to an output character before writing it. While recent work in sequence transduction advocate jointly training the alignment and the decoding mechanisms (Bahdanau et al., 2015; Yu et al., 2016), we instead show that in our case it is worthwhile to decouple these stages and learn a hard alignment beforehand, using it to guide the training of the encoder-decoder network and enabling the use of correct alignments for the attention mechanism from the beginning of the network training phase. Thus, our training procedure consists of three stages: learning hard alignments, deriving oracle actions from the alignments, and learning a neural transduction model given the oracle actions.

Learning Hard Alignments We use the character alignment model of Sudoh et al. (2013), based on a Chinese Restaurant Process which weights single alignments (character-to-character) in proportion to how many times such an alignment has been seen elsewhere out of all possible alignments. The aligner implementation we used produces either 0-to-1, 1-to-0 or 1-to-1 alignments.

Deriving Oracle Actions We infer the sequence of actions $s_{1:q}$ from the alignments by the deterministic procedure described in Algorithm 1. An example of an alignment with the resulting oracle action sequence is shown in Figure 2, where a_4 is a 0-to-1 alignment and the rest are 1-to-1 alignments.

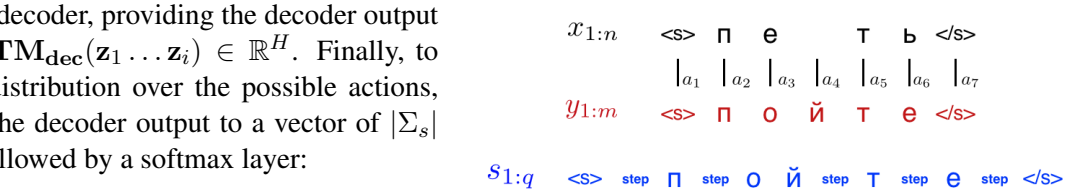


Figure 2: Top: an alignment between a lemma $x_{1:n}$ and an inflection $y_{1:m}$ as predicted by the aligner. Bottom: $s_{1:q}$, the sequence of actions to be predicted by the network, as produced by Algorithm 1 for the given alignment.

Algorithm 1 Generates the oracle action sequence $s_{1:q}$ from the alignment between $x_{1:n}$ and $y_{1:m}$

Require: a, the list of either 1-to-1, 1-to-0 or 0-to-1 alignments between $x_{1:n}$ and $y_{1:m}$

```
Initialize s as an empty sequence
for each a_i = (x_{a_i}, y_{a_i}) ∈ a do
    if a_i is a 1-to-0 alignment then
        s.append(step)
    else
        s.append(y_{a_i})
        if a_{i+1} is not a 0-to-1 alignment then
        s.append(step)
return s
```

This procedure makes sure that all the source input elements aligned to an output element are read (using the step action) before writing it.

Learning a Neural Transduction Model The network is trained to mimic the actions of the oracle, and at inference time, it will predict the actions according to the input. We train it using a conventional cross-entropy loss function per example:

$$\mathcal{L}(x_{1:n}, y_{1:m}, f, \Theta) = -\sum_{s_j \in s_{1:q}} \log softmax_{s_j}(\mathbf{d}),$$
$$\mathbf{d} = \mathbf{W} \cdot \mathbf{LSTM_{dec}}(\mathbf{z}_1 \dots \mathbf{z}_i) + \mathbf{b} \quad (5)$$

Transition System An alternative view of our model is that of a *transition system* with ADVANCE and WRITE(CH) actions, where the oracle is derived from a given hard alignment, the input is encoded using a biRNN, and the next action is determined by an RNN over the previous inputs and actions.

3 Experiments

We perform extensive experiments with three previously studied morphological inflection generation datasets to evaluate our hard attention model in various settings. In all experiments we compare our hard attention model to the best performing neural and non-neural models which were previously published on those datasets, and to our implementation of the global (soft) attention model presented by Luong et al. (2015) which we train with identical hyper-parameters as our hard-attention model. The implementation details for our models are described in the supplementary material section of this paper. The source code and data for our models is available on github.[2]

CELEX Our first evaluation is on a very small dataset, to see if our model indeed avoids the tendency to overfit with small training sets. We report exact match accuracy on the German inflection generation dataset compiled by Dreyer et al. (2008) from the CELEX database (Baayen et al., 1993). The dataset includes only 500 training examples for each of the four inflection types: 13SIA→13SKE, 2PIE→13PKE, 2PKE→z, and rP→pA which we refer to as 13SIA, 2PIE, 2PKE and rP, respectively.[3] We first compare our model to three competitive models from the literature that reported results on this dataset: the Morphological Encoder-Decoder (MED) of Kann and Schütze (2016a) which is based on the soft-attention model of Bahdanau et al. (2015), the neural-weighted FST of Rastogi et al. (2016) which uses stacked bi-directional LSTM's to weigh its arcs (NWFST), and the model of Dreyer et al. (2008) which uses a weighted FST with latent-variables structured particularly for morphological string transduction tasks (LAT). Following previous reports on this dataset, we use the same data splits as Dreyer et al. (2008), dividing the data for each inflection type into five folds, each consisting of 500 training, 1000 development and 1000 test examples. We train a separate model for each fold and report exact match accuracy, averaged over the five folds.

Wiktionary To neutralize the negative effect of very small training sets on the performance of the different learning approaches, we also evaluate our model on the dataset created by Durrett and DeNero (2013), which contains up to 360k training examples per language. It was built by extracting Finnish, German and Spanish inflection tables from Wiktionary, used in order to evaluate their system based on string alignments and a semi-CRF sequence classifier with linguistically inspired features, which we use a baseline. We also used the dataset expansion made by Nicolai et al. (2015) to include French and Dutch inflections as well. Their system also performs an align-and-transduce approach, extracting rules from the aligned training set and applying them in inference time with a proprietary character sequence classifier. In addition to those systems we also compare to the results of the recent neural approach of Faruqui et al. (2016), which did not use an attention mechanism, and Yu et al. (2016), which coupled the alignment and transduction tasks.

SIGMORPHON As different languages show different morphological phenomena, we also experiment with how our model copes with these various phenomena using the morphological inflection dataset from the SIGMORPHON2016 shared task (Cotterell et al., 2016). Here the training data consists of ten languages, with five morphological system types (detailed in Table 3): Russian (RU), German (DE), Spanish (ES), Georgian (GE), Finnish (FI), Turkish (TU), Arabic (AR), Navajo (NA), Hungarian (HU) and Maltese (MA) with roughly 12,800 training and 1600 development examples per language. We compare our model to two soft attention baselines on this dataset: MED (Kann and Schütze, 2016b), which was the best participating system in the shared task, and our implementation of the global (soft) attention model presented by Luong et al. (2015).

4 Results

In all experiments, for both the hard and soft attention models we implemented, we report results using an ensemble of 5 models with different random initializations by using majority voting on the final sequences the models predicted, as proposed by Kann and Schütze (2016a). This was done to perform fair comparison to the models of Kann and Schütze (2016a,b); Faruqui et al. (2016) which we compare to, that also perform a similar ensem-

[2] https://github.com/roeeaharoni/morphological-reinflection

[3] The acronyms stand for: 13SIA=1st/3rd person, singular, indefinite, past;13SKE=1st/3rd person, subjunctive, present; 2PIE=2nd person, plural, indefinite, present;13PKE=1st/3rd person, plural, subjunctive, present; 2PKE=2nd person, plural, subjunctive, present; z=infinitive; rP=imperative, plural; pA=past participle.

	13SIA	2PIE	2PKE	rP	Avg.
MED (Kann and Schütze, 2016a)	83.9	95	87.6	84	87.62
NWFST (Rastogi et al., 2016)	86.8	94.8	87.9	81.1	87.65
LAT (Dreyer et al., 2008)	**87.5**	93.4	87.4	84.9	88.3
Soft	83.1	93.8	88	83.2	87
Hard	85.8	**95.1**	**89.5**	**87.2**	**89.44**

Table 1: Results on the CELEX dataset

	DE-N	DE-V	ES-V	FI-NA	FI-V	FR-V	NL-V	Avg.
Durrett and DeNero (2013)	88.31	94.76	99.61	92.14	97.23	98.80	90.50	94.47
Nicolai et al. (2015)	88.6	97.50	99.80	93.00	**98.10**	**99.20**	96.10	96.04
Faruqui et al. (2016)	88.12	**97.72**	**99.81**	95.44	97.81	98.82	96.71	96.34
Yu et al. (2016)	87.5	92.11	99.52	95.48	**98.10**	98.65	95.90	95.32
Soft	88.18	95.62	99.73	93.16	97.74	98.79	96.73	95.7
Hard	**88.87**	97.35	99.79	**95.75**	98.07	99.04	**97.03**	**96.55**

Table 2: Results on the Wiktionary datasets

	suffixing+stem changes			circ.	suffixing+agg.+v.h.			c.h.	templatic		
	RU	DE	ES	GE	FI	TU	HU	NA	AR	MA	Avg.
MED	91.46	95.8	98.84	98.5	95.47	98.93	96.8	91.48	**99.3**	**88.99**	95.56
Soft	92.18	96.51	98.88	**98.88**	**96.99**	**99.37**	**97.01**	**95.41**	**99.3**	88.86	**96.34**
Hard	**92.21**	**96.58**	**98.92**	98.12	95.91	97.99	96.25	93.01	98.77	88.32	95.61

Table 3: Results on the SIGMORPHON 2016 morphological inflection dataset. The text above each language lists the morphological phenomena it includes: circ.=circumfixing, agg.=agglutinative, v.h.=vowel harmony, c.h.=consonant harmony

bling technique.

On the low resource setting (CELEX), our hard attention model significantly outperforms both the recent neural models of Kann and Schütze (2016a) (MED) and Rastogi et al. (2016) (NWFST) and the morphologically aware latent variable model of Dreyer et al. (2008) (LAT), as detailed in Table 1. In addition, it significantly outperforms our implementation of the soft attention model (Soft). It is also, to our knowledge, the first model that surpassed in overall accuracy the latent variable model on this dataset. We attribute our advantage over the soft attention models to the ability of the hard attention control mechanism to harness the monotonic alignments found in the data. The advantage over the FST models may be explained by our conditioning on the entire output history which is not available in those models. Figure 3 plots the train-set and dev-set accuracies of the soft and hard attention models as a function of the training epoch. While both models perform similarly on the train-set (with the soft attention model fitting it slightly faster), the hard attention model performs significantly better on the dev-set. This shows the soft attention model's tendency to overfit on the small dataset, as it is not enforcing the monotonic assumption of the hard attention model.

On the large training set experiments (Wiktionary), our model is the best performing model on German verbs, Finnish nouns/adjectives and Dutch verbs, resulting in the highest reported average accuracy across all inflection types when compared to the four previous neural and non-neural state of the art baselines, as detailed in Table 2. This shows the robustness of our model also with large amounts of training examples, and the advantage the hard attention mechanism provides over the encoder-decoder approach of Faruqui et al. (2016) which does not employ an attention mechanism. Our model is also significantly more accurate than the model of Yu et al. (2016), which shows the advantage of using independently learned alignments to guide the network's attention from the beginning of the training process. While our soft-attention implementation outperformed the models of Yu et al. (2016) and Durrett and DeNero (2013), it still performed inferiorly to the hard attention model.

As can be seen in Table 3, on the SIGMORPHON 2016 dataset our model performs

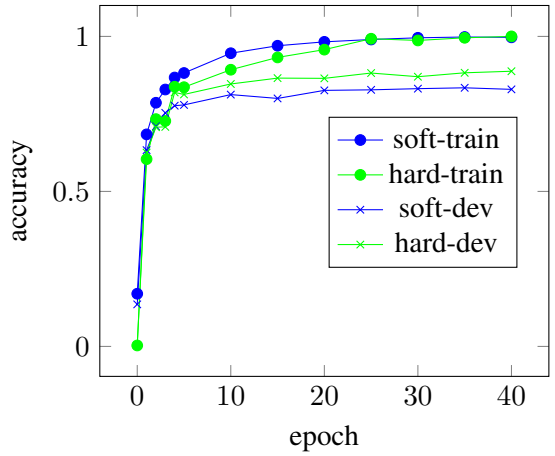


Figure 3: Learning curves for the soft and hard attention models on the first fold of the CELEX dataset

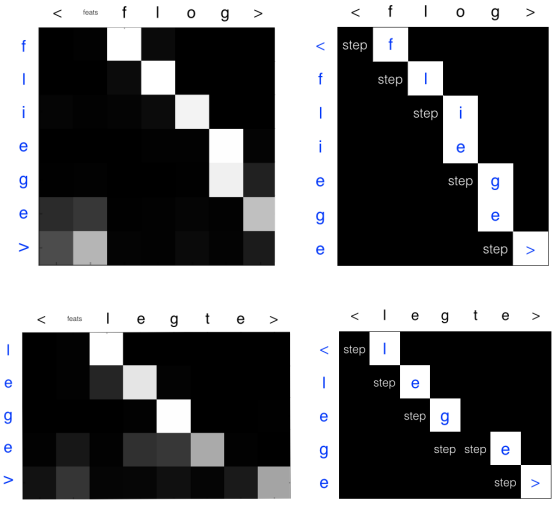


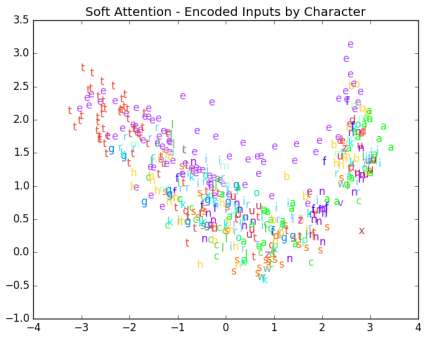
Figure 4: A comparison of the alignments as predicted by the soft attention (left) and the hard attention (right) models on examples from CELEX.

better than both soft-attention baselines for the suffixing+stem-change languages (Russian, German and Spanish) and is slightly less accurate than our implementation of the soft attention model on the rest of the languages, which is now the best performing model on this dataset to our knowledge. We explain this by looking at the languages from a linguistic typology point of view, as detailed in Cotterell et al. (2016). Since Russian, German and Spanish employ a suffixing morphology with internal stem changes, they are more suitable for monotonic alignment as the transformations they need to model are the addition of suffixes and changing characters in the stem. The rest of the languages in the dataset employ more context sensitive morphological phenomena like vowel harmony and consonant harmony, which require to model long range dependencies in the input sequence which better suits the soft attention mechanism. While our implementation of the soft attention model and MED are very similar modelwise, we hypothesize that our soft attention model results are better due to the fact that we trained the model for 100 epochs and picked the best performing model on the development set, while the MED system was trained for a fixed amount of 20 epochs (although trained on more data – both train and development sets).

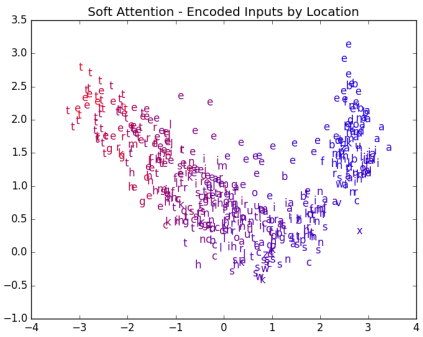
5 Analysis

The Learned Alignments In order to see if the alignments predicted by our model fit the monotonic alignment structure found in the data, and whether are they more suitable for the task when compared to the alignments found by the soft attention model, we examined alignment predictions of the two models on examples from the development portion of the CELEX dataset, as depicted in Figure 4. First, we notice the alignments found by the soft attention model are also monotonic, supporting our modeling approach for the task. Figure 4 (bottom-right) also shows how the hard-attention model performs deletion (legte→lege) by predicting a sequence of two step operations. Another notable morphological transformation is the one-to-many alignment, found in the top example: flog→fliege, where the model needs to transform a character in the input, *o*, to two characters in the output, *ie*. This is performed by two consecutive *write* operations after the *step* operation of the relevant character to be replaced. Notice that in this case, the soft attention model performs a different alignment by aligning the character *i* to *o* and the character *g* to the sequence *eg*, which is not the expected alignment in this case from a linguistic point of view.

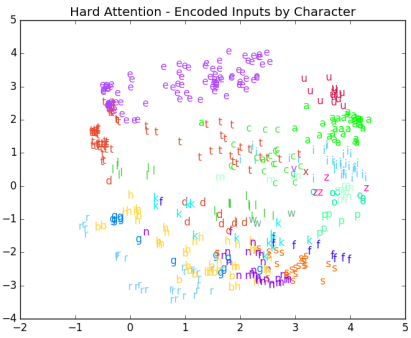
The Learned Representations How does the soft-attention model manage to learn nearly-perfect monotonic alignments? Perhaps the the network learns to encode the sequential position as part of its encoding of an input element? More generally, what information is encoded by the soft and hard alignment encoders? We selected 500 random encoded characters-in-context from input



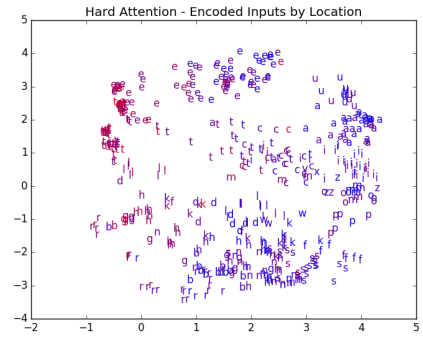
(a) Colors indicate which character is encoded.



(c) Colors indicate the character's position.



(b) Colors indicate which character is encoded.



(d) Colors indicate the character's position.

Figure 5: SVD dimension reduction to 2D of 500 character representations in context from the encoder, for both the soft attention (top) and hard attention (bottom) models.

words in the CELEX development set, where every encoded representation is a vector in $\mathbb{R}^{200}$. Since those vectors are outputs from the bi-LSTM encoders of the models, every vector of this form carries information of the specific character with its entire context. We project these encodings into 2-D using SVD and plot them twice, each time using a different coloring scheme. We first color each point according to the character it represents (Figures 5a, 5b). In the second coloring scheme (Figures 5c, 5d), each point is colored according to the character's sequential position in the word it came from, blue indicating positions near the beginning of the word, and red positions near its end.

While both models tend to cluster representations for similar characters together (Figures 5a, 5b), the hard attention model tends to have much more isolated character clusters. Figures 5c, 5d show that both models also tend to learn representations which are sensitive to the position of the character, although it seems that here the soft attention model is more sensitive to this information as its coloring forms a nearly-perfect red-to-blue transition on the X axis. This may be explained by the soft-attention mechanism encouraging the encoder to encode positional information in the input representations, which may help it to predict better attention scores, and to avoid collisions when computing the weighted sum of representations for the context vector. In contrast, our hard-attention model has other means of obtaining the position information in the decoder using the step actions, and for that reason it does not encode it as strongly in the representations of the inputs. This behavior may allow it to perform well even with fewer examples, as the location information is represented more explicitly in the model using the step actions.

6 Related Work

Many previous works on inflection generation used machine learning methods (Yarowsky and Wicentowski, 2000; Dreyer and Eisner, 2011; Durrett and DeNero, 2013; Hulden et al., 2014; Ahlberg et al., 2015; Nicolai et al., 2015) with assumptions about the set of possible processes needed to create the output word. Our work was mainly inspired by Faruqui et al. (2016) which trained an independent encoder-decoder neural

network for every inflection type in the training data, alleviating the need for feature engineering. Kann and Schütze (2016b,a) tackled the task with a *single* soft attention model (Bahdanau et al., 2015) for all inflection types, which resulted in the best submission at the SIGMORPHON 2016 shared task (Cotterell et al., 2016). In another closely related work, Rastogi et al. (2016) model the task with a WFST in which the arc weights are learned by optimizing a global loss function over all the possible paths in the state graph, while modeling contextual features with bi-directional LSTMS. This is similar to our approach, where instead of learning to mimic a single greedy alignment as we do, they sum over all possible alignments. While not committing to a single greedy alignment could in theory be beneficial, we see in Table 1 that—at least for the low resource scenario—our greedy approach is more effective in practice. Another recent work (Kann et al., 2016) proposed performing neural multi-source morphological reinflection, generating an inflection from several source forms of a word.

Previous works on neural sequence transduction include the RNN Transducer (Graves, 2012) which uses two independent RNN's over monotonically aligned sequences to predict a distribution over the possible output symbols in each step, including a null symbol to model the alignment. Yu et al. (2016) improved this by replacing the null symbol with a dedicated learned transition probability. Both models are trained using a forward-backward approach, marginalizing over all possible alignments. Our model differs from the above by learning the alignments independently, thus enabling a dependency between the encoder and decoder. While providing better results than Yu et al. (2016), this also simplifies the model training using a simple cross-entropy loss. A recent work by Raffel et al. (2017) jointly learns the hard monotonic alignments and transduction while maintaining the dependency between the encoder and the decoder. Jaitly et al. (2015) proposed the Neural Transducer model, which is also trained on external alignments. They divide the input into blocks of a constant size and perform soft attention separately on each block. Lu et al. (2016) used a combination of an RNN encoder with a CRF layer to model the dependencies in the output sequence. An interesting comparison between "traditional" sequence transduction models (Bisani and Ney, 2008; Jiampojamarn et al., 2010; Novak et al., 2012) and encoder-decoder neural networks for monotone string transduction tasks was done by Schnober et al. (2016), showing that in many cases there is no clear advantage to one approach over the other.

Regarding task-specific improvements to the attention mechanism, a line of work on attention-based speech recognition (Chorowski et al., 2015; Bahdanau et al., 2016) proposed adding location awareness by using the previous attention weights when computing the next ones, and preventing the model from attending on too many or too few inputs using "sharpening" and "smoothing" techniques on the attention weight distributions. Cohn et al. (2016) offered several changes to the attention score computation to encourage well-known modeling biases found in traditional machine translation models like word fertility, position and alignment symmetry. Regarding the utilization of independent alignment models for training attention-based networks, Mi et al. (2016) showed that the distance between the attention-infused alignments and the ones learned by an independent alignment model can be added to the networks' training objective, resulting in an improved translation and alignment quality.

7 Conclusion

We presented a hard attention model for morphological inflection generation. The model employs an explicit alignment which is used to train a neural network to perform transduction by decoding with a hard attention mechanism. Our model performs better than previous neural and non-neural approaches on various morphological inflection generation datasets, while staying competitive with dedicated models even with very few training examples. It is also computationally appealing as it enables linear time decoding while staying resolution preserving, i.e. not requiring to compress the input sequence to a single fixed-sized vector. Future work may include applying our model to other nearly-monotonic align-and-transduce tasks like abstractive summarization, transliteration or machine translation.

Acknowledgments

This work was supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI), and The Israeli Science Foundation (grant number 1555/15).

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *NAACL HLT 2015*. pages 1024–1029.

R Harald Baayen, Richard Piepenbrock, and Rijn van H. 1993. The {CELEX} lexical data base on {CD-ROM} .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of the International Conference on Learning Representations (ICLR)* .

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Yoshua Bengio, et al. 2016. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 4945–4949.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Commun.* 50(5):434–451. https://doi.org/10.1016/j.specom.2008.01.002.

Victor Chahuneau, Eva Schlinger, Noah A. Smith, and Chris Dyer. 2013. Translating into morphologically rich languages with synthetic phrases. In *EMNLP*. pages 1677–1687.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems 28*, pages 577–585.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *ACL*. pages 32–42.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 876–885. http://www.aclweb.org/anthology/N16-1102.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 2016 Meeting of SIGMORPHON*.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *EMNLP*. pages 616–627.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the conference on empirical methods in natural language processing*. pages 1080–1089.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *NAACL HLT 2013*. pages 1185–1195.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th annual meeting on Association for Computational Linguistics*. pages 1–8.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *NAACL HLT 2016*.

Alexander M. Fraser, Marion Weller, Aoife Cahill, and Fabienne Cap. 2012. Modeling inflection and word-formation in smt. In *EACL*. pages 664–674.

Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. 2016. Factored neural machine translation. *arXiv preprint arXiv:1609.04621* .

A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18(5-6):602–610.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711* .

Mans Hulden, Markus Forsberg, and Malin Ahlberg. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *EACL*. pages 569–578.

Navdeep Jaitly, David Sussillo, Quoc V Le, Oriol Vinyals, Ilya Sutskever, and Samy Bengio. 2015. A neural transducer. *arXiv preprint arXiv:1511.04868* .

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2010. Integrating joint n-gram features into a discriminative training framework. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 697–700. http://www.aclweb.org/anthology/N10-1103.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* .

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural multi-source morphological reinflection. *EACL 2017* .

Katharina Kann and Hinrich Schütze. 2016a. Med: The lmu system for the sigmorphon 2016 shared task on morphological reinflection.

Katharina Kann and Hinrich Schütze. 2016b. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*.

Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20(3):331–378.

Kimmo Koskenniemi. 1983. Two-level morphology: A general computational model of word-form recognition and production. Technical report.

Liang Lu, Lingpeng Kong, Chris Dyer, Noah A Smith, and Steve Renals. 2016. Segmental recurrent neural networks for end-to-end speech recognition. *arXiv preprint arXiv:1603.00223* .

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2283–2288. https://aclweb.org/anthology/D16-1249.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 128–135. http://www.aclweb.org/anthology/P07-1017.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 1997. A rational design for a weighted finite-state transducer library. In *International Workshop on Implementing Automata*. pages 144–158.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *NAACL HLT 2015*. pages 922–931.

Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012. WFST-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*. Association for Computational Linguistics, Donostia–San Sebastin, pages 45–49. http://www.aclweb.org/anthology/W12-6208.

C. Raffel, T. Luong, P. J. Liu, R. J. Weiss, and D. Eck. 2017. Online and Linear-Time Attention by Enforcing Monotonic Alignments. *arXiv preprint arXiv:1704.00784* .

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proc. of NAACL*.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1703–1714. http://aclweb.org/anthology/C16-1160.

Katsuhito Sudoh, Shinsuke Mori, and Masaaki Nagata. 2013. Noise-aware character alignment for bootstrapping statistical machine transliteration from bilingual corpora. In *EMNLP 2013*. pages 204–209.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *ACL*. pages 514–522.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *ACL*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1307–1316. https://aclweb.org/anthology/D16-1138.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Supplementary Material

Training Details, Implementation and Hyper Parameters

To train our models, we used the train portion of the datasets as-is and evaluated on the test portion the model which performed best on the development portion of the dataset, without conducting any specific pre-processing steps on the data. We train the models for a maximum of 100 epochs over the training set. To avoid long training time, we trained the model for 20 epochs for datasets larger than 50k examples, and for 5 epochs for datasets larger than 200k examples. The models were implemented using the python bindings of the dynet toolkit.[4]

We trained the network by optimizing the expected output sequence likelihood using cross-entropy loss as mentioned in equation 5. For optimization we used ADADELTA (Zeiler, 2012) without regularization. We updated the weights after every example (i.e. mini-batches of size 1). We used the dynet toolkit implementation of an LSTM network with two layers for all models, each having 100 entries in both the encoder and decoder. The character embeddings were also vectors with 100 entries for the CELEX experiments, and with 300 entries for the SIGMORPHON and Wiktionary experiments.

The morpho-syntactic attribute embeddings were vectors of 20 entries in all experiments. We did not use beam search while decoding for both the hard and soft attention models as it is significantly slower and did not show clear improvement in previous experiments we conducted. For the character level alignment process we use the implementation provided by the organizers of the SIGMORPHON2016 shared task.[5]

LSTM Equations

We used the LSTM variant implemented in the dynet toolkit, which corresponds to the following equations:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{f}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\widetilde{\mathbf{c}} &= \tanh(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{c}_{t-1} \circ \mathbf{f}_t + \widetilde{\mathbf{c}} \circ \mathbf{i}_t \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{ox}\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \tanh(\mathbf{c}_t) \circ \mathbf{o}_t
\end{aligned}
\tag{6}
$$

[4] https://github.com/clab/dynet

[5] https://github.com/ryancotterell/sigmorphon2016

From Characters to Words to in Between: Do We Capture Morphology?

Clara Vania and **Adam Lopez**
Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh
c.vania@ed.ac.uk, alopez@inf.ed.ac.uk

Abstract

Words can be represented by composing the representations of subword units such as word segments, characters, and/or character n-grams. While such representations are effective and may capture the morphological regularities of words, they have not been systematically compared, and it is not understood how they interact with different morphological typologies. On a language modeling task, we present experiments that systematically vary (1) the basic unit of representation, (2) the composition of these representations, and (3) the morphological typology of the language modeled. Our results extend previous findings that character representations are effective across typologies, and we find that a previously unstudied combination of character trigram representations composed with bi-LSTMs outperforms most others. But we also find room for improvement: none of the character-level models match the predictive accuracy of a model with access to true morphological analyses, even when learned from an order of magnitude more data.

1 Introduction

Continuous representations of words learned by neural networks are central to many NLP tasks (Cho et al., 2014; Chen and Manning, 2014; Dyer et al., 2015). However, directly mapping a finite set of word types to a continuous representation has well-known limitations. First, it makes a closed vocabulary assumption, enabling only generic out-of-vocabulary handling. Second, it cannot exploit systematic functional relationships in learning. For example, *cat* and *cats* stand in the same relationship as *dog* and *dogs*. While this relationship might be discovered for these specific frequent words, it does not help us learn that the same relationship also holds for the much rarer words *sloth* and *sloths*.

These functional relationships reflect the fact that words are composed from smaller units of meaning, or morphemes. For instance, *cats* consists of two morphemes, *cat* and *-s*, with the latter shared by the words *dogs* and *tarsiers*. Modeling this effect is crucial for languages with rich morphology, where vocabulary sizes are larger, many more words are rare, and many more such functional relationships exist. Hence, some models produce word representations as a function of subword units obtained from morphological segmentation or analysis (Luong et al., 2013; Botha and Blunsom, 2014; Cotterell and Schütze, 2015). A downside of these models is that they depend on morphological segmenters or analyzers.

Morphemes typically have similar orthographic representations across words. For example, the morpheme *-s* is realized as *-es* in *finches*. Since this variation is limited, the general relationship between morphology and orthography can be exploited by composing the representations of characters (Ling et al., 2015; Kim et al., 2016), character n-grams (Sperr et al., 2013; Wieting et al., 2016; Bojanowski et al., 2016; Botha and Blunsom, 2014), bytes (Plank et al., 2016; Gillick et al., 2016), or combinations thereof (Santos and Zadrozny, 2014; Qiu et al., 2014). These models are compact, can represent rare and unknown words, and do not require morphological analyzers. They raise a provocative question: Does NLP benefit from models of morphology, or can they be replaced entirely by models of characters?

The relative merits of word, subword. and character-level models are not fully understood because each new model has been compared on

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2016–2027
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1184

different tasks and datasets, and often compared against word-level models. A number of questions remain open:

1. How do representations based on morphemes compare with those based on characters?
2. What is the best way to compose subword representations?
3. Do character-level models capture morphology in terms of predictive utility?
4. How do different representations interact with languages of different morphological typologies?

The last question is raised by Bender (2013): languages are typologically diverse, and the behavior of a model on one language may not generalize to others. Character-level models implicitly assume concatenative morphology, but many widely-spoken languages feature non-concatenative morphology, and it is unclear how such models will behave on these languages.

To answer these questions, we performed a systematic comparison across different models for the simple and ubiquitous task of language modeling. We present experiments that vary (1) the type of subword unit; (2) the composition function; and (3) morphological typology. To understand the extent to which character-level models capture true morphological regularities, we present oracle experiments using human morphological annotations instead of automatic morphological segments. Our results show that:

1. For most languages, character-level representations outperform the standard word representations. Most interestingly, a previously unstudied combination of character trigrams composed with bi-LSTMs performs best on the majority of languages.
2. Bi-LSTMs and CNNs are more effective composition functions than addition.
3. Character-level models learn functional relationships between orthographically similar words, but don't (yet) match the predictive accuracy of models with access to true morphological analyses.
4. Character-level models are effective across a range of morphological typologies, but orthography influences their effectiveness.

word	`tries`
morphemes	`try+s`
morphs	`tri+es`
morph. analysis	`try+VB+3rd+SG+Pres`

Table 1: The morphemes, morphs, and morphological analysis of *tries*.

2 Morphological Typology

A **morpheme** is the smallest unit of meaning in a word. Some morphemes express core meaning (**roots**), while others express one or more dependent **features** of the core meaning, such as person, gender, or aspect. A **morphological analysis** identifies the lemma and features of a word. A **morph** is the surface realization of a morpheme (Morley, 2000), which may vary from word to word. These distinctions are shown in Table 1.

Morphological typology classifies languages based on the processes by which morphemes are composed to form words. While most languages will exhibit a variety of such processes, for any given language, some processes are much more frequent than others, and we will broadly identify our experimental languages with these processes.

When morphemes are combined sequentially, the morphology is **concatenative**. However, morphemes can also be composed by **non-concatenative** processes. We consider four broad categories of both concatenative and non-concatenative processes in our experiments.

Fusional languages realize multiple features in a single concatenated morpheme. For example, English verbs can express number, person, and tense in a single morpheme:

wanted (English)
want + *ed*
want + `VB+1st+SG+Past`

Agglutinative languages assign one feature per morpheme. Morphemes are concatenated to form a word and the morpheme boundaries are clear. For example (Haspelmath, 2010):

okursam (Turkish)
oku+r+sa+m
"read"+`AOR+COND+1SG`

Root and Pattern Morphology forms words by inserting consonants and vowels of dependent morphemes into a consonantal root based on a given pattern. For example, the Arabic root *ktb* ("write") produces (Roark and Sproat, 2007):

katab "wrote" (Arabic)

takaatab "wrote to each other" (Arabic)

Reduplication is a process where a word form is produced by repeating part or all of the root to express new features. For example:

anak "child" (Indonesian)
anak-anak "children" (Indonesian)
buah "fruit" (Indonesian)
buah-buahan "various fruits" (Indonesian)

3 Representation Models

We compare ten different models, varying subword units and composition functions that have commonly been used in recent work, but evaluated on various different tasks (Table 2). Given word w, we compute its representation $\mathbf{w}$ as:

$$\mathbf{w} = f(\mathbf{W}_s, \sigma(w)) \quad (1)$$

where σ is a deterministic function that returns a sequence of subword units; $\mathbf{W}_s$ is a parameter matrix of representations for the vocabulary of subword units; and f is a composition function which takes $\sigma(w)$ and $\mathbf{W}_s$ as input and returns $\mathbf{w}$. All of the representations that we consider take this form, varying only in f and σ.

3.1 Subword Units

We consider four variants of σ in Equation 1, each returning a different type of subword unit: character, character trigram, or one of two types of morph. Morphs are obtained from Morfessor (Smit et al., 2014) or a word segmentation based on Byte Pair Encoding (BPE; Gage (1994)), which has been shown to be effective for handling rare words in neural machine translation (Sennrich et al., 2016). BPE works by iteratively replacing frequent pairs of characters with a single unused character. For Morfessor, we use default parameters while for BPE we set the number of merge operations to 10,000.[1] When we segment into character trigrams, we consider all trigrams in the word, including those covering notional beginning and end of word characters, as in Sperr et al. (2013). Example output of σ is shown in Table 3.

3.2 Composition Functions

We use three variants of f in Eq. 1. The first constructs the representation $\mathbf{w}$ of word w by adding the representations of its subwords $s_1, \ldots, s_n = \sigma(w)$, where the representation of s_i is vector $\mathbf{s}_i$.

$$\mathbf{w} = \sum_{i=1}^{n} \mathbf{s}_i \quad (2)$$

The only subword unit that we don't compose by addition is characters, since this will produce the same representation for many different words.

Our second composition function is a bidirectional long-short-term memory (**bi-LSTM**), which we adapt based on its use in the character-level model of Ling et al. (2015) and its widespread use in NLP generally. Given $\mathbf{s}_i$ and the previous LSTM hidden state $\mathbf{h}_{i-1}$, an LSTM (Hochreiter and Schmidhuber, 1997) computes the following outputs for the subword at position i:

$$\mathbf{h}_i = LSTM(\mathbf{s}_i, \mathbf{h}_{i-1}) \quad (3)$$

$$\hat{s}_{i+1} = g(\mathbf{V}^T \cdot \mathbf{h}_i) \quad (4)$$

where $\hat{s}_{i+1}$ is the predicted target subword, g is the softmax function and $\mathbf{V}$ is a weight matrix.

A bi-LSTM (Graves et al., 2005) combines the final state of an LSTM over the input sequence with one over the reversed input sequence. Given the hidden state produced from the final input of the forward LSTM, $\mathbf{h}_n^{fw}$ and the hidden state produced from the final input of the backward LSTM $\mathbf{h}_0^{bw}$, we compute the word representation as:

$$\mathbf{w}_t = \mathbf{W}_f \cdot \mathbf{h}_n^{fw} + \mathbf{W}_b \cdot \mathbf{h}_0^{bw} + \mathbf{b} \quad (5)$$

where $\mathbf{W}_f$, $\mathbf{W}_b$, and $\mathbf{b}$ are parameter matrices and $\mathbf{h}_n^{fw}$ and $\mathbf{h}_0^{bw}$ are forward and backward LSTM states, respectively.

The third composition function is a convolutional neural network (**CNN**) with highway layers, as in Kim et al. (2016). Let $c_1, \ldots, c_k$ be the sequence of characters of word w. The character embedding matrix is $\mathbf{C} \in \mathbb{R}^{d \times k}$, where the i-th column corresponds to the embeddings of c_i. We first apply a narrow convolution between $\mathbf{C}$ and a filter $\mathbf{F} \in \mathbb{R}^{d \times n}$ of width n to obtain a feature map $\mathbf{f} \in \mathbf{R}^{k-n+1}$. In particular, the computation of the j-th element of $\mathbf{f}$ is defined as

$$\mathbf{f}[j] = tanh(\langle \mathbf{C}[*, j : j+n-1], \mathbf{F} \rangle + b) \quad (6)$$

where $\langle A, B \rangle = \mathtt{Tr}(\mathbf{A}\mathbf{B}^T)$ is the Frobenius inner product and b is a bias. The CNN model applies filters of varying width, representing features

[1] BPE takes a single parameter: the number of merge operations. We tried different parameter values (1k, 10k, 100k) and manually examined the resulting segmentation on the English dataset. Qualitatively, 10k gave the most plausible segmentation and we used this setting across all languages.

Models	Subword Unit(s)	Composition Function
Sperr et al. (2013)	words, character n-grams	addition
Luong et al. (2013)	morphs (Morfessor)	recursive NN
Botha and Blunsom (2014)	words, morphs (Morfessor)	addition
Qiu et al. (2014)	words, morphs (Morfessor)	addition
Santos and Zadrozny (2014)	words, characters	CNN
Cotterell and Schütze (2015)	words, morphological analyses	addition
Sennrich et al. (2016)	morphs (BPE)	none
Kim et al. (2016)	characters	CNN
Ling et al. (2015)	characters	bi-LSTM
Wieting et al. (2016)	character n-grams	addition
Bojanowski et al. (2016)	character n-grams	addition
Vylomova et al. (2016)	characters, morphs (Morfessor)	bi-LSTM, CNN
Miyamoto and Cho (2016)	words, characters	bi-LSTM
Rei et al. (2016)	words, characters	bi-LSTM
Lee et al. (2016)	characters	CNN
Kann and Schütze (2016)	characters, morphological analyses	none
Heigold et al. (2017)	words, characters	bi-LSTM, CNN

Table 2: Summary of previous work on representing words through compositions of subword units.

Unit	Output of $\sigma(\textit{wants})$
Morfessor	ˆwant, s$
BPE	ˆw, ants$
char-trigram	ˆwa, wan, ant, nts ts$
character	ˆ, w, a, n, t, s, $
analysis	want+VB, +3rd, +SG, +Pres

Table 3: Input representations for *wants*.

of character n-grams. We then calculate the max-over-time of each feature map.

$$y_j = \max_j \mathbf{f}[j] \tag{7}$$

and concatenate them to derive the word representation $\mathbf{w}_t = [y_1, \ldots, y_m]$, where m is the number of filters applied. Highway layers allow some dimensions of $\mathbf{w}_t$ to be carried or transformed. Since it can learn character n-grams directly, we only use the CNN with character input.

3.3 Language Model

We use language models (LM) because they are simple and fundamental to many NLP applications. Given a sequence of text $s = w_1, \ldots, w_T$, our LM computes the probability of s as:

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(y_t | w_1, \ldots, w_{t-1}) \tag{8}$$

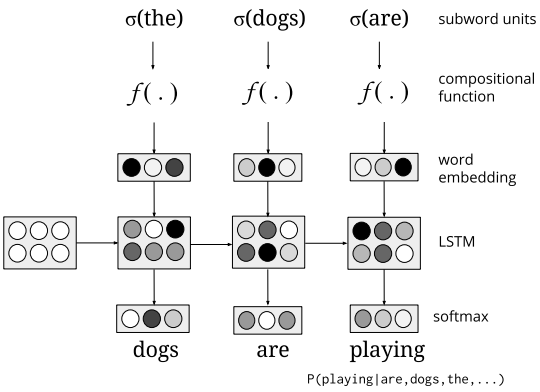


Figure 1: Our LSTM-LM architecture.

where $y_t = w_t$ if w_t is in the output vocabulary and $y_t = \texttt{UNK}$ otherwise.

Our language model is an LSTM variant of recurrent neural network language (RNN) LM (Mikolov et al., 2010). At time step t, it receives input w_t and predicts y_{t+1}. Using Eq. 1, it first computes representation $\mathbf{w}_t$ of w_t. Given this representation and previous state $\mathbf{h}_{t-1}$, it produces a new state $\mathbf{h}_t$ and predicts y_{t+1}:

$$\mathbf{h}_t = LSTM(\mathbf{w}_t, \mathbf{h}_{t-1}) \tag{9}$$

$$\hat{y}_{t+1} = g(\mathbf{V}^T \cdot \mathbf{h}_t) \tag{10}$$

where g is a softmax function over the vocabulary yielding the probability in Equation 8. Note that this design means that we can *predict* only words

Typology	Languages	#tokens	#types
Fusional	Czech	1.2M	125.4K
	English	1.2M	81.1K
	Russian	0.8M	103.5K
Agglutinative	Finnish	1.2M	188.4K
	Japanese	1.2M	59.2K
	Turkish	0.6M	126.2K
Root&Pattern	Arabic	1.4M	137.5K
	Hebrew	1.1M	104.9K
Reduplication	Indonesian	1.2M	76.5K
	Malaysian	1.2M	77.7K

Table 4: Statistics of our datasets.

from a finite output vocabulary, so our models differ only in their representation of context words. This design makes it possible to compare language models using perplexity, since they have the same event space, though open vocabulary word prediction is an interesting direction for future work.

The complete architecture of our system is shown in Figure 1, showing segmentation function σ and composition function f from Equation 1.

4 Experiments

We perform experiments on ten languages (Table 4). We use datasets from Ling et al. (2015) for English and Turkish. For Czech and Russian we use Universal Dependencies (UD) v1.3 (Nivre et al., 2015). For other languages, we use preprocessed Wikipedia data (Al-Rfou et al., 2013).[2] For each dataset, we use approximately 1.2M tokens to train, and approximately 150K tokens each for development and testing. Preprocessing involves lowercasing (except for character models) and removing hyperlinks.

To ensure that we compared models and not implementations, we reimplemented all models in a single framework using Tensorflow (Abadi et al., 2015).[3] We use a common setup for all experiments based on that of Ling et al. (2015), Kim et al. (2016), and Miyamoto and Cho (2016). In preliminary experiments, we confirmed that our models produced similar patterns of perplexities for the reimplemented word and character LSTM models of Ling et al. (2015). Even following detailed discussion with Ling (p.c.), we were unable to reproduce their perplexities exactly—our English reimplementation gives lower perplexities; our Turkish higher—but we do reproduce their general result that character bi-LSTMs outperform word models. We suspect that different preprocessing and the stochastic learning explains differences in perplexities. Our final model with bi-LSTMs composition follows Miyamoto and Cho (2016) as it gives us the same perplexity results for our preliminary experiments on the Penn Treebank dataset (Marcus et al., 1993), preprocessed by Mikolov et al. (2010).

4.1 Training and Evaluation

Our LSTM-LM uses two hidden layers with 200 hidden units and representation vectors for words, characters, and morphs all have dimension 200. All parameters are initialized uniformly at random from -0.1 to 0.1, trained by stochastic gradient descent with mini-batch size of 32, time steps of 20, for 50 epochs. To avoid overfitting, we apply dropout with probability 0.5 on the input-to-hidden layer and all of the LSTM cells (including those in the bi-LSTM, if used). For all models which do not use bi-LSTM composition, we start with a learning rate of 1.0 and decrease it by half if the validation perplexity does not decrease by 0.1 after 3 epochs. For models with bi-LSTMs composition, we use a constant learning rate of 0.2 and stop training when validation perplexity does not improve after 3 epochs. For the character CNN model, we use the same settings as the *small model* of Kim et al. (2016).

To make our results comparable to Ling et al. (2015), for each language we limit the output vocabulary to the most frequent 5,000 training words plus an unknown word token. To learn to predict unknown words, we follow Ling et al. (2015): in training, words that occur only once are stochastically replaced with the unknown token with probability 0.5. To evaluate the models, we compute perplexity on the test data.

5 Results and Analysis

Table 5 presents our main results. In six of ten languages, character-trigram representations composed with bi-LSTMs achieve the lowest perplexities. As far as we know, this particular model has not been tested before, though it is similar

[2]The Arabic and Hebrew dataset are unvocalized. Japanese mixes Kanji, Katakana, Hiragana, and Latin characters (for foreign words). Hence, a Japanese character can correspond to a character, syllable, or word. The preprocessed dataset is already word-segmented.

[3]Our implementation of these models can be found at https://github.com/claravania/subword-lstm-lm

Language	word	character		char trigrams		BPE		Morfessor		%imp
		bi-lstm	CNN	add	bi-lstm	add	bi-lstm	add	bi-lstm	
Czech	41.46	34.25	36.60	42.73	**33.59**	49.96	33.74	47.74	36.87	18.98
English	46.40	43.53	44.67	45.41	**42.97**	47.51	43.30	49.72	49.72	7.39
Russian	34.93	28.44	29.47	35.15	**27.72**	40.10	28.52	39.60	31.31	20.64
Finnish	24.21	20.05	20.29	24.89	**18.62**	26.77	19.08	27.79	22.45	23.09
Japanese	98.14	98.14	**91.63**	101.99	101.09	126.53	96.80	111.97	99.23	6.63
Turkish	66.97	54.46	55.07	**50.07**	54.23	59.49	57.32	62.20	62.70	25.24
Arabic	48.20	42.02	43.17	50.85	**39.87**	50.85	42.79	52.88	45.46	17.28
Hebrew	38.23	31.63	33.19	39.67	**30.40**	44.15	32.91	44.94	34.28	20.48
Indonesian	46.07	45.47	46.60	58.51	45.96	59.17	**43.37**	59.33	44.86	5.86
Malay	54.67	53.01	**50.56**	68.51	50.74	68.99	51.21	68.20	52.50	7.52

Table 5: Language model perplexities on test. The best model for each language is highlighted in **bold** and the improvement of this model over the word-level model is shown in the final column.

to (but more general than) the model of Sperr et al. (2013). We can see that the performance of character, character trigrams, and BPE are very competitive. Composition by bi-LSTMs or CNN is more effective than addition, except for Turkish. We also observe that BPE always outperforms Morfessor, even for the agglutinative languages. We now turn to a more detailed analysis by morphological typology.

Fusional languages. For these languages, character trigrams composed with bi-LSTMs outperformed all other models, particularly for Czech and Russian (up to 20%), which is unsurprising since both are morphologically richer than English.

Agglutinative languages. We observe different results for each language. For Finnish, character trigrams composed with bi-LSTMs achieves the best perplexity. Surprisingly, for Turkish character trigrams composed via addition is best and addition also performs quite well for other representations, potentially useful since the addition function is simpler and faster than bi-LSTMs. We suspect that this is due to the fact that Turkish morphemes are reasonably short, hence well-approximated by character trigrams. For Japanese, we improvements from character models are more modest than in other languages.

Root and Pattern. For these languages, character trigrams composed with bi-LSTMs also achieve the best perplexity. We had wondered whether CNNs would be more effective for root-and-pattern morphology, but since these data are unvocalized, it is more likely that non-concatenative effects are minimized, though we do still find morphological variants with consonantal inflections that behave more like concatenation. For example, *maktab* (root:*ktb*) is written as *mktb*. We suspect this makes character trigrams quite effective since they match the tri-consonantal root patterns among words which share the same root.

Reduplication. For Indonesian, BPE morphs composed with bi-LSTMs model obtain the best perplexity. For Malay, the character CNN outperforms other models. However, these improvements are small compared to other languages. This likely reflects that Indonesian and Malay are only moderately inflected, where inflection involves both concatenative and non-concatenative processes.

5.1 Effects of Morphological Analysis

In the experiments above, we used unsupervised morphological *segmentation* as a proxy for morphological *analysis* (Table 3). However, as discussed in Section 2, this is quite approximate, so it is natural to wonder what would happen if we had the true morphological analysis. If character-level models are powerful enough to capture the effects of morphology, then they should have the predictive accuracy of a model with access to this analysis. To find out, we conducted an oracle experiment using the human-annotated morphological analyses provided in the UD datasets for Czech and Russian, the only languages in our set for which these analyses were available. In these experiments we treat the lemma and each morphological feature as a subword unit.

The results (Table 6) show that bi-LSTM composition of these representations outperforms all

Languages	Addition	bi-LSTM
Czech	51.8	**30.07**
Russian	41.82	**26.44**

Table 6: Perplexity results using hand-annotated morphological analyses (cf. Table 5).

other models for both languages. These results demonstrate that neither character representations nor unsupervised segmentation is a perfect replacement for manual morphological analysis, at least in terms of predictive accuracy. In light of character-level results, they imply that current unsupervised morphological analyzers are poor substitutes for real morphological analysis.

However, we can obtain much more unannotated than annotated data, and we might guess that the character-level models would outperform those based on morphological analyses if trained on larger data. To test this, we ran experiments that varied the training data size on three representation models: word, character-trigram bi-LSTM, and character CNN. Since we want to see how much training data is needed to reach perplexity obtained using annotated data, we use the same output vocabulary derived from the original training. While this makes it possible to compare perplexities across models, it is unfavorable to the models trained on larger data, which may focus on other words. This is a limitation of our experimental setup, but does allow us to draw some tentative conclusions. As shown in Table 7, a character-level model trained on an order of magnitude more data still does not match the predictive accuracy of a model with access to morphological analysis.

5.2 Automatic Morphological Analysis

The oracle experiments show promising results if we have annotated data. But these annotations are expensive, so we also investigated the use of automatic morphological analysis. We obtained analyses for Arabic with the MADAMIRA (Pasha et al., 2014).[4] As in the experiment using annotations, we treated each morphological feature as a subword unit. The resulting perplexities of **71.94** and **42.85** for addition and bi-LSTMs, respectively, are worse than those obtained with character trigrams (**39.87**), though it approaches the best perplexities.

[4]We only experimented with Arabic since MADAMIRA disambiguates words in contexts; most other analyzers we found did not do this, and would require additional work to add disambiguation.

#tokens	word	char trigram bi-LSTM	char CNN
1M	39.69	32.34	35.15
2M	37.59	36.44	35.58
3M	36.71	35.60	35.75
4M	35.89	32.68	35.93
5M	35.20	34.80	37.02
10M	35.60	35.82	39.09

Table 7: Perplexity results on the Czech development data, varying training data size. Perplexity using ~1M tokens annotated data is **28.83**.

5.3 Targeted Perplexity Results

A difficulty in interpreting the results of Table 5 with respect to specific morphological processes is that perplexity is measured for all words. But these processes do not apply to all words, so it may be that the effects of specific morphological processes are washed out. To get a clearer picture, we measured perplexity for only specific subsets of words in our test data: specifically, given target word w_i, we measure perplexity of word w_{i+1}. In other words, we analyze the perplexities *when the inflected words of interest are in the most recent history*, exploiting the recency bias of our LSTM-LM. This is the perplexity most likely to be strongly affected by different representations, since we do not vary representations of the predicted word itself.

We look at several cases: nouns and verbs in Czech and Russian, where word classes can be identified from annotations, and reduplication in Indonesian, which we can identify mostly automatically. For each analysis, we also distinguish between *frequent* cases, where the inflected word occurs more than ten times in the training data, and *rare* cases, where it occurs fewer than ten times. We compare only bi-LSTM models.

For Czech and Russian, we again use the UD annotation to identify words of interest. The results (Table 8), show that manual morphological analysis uniformly outperforms other subword models, with an especially strong effect for Czech nouns, suggesting that other models do not capture useful predictive properties of a morphological analysis. We do however note that character trigrams achieve low perplexities in most cases, similar to overall results (Table 5). We also observe that the subword models are more effective for rare words.

Inflection	Model	all	frequent	rare
Czech nouns	word	61.21	56.84	72.96
	characters	51.01	47.94	59.01
	char-trigrams	50.34	48.05	56.13
	BPE	53.38	49.96	62.81
	morph. analysis	**40.86**	**40.08**	**42.64**
Czech verbs	word	81.37	74.29	99.40
	characters	70.75	68.07	77.11
	char-trigrams	65.77	63.71	70.58
	BPE	74.18	72.45	78.25
	morph. analysis	**59.48**	**58.56**	**61.78**
Russian nouns	word	45.11	41.88	48.26
	characters	37.90	37.52	38.25
	char-trigrams	36.32	34.19	38.40
	BPE	43.57	43.67	43.47
	morph. analysis	**31.38**	**31.30**	**31.50**
Russian verbs	word	56.45	47.65	69.46
	characters	45.00	40.86	50.60
	char-trigrams	42.55	39.05	47.17
	BPE	54.58	47.81	64.12
	morph. analysis	**41.31**	**39.8**	**43.18**

Table 8: Average perplexities of words that occur after nouns and verbs. Frequent words occur more than ten times in the training data; rare words occur fewer times than this. The best perplexity is in **bold** while the second best is underlined.

For Indonesian, we exploit the fact that the hyphen symbol '-' typically separates the first and second occurrence of a reduplicated morpheme, as in the examples of Section 2. We use the presence of word tokens containing hyphens to estimate the percentage of those exhibiting reduplication. As shown in Table 9, the numbers are quite low.

Table 10 shows results for reduplication. In contrast with the overall results, the BPE bi-LSTM model has the worst perplexities, while character bi-LSTM has the best, suggesting that these models are more effective for reduplication.

Looking more closely at BPE segmentation of reduplicated words, we found that only 6 of 252 reduplicated words have a correct word segmentation, with the reduplicated morpheme often combining differently with the notional start-of-word or hyphen character. One the other hand BPE correctly learns 8 out of 9 Indonesian prefixes and 4 out of 7 Indonesian suffixes.[5] This analysis supports our intuition that the improvement from BPE might come from its modeling of concatenative morphology.

5.4 Qualitative Analysis

Table 11 presents nearest neighbors under cosine similarity for in-vocabulary, rare, and out-of-vocabulary (OOV) words.[6] For frequent words, standard word embeddings are clearly superior for lexical meaning. Character and morph representations tend to find words that are orthographically similar, suggesting that they are better at modeling dependent than root morphemes. The same pattern holds for rare and OOV words. We suspect that the subword models outperform words on language modeling because they exploit affixes to signal word class. We also noticed similar patterns in Japanese.

We analyze reduplication by querying reduplicated words to find their nearest neighbors using the BPE bi-LSTM model. If the model were sensitive to reduplication, we would expect to see morphological variants of the query word among its nearest neighbors. However, from Table 12, this is not so. With the partially reduplicated query *berlembah-lembah*, we do not find the lemma *lembah*.

Language	type-level (%)	token-level (%)
Indonesian	1.10	2.60
Malay	1.29	2.89

Table 9: Percentage of full reduplication on the type and token level.

Model	all	frequent	rare
word	101.71	91.71	156.98
characters	**99.21**	**91.35**	**137.42**
BPE	117.2	108.86	156.81

Table 10: Average perplexities of words that occur after reduplicated words in the test set.

6 Conclusion

We presented a systematic comparison of word representation models with different levels of morphological awareness, across languages with different morphological typologies. Our results confirm previous findings that character-level models are effective for many languages, but these models do not match the predictive accuracy of model with explicit knowledge of morphology, even after we increase the training data size by ten times. Moreover, our qualitative analysis suggests that they learn orthographic similarity of affixes, and lose the meaning of root morphemes.

Although morphological analyses are available

[5] We use Indonesian affixes listed in Larasati et al. (2011)

[6] https://radimrehurek.com/gensim/

Model	Frequent Words			Rare Words		OOV words	
	man	*including*	*relatively*	*unconditional*	*hydroplane*	*uploading*	*foodism*
word	person	like	extremely	nazi	molybdenum	-	-
	anyone	featuring	making	fairly	your	-	-
	children	include	very	joints	imperial	-	-
	men	includes	quite	supreme	intervene	-	-
BPE LSTM	ii	called	newly	unintentional	emphasize	upbeat	vigilantism
	hill	involve	never	ungenerous	heartbeat	uprising	pyrethrum
	text	like	essentially	unanimous	hybridized	handling	pausanias
	netherlands	creating	least	unpalatable	unplatable	hand-colored	footway
char-trigrams LSTM	mak	include	resolutely	unconstitutional	selenocysteine	drifted	tuaregs
	vill	includes	regeneratively	constitutional	guerrillas	affected	quft
	cow	undermining	reproductively	unimolecular	scrofula	conflicted	subjectivism
	maga	under	commonly	medicinal	seleucia	convicted	tune-up
char-LSTM	mayr	inclusion	relates	undamaged	hydrolyzed	musagte	formulas
	many	insularity	replicate	unmyelinated	hydraulics	mutualism	formally
	mary	includes	relativity	unconditionally	hysterotomy	mutualists	fecal
	may	include	gravestones	uncoordinated	hydraulic	meursault	foreland
char-CNN	mtn	include	legislatively	unconventional	hydroxyproline	unloading	fordism
	mann	includes	lovely	unintentional	hydrate	loading	dadaism
	jan	excluding	creatively	unconstitutional	hydrangea	upgrading	popism
	nun	included	negatively	untraditional	hyena	upholding	endemism

Table 11: Nearest neighbours of semantically and syntactically similar words.

Query	Top nearest neighbours
kota-kota (*cities*)	wilayah-wilayah (*areas*), pulau-pulau (*islands*), negara-negara (*countries*), bahasa-bahasa (*languages*), koloni-koloni (*colonies*)
berlembah-lembah (*have many valleys*)	berargumentasi (*argue*), bercakap-cakap (*converse*), berkemauan (*will*), berimplikasi (*imply*), berketebalan (*have a thickness*)

Table 12: Nearest neighbours of Indonesian reduplicated words in the BPE bi-LSTM model.

in limited quantities, our results suggest that there might be utility in semi-supervised learning from partially annotated data. Across languages with different typologies, our experiments show that the subword unit models are most effective on agglutinative languages. However, these results do not generalize to all languages, since factors such as morphology and orthography affect the utility of these representations. We plan to explore these effects in future work.

Acknowledgments

Clara Vania is supported by the Indonesian Endowment Fund for Education (LPDP), the Centre for Doctoral Training in Data Science, funded by the UK EPSRC (grant EP/L016427/1), and the University of Edinburgh. We thank Sameer Bansal, Toms Bergmanis, Marco Damonte, Federico Fancellu, Sorcha Gilroy, Sharon Goldwater, Frank Keller, Mirella Lapata, Felicia Liu, Jonathan Mallinson, Joana Ribeiro, Naomi Saphra, Ida Szubert, and the anonymous reviewers for helpful discussion of this work and comments on previous drafts of the paper.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. http://tensorflow.org/.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, pages 183–192. http://www.aclweb.org/anthology/W13-3520.

Emily M. Bender. 2013. *Linguistic Fundamentals for Natural Language Processing: 100 Essentials from Morphology and Syntax*. Morgan & Claypool Publishers.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR* abs/1607.04606. http://arxiv.org/abs/1607.04606.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modeling. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*. Beijing, China. http://jmlr.org/proceedings/papers/v32/botha14.pdf.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 740–750. http://www.aclweb.org/anthology/D14-1082.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1287–1292. http://www.aclweb.org/anthology/N15-1140.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 334–343. http://www.aclweb.org/anthology/P15-1033.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.* 12(2):23–38. http://dl.acm.org/citation.cfm?id=177910.177914.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1296–1306. http://www.aclweb.org/anthology/N16-1155.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*. Springer-Verlag, Berlin, Heidelberg, ICANN'05, pages 799–804. http://dl.acm.org/citation.cfm?id=1986079.1986220.

Martin Haspelmath. 2010. *Understanding Morphology*. Understanding Language Series. Arnold, London, second edition.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 505–513. http://aclweb.org/anthology/E17-1048.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Katharina Kann and Hinrich Schütze. 2016. *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Association for Computational Linguistics, chapter MED: The LMU System for the SIGMORPHON 2016 Shared Task on Morphological Reinflection, pages 62–70. https://doi.org/10.18653/v1/W16-2010.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. 2016. Character-aware neural language models. In *Proceedings of the 2016 Conference on Artificial Intelligence (AAAI)*.

Septina Dian Larasati, Vladislav Kuboň, and Daniel Zeman. 2011. *Indonesian Morphology Tool (MorphInd): Towards an Indonesian Corpus*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 119–129. https://doi.org/10.1007/978-3-642-23138-4_8.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *CoRR* abs/1610.03017. http://arxiv.org/abs/1610.03017.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1520–1530. http://aclweb.org/anthology/D15-1176.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for

Computational Linguistics, Sofia, Bulgaria, pages 104–113. http://www.aclweb.org/anthology/W13-3512.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.* 19(2):313–330. http://dl.acm.org/citation.cfm?id=972470.972475.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*. International Speech Communication Association, volume 2010, pages 1045–1048. http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1992–1997. https://aclweb.org/anthology/D16-1209.

G. David Morley. 2000. *Syntax in Functional Grammar: An Introduction to Lexicogrammar in Systemic Linguistics*. Continuum.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Ctlina Mrnduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015. Universal dependencies 1.2 LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague. http://hdl.handle.net/11234/1-1548.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland, pages 1094–1101. ACL Anthology Identifier: L14-1479.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 412–418. http://anthology.aclweb.org/P16-2067.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 141–150. http://www.aclweb.org/anthology/C14-1015.

Marek Rei, Gamal Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 309–318. http://aclweb.org/anthology/C16-1030.

Brian Roark and Richard Sproat. 2007. *Computational Approach to Morphology and Syntax*. Oxford University Press.

Cicero Dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*. PMLR, Bejing, China, volume 32 of *Proceedings of Machine Learning Research*, pages 1818–1826. http://proceedings.mlr.press/v32/santos14.html.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725. http://www.aclweb.org/anthology/P16-1162.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, pages 21–24. http://www.aclweb.org/anthology/E14-2006.

Henning Sperr, Jan Niehues, and Alex Waibel. 2013. Letter n-gram-based input encoding for continuous space language models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*. Association for Computational Linguistics, Sofia, Bulgaria, pages 30–39. http://www.aclweb.org/anthology/W13-3204.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2016. Word representation models for morphologically rich languages in neural machine translation. *CoRR* abs/1606.04217. http://arxiv.org/abs/1606.04217.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1504–1515. https://aclweb.org/anthology/D16-1157.

Riemannian Optimization for Skip-Gram Negative Sampling

Alexander Fonarev[1,2,4,*], **Oleksii Hrinchuk**[1,2,3,*],
Gleb Gusev[2,3], **Pavel Serdyukov**[2], and **Ivan Oseledets**[1,5]

[1]Skolkovo Institute of Science and Technology, Moscow, Russia
[2]Yandex LLC, Moscow, Russia
[3]Moscow Institute of Physics and Technology, Moscow, Russia
[4]SBDA Group, Dublin, Ireland
[5]Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow, Russia

Abstract

Skip-Gram Negative Sampling (SGNS) word embedding model, well known by its implementation in "word2vec" software, is usually optimized by stochastic gradient descent. However, the optimization of SGNS objective can be viewed as a problem of searching for a good matrix with the low-rank constraint. The most standard way to solve this type of problems is to apply Riemannian optimization framework to optimize the SGNS objective over the manifold of required low-rank matrices. In this paper, we propose an algorithm that optimizes SGNS objective using Riemannian optimization and demonstrates its superiority over popular competitors, such as the original method to train SGNS and SVD over SPPMI matrix.

1 Introduction

In this paper, we consider the problem of embedding words into a low-dimensional space in order to measure the semantic similarity between them. As an example, how to find whether the word "table" is semantically more similar to the word "stool" than to the word "sky"? That is achieved by constructing a low-dimensional vector representation for each word and measuring similarity between the words as the similarity between the corresponding vectors.

One of the most popular word embedding models (Mikolov et al., 2013) is a discriminative neural network that optimizes Skip-Gram Negative Sampling (SGNS) objective (see Equation 3). It aims at predicting whether two words can be found close to each other within a text. As shown in Section 2, the process of word embeddings training using SGNS can be divided into two general steps with clear objectives:

Step 1. Search for a low-rank matrix X that provides a good SGNS objective value;

Step 2. Search for a good low-rank representation $X = WC^\top$ in terms of linguistic metrics, where W is a matrix of word embeddings and C is a matrix of so-called context embeddings.

Unfortunately, most previous approaches mixed these two steps into a single one, what entails a not completely correct formulation of the optimization problem. For example, popular approaches to train embeddings (including the original "word2vec" implementation) do not take into account that the objective from Step 1 depends only on the product $X = WC^\top$: instead of straightforward computing of the derivative w.r.t. X, these methods are explicitly based on the derivatives w.r.t. W and C, what complicates the optimization procedure. Moreover, such approaches do not take into account that parametrization $WC^\top$ of matrix X is non-unique and Step 2 is required. Indeed, for any invertible matrix S, we have

$$X = W_1 C_1^\top = W_1 S S^{-1} C_1^\top = W_2 C_2^\top,$$

therefore, solutions $W_1 C_1^\top$ and $W_2 C_2^\top$ are equally good in terms of the SGNS objective but entail different cosine similarities between embeddings and, as a result, different performance in terms of linguistic metrics (see Section 4.2 for details).

A successful attempt to follow the above described steps, which outperforms the original SGNS optimization approach in terms of various linguistic tasks, was proposed in (Levy and Goldberg, 2014). In order to obtain a low-rank matrix X on Step 1, the method reduces the dimensionality of Shifted Positive Pointwise Mutual Informa-

*The first two authors contributed equally to this work

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2028–2036
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1185

tion (SPPMI) matrix via Singular Value Decomposition (SVD). On Step 2, it computes embeddings W and C via a simple formula that depends on the factors obtained by SVD. However, this method has one important limitation: SVD provides a solution to a surrogate optimization problem, which has no direct relation to the SGNS objective. In fact, SVD minimizes the Mean Squared Error (MSE) between X and SPPMI matrix, what does not lead to minimization of SGNS objective in general (see Section 6.1 and Section 4.2 in (Levy and Goldberg, 2014) for details).

These issues bring us to the main idea of our paper: while keeping the low-rank matrix search setup on Step 1, optimize the original SGNS objective directly. This leads to an optimization problem over matrix X with the low-rank constraint, which is often (Mishra et al., 2014) solved by applying *Riemannian optimization* framework (Udriste, 1994). In our paper, we use the projector-splitting algorithm (Lubich and Oseledets, 2014), which is easy to implement and has low computational complexity. Of course, Step 2 may be improved as well, but we regard this as a direction of future work.

As a result, our approach achieves the significant improvement in terms of SGNS optimization on Step 1 and, moreover, the improvement on Step 1 entails the improvement on Step 2 in terms of linguistic metrics. That is why, the proposed two-step decomposition of the problem makes sense, what, most importantly, opens the way to applying even more advanced approaches based on it (e.g., more advanced Riemannian optimization techniques for Step 1 or a more sophisticated treatment of Step 2).

To summarize, the main contributions of our paper are:

- We reformulated the problem of SGNS word embedding learning as a two-step procedure with clear objectives;
- For Step 1, we developed an algorithm based on Riemannian optimization framework that optimizes SGNS objective over low-rank matrix X directly;
- Our algorithm outperforms state-of-the-art competitors in terms of SGNS objective and the semantic similarity linguistic metric (Levy and Goldberg, 2014; Mikolov et al., 2013; Schnabel et al., 2015).

2 Problem Setting

2.1 Skip-Gram Negative Sampling

In this paper, we consider the Skip-Gram Negative Sampling (SGNS) word embedding model (Mikolov et al., 2013), which is a probabilistic discriminative model. Assume we have a text corpus given as a sequence of words $w_1, \dots, w_n$, where n may be larger than 10^{12} and $w_i \in V_W$ belongs to a vocabulary of words V_W. A *context* $c \in V_C$ of the word w_i is a word from set $\{w_{i-L}, ..., w_{i-1}, w_{i+1}, ..., w_{i+L}\}$ for some fixed window size L. Let $\mathbf{w}, \mathbf{c} \in \mathbb{R}^d$ be the *word embeddings* of word w and context c, respectively. Assume they are specified by the following mappings:

$$\mathcal{W} : V_W \to \mathbb{R}^d, \quad \mathcal{C} : V_C \to \mathbb{R}^d.$$

The ultimate goal of SGNS word embedding training is to fit good mappings $\mathcal{W}$ and $\mathcal{C}$.

Let D be a multiset of all word-context pairs observed in the corpus. In the SGNS model, the probability that word-context pair (w, c) is observed in the corpus is modeled as a following dsitribution:

$$P\left(\#(w,c) \neq 0 | w, c\right) = \\ = \sigma(\langle \mathbf{w}, \mathbf{c} \rangle) = \frac{1}{1 + \exp(-\langle \mathbf{w}, \mathbf{c} \rangle)}, \tag{1}$$

where $\#(w, c)$ is the number of times the pair (w, c) appears in D and $\langle \mathbf{x}, \mathbf{y} \rangle$ is the scalar product of vectors $\mathbf{x}$ and $\mathbf{y}$. Number d is a hyperparameter that adjusts the flexibility of the model. It usually takes values from tens to hundreds.

In order to collect a training set, we take all pairs (w, c) from D as positive examples and k randomly generated pairs (w, c) as negative ones. The number of times the word w and the context c appear in D can be computed as

$$\#(w) = \sum_{c \in V_c} \#(w, c),$$

$$\#(c) = \sum_{w \in V_w} \#(w, c)$$

accordingly. Then negative examples are generated from the distribution defined by $\#(c)$ counters:

$$P_D(c) = \frac{\#(c)}{|D|}.$$

In this way, we have a model maximizing the following logarithmic likelihood objective for all word-context pairs (w, c):

$$l_{wc} = \#(w,c)(\log \sigma(\langle \mathbf{w}, \mathbf{c} \rangle) + \\ + k \cdot \mathbb{E}_{c' \sim P_D} \log \sigma(-\langle \mathbf{w}, \mathbf{c}' \rangle)). \quad (2)$$

In order to maximize the objective over all observations for each pair (w, c), we arrive at the following SGNS optimization problem over all possible mappings $\mathcal{W}$ and $\mathcal{C}$:

$$l = \sum_{w \in V_W} \sum_{c \in V_C} (\#(w,c)(\log \sigma(\langle \mathbf{w}, \mathbf{c} \rangle) + \\ + k \cdot \mathbb{E}_{c' \sim P_D} \log \sigma(-\langle \mathbf{w}, \mathbf{c}' \rangle))) \to \max_{\mathcal{W}, \mathcal{C}}. \quad (3)$$

Usually, this optimization is done via the stochastic gradient descent procedure that is performed during passing through the corpus (Mikolov et al., 2013; Rong, 2014).

2.2 Optimization over Low-Rank Matrices

Relying on the prospect proposed in (Levy and Goldberg, 2014), let us show that the optimization problem given by (3) can be considered as a problem of searching for a matrix that maximizes a certain objective function and has the rank-d constraint (Step 1 in the scheme described in Section 1).

2.2.1 SGNS Loss Function

As shown in (Levy and Goldberg, 2014), the logarithmic likelihood (3) can be represented as the sum of $l_{w,c}(\mathbf{w}, \mathbf{c})$ over all pairs (w, c), where $l_{w,c}(\mathbf{w}, \mathbf{c})$ has the following form:

$$l_{w,c}(\mathbf{w}, \mathbf{c}) = \#(w,c) \log \sigma(\langle \mathbf{w}, \mathbf{c} \rangle) + \\ + k \frac{\#(w)\#(c)}{|D|} \log \sigma(-\langle \mathbf{w}, \mathbf{c} \rangle). \quad (4)$$

A crucial observation is that this loss function depends only on the scalar product $\langle \mathbf{w}, \mathbf{c} \rangle$ but not on embeddings $\mathbf{w}$ and $\mathbf{c}$ separately:

$$l_{w,c}(\mathbf{w}, \mathbf{c}) = f_{w,c}(x_{w,c}),$$

where

$$f_{w,c}(x_{w,c}) = a_{w,c} \log \sigma(x_{w,c}) + b_{w,c} \log \sigma(-x_{w,c}),$$

and $x_{w,c}$ is the scalar product $\langle \mathbf{w}, \mathbf{c} \rangle$, and

$$a_{w,c} = \#(w,c), \quad b_{w,c} = k \frac{\#(w)\#(c)}{|D|}$$

are constants.

2.2.2 Matrix Notation

Denote $|V_W|$ as n and $|V_C|$ as m. Let $W \in \mathbb{R}^{n \times d}$ and $C \in \mathbb{R}^{m \times d}$ be matrices, where each row $\mathbf{w} \in \mathbb{R}^d$ of matrix W is the word embedding of the corresponding word w and each row $\mathbf{c} \in \mathbb{R}^d$ of matrix C is the context embedding of the corresponding context c. Then the elements of the product of these matrices

$$X = WC^\top$$

are the scalar products $x_{w,c}$ of all pairs (w, c):

$$X = (x_{w,c}), \quad w \in V_W, c \in V_C.$$

Note that this matrix has rank d, because X equals to the product of two matrices with sizes $(n \times d)$ and $(d \times m)$. Now we can write SGNS objective given by (3) as a function of X:

$$F(X) = \sum_{w \in V_W} \sum_{c \in V_C} f_{w,c}(x_{w,c}), \quad F : \mathbb{R}^{n \times m} \to \mathbb{R}. \quad (5)$$

This arrives us at the following proposition:

Proposition 1 *SGNS optimization problem given by (3) can be rewritten in the following constrained form:*

$$\begin{aligned} &\underset{X \in \mathbb{R}^{n \times m}}{\textit{maximize}} && F(X), \\ &\textit{subject to} && X \in \mathcal{M}_d, \end{aligned} \quad (6)$$

where $\mathcal{M}_d$ is the manifold (Udriste, 1994) of all matrices in $\mathbb{R}^{n \times m}$ with rank d:

$$\mathcal{M}_d = \{X \in \mathbb{R}^{n \times m} : \textit{rank}(X) = d\}.$$

The key idea of this paper is to solve the optimization problem given by (6) via the framework of Riemannian optimization, which we introduce in Section 3.

Important to note that this prospect does not suppose the optimization over parameters W and C directly. This entails the optimization in the space with $((n + m - d) \cdot d)$ degrees of freedom (Mukherjee et al., 2015) instead of $((n+m) \cdot d)$, what simplifies the optimization process (see Section 5 for the experimental results).

2.3 Computing Embeddings from a Low-Rank Solution

Once X is found, we need to recover W and C such that $X = WC^\top$ (Step 2 in the scheme described in Section 1). This problem does not

have a unique solution, since if (W, C) satisfy this equation, then WS^{-1} and $CS^{\top}$ satisfy it as well for any non-singular matrix S. Moreover, different solutions may achieve different values of the linguistic metrics (see Section 4.2 for details). While our paper focuses on Step 1, we use, for Step 2, a heuristic approach that was proposed in (Levy et al., 2015) and it shows good results in practice. We compute SVD of X in the form

$$X = U\Sigma V^{\top},$$

where U and V have orthonormal columns, and Σ is the diagonal matrix, and use

$$W = U\sqrt{\Sigma}, \quad C = V\sqrt{\Sigma}$$

as matrices of embeddings.

A simple justification of this solution is the following: we need to map words into vectors in a way that similar words would have similar embeddings in terms of cosine similarities:

$$\cos(\mathbf{w}_1, \mathbf{w}_2) = \frac{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}{\|\mathbf{w}_1\| \cdot \|\mathbf{w}_2\|}.$$

It is reasonable to assume that two words are similar, if they share contexts. Therefore, we can estimate the similarity of two words w_1, w_2 as

$$s(w_1, w_2) = \sum_{c \in V_C} x_{w_1,c} \cdot x_{w_2,c},$$

what is the element of the matrix $XX^{\top}$ with indices (w_1, w_2). Note that

$$XX^{\top} = U\Sigma V^{\top} V \Sigma U^{\top} = U\Sigma^2 U^{\top}.$$

If we choose $W = U\Sigma$, we exactly obtain $\langle \mathbf{w_1}, \mathbf{w_2} \rangle = s(w_1, w_2)$, since $WW^{\top} = XX^{\top}$ in this case. That is, the cosine similarity of the embeddings $\mathbf{w_1}, \mathbf{w_2}$ coincides with the intuitive similarity $s(w_1, w_2)$. However, scaling by $\sqrt{\Sigma}$ instead of Σ was shown in (Levy et al., 2015) to be a better solution in experiments.

3 Proposed Method

3.1 Riemannian Optimization

3.1.1 General Scheme

The main idea of Riemannian optimization (Udriste, 1994) is to consider (6) as a constrained optimization problem. Assume we have an approximated solution X_i on a current step of the optimization process, where i is the step number. In order to improve X_i, the next step of the standard gradient ascent outputs the point

$$X_i + \nabla F(X_i),$$

where $\nabla F(X_i)$ is the gradient of objective F at the point X_i. Note that the gradient $\nabla F(X_i)$ can be naturally considered as a matrix in $\mathbb{R}^{n \times m}$. Point $X_i + \nabla F(X_i)$ leaves the manifold $\mathcal{M}_d$, because its rank is generally greater than d. That is why Riemannian optimization methods map point $X_i + \nabla F(X_i)$ back to manifold $\mathcal{M}_d$. The standard Riemannian gradient method first projects the gradient step onto the tangent space at the current point X_i and then *retracts* it back to the manifold:

$$X_{i+1} = R\left(P_{\mathcal{T}_M}(X_i + \nabla F(X_i))\right),$$

where R is the *retraction* operator, and $P_{\mathcal{T}_M}$ is the projection onto the tangent space.

Although the optimization problem is non-convex, Riemannian optimization methods show good performance on it. Theoretical properties and convergence guarantees of such methods are discussed in (Wei et al., 2016) more thoroughly.

3.1.2 Projector-Splitting Algorithm

In our paper, we use a simplified version of such approach that retracts point $X_i + \nabla F(X_i)$ directly to the manifold and does not require projection onto the tangent space $P_{\mathcal{T}_M}$ as illustrated in Figure 1:

$$X_{i+1} = R(X_i + \nabla F(X_i)).$$

Intuitively, retractor R finds a rank-d matrix on the manifold $\mathcal{M}_d$ that is similar to high-rank matrix $X_i + \nabla F(X_i)$ in terms of Frobenius norm. How can we do it? The most straightforward way to reduce the rank of $X_i + \nabla F(X_i)$ is to perform the SVD, which keeps d largest singular values of it:

$$\begin{aligned} &1\text{: } U_{i+1}, S_{i+1}, V_{i+1}^{\top} \leftarrow \text{SVD}(X_i + \nabla F(X_i)), \\ &2\text{: } X_{i+1} \leftarrow U_{i+1} S_{i+1} V_{i+1}^{\top}. \end{aligned} \tag{7}$$

However, it is computationally expensive. Instead of this approach, we use the projector-splitting method (Lubich and Oseledets, 2014), which is a second-order retraction onto the manifold (for details, see the review (Absil and Oseledets, 2015)). Its practical implementation is

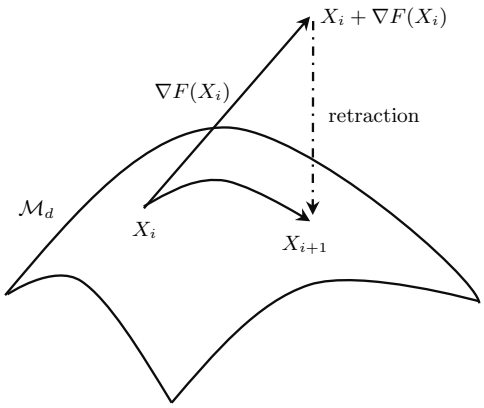


Figure 1: Geometric interpretation of one step of projector-splitting optimization procedure: the gradient step an the retraction of the high-rank matrix $X_i + \nabla F(X_i)$ to the manifold of low-rank matrices $\mathcal{M}_d$.

also quite intuitive: instead of computing the full SVD of $X_i + \nabla F(X_i)$ according to the gradient projection method, we use just one step of the block power numerical method (Bentbib and Kanber, 2015) which computes the SVD, what reduces the computational complexity.

Let us keep the current point in the following factorized form:

$$X_i = U_i S_i V_i^\top, \tag{8}$$

where matrices $U_i \in \mathbb{R}^{n \times d}$ and $V_i \in \mathbb{R}^{m \times d}$ have d orthonormal columns and $S_i \in \mathbb{R}^{d \times d}$. Then we need to perform two QR-decompositions to retract point $X_i + \nabla F(X_i)$ back to the manifold:

$$1: U_{i+1}, S_{i+1} \leftarrow \text{QR}\left((X_i + \nabla F(X_i))V_i\right),$$

$$2: V_{i+1}, S_{i+1}^\top \leftarrow \text{QR}\left((X_i + \nabla F(X_i))^\top U_{i+1}\right),$$

$$3: X_{i+1} \leftarrow U_{i+1} S_{i+1} V_{i+1}^\top.$$

In this way, we always keep the solution $X_{i+1} = U_{i+1} S_{i+1} V_{i+1}^\top$ on the manifold $\mathcal{M}_d$ and in the form (8).

What is important, we only need to compute $\nabla F(X_i)$, so the gradients with respect to U, S and V are never computed explicitly, thus avoiding the subtle case where S is close to singular (so-called singular (critical) point on the manifold). Indeed, the gradient with respect to U (while keeping the orthogonality constraints) can be written (Koch and Lubich, 2007) as:

$$\frac{\partial F}{\partial U} = \frac{\partial F}{\partial X} V S^{-1},$$

which means that the gradient will be large if S is close to singular. The projector-splitting scheme is free from this problem.

3.2 Algorithm

In case of SGNS objective given by (5), an element of gradient ∇F has the form:

$$(\nabla F(X))_{w,c} = \frac{\partial f_{w,c}(x_{w,c})}{\partial x_{w,c}} =$$

$$= \#(w,c) \cdot \sigma\left(-x_{w,c}\right) - k \frac{\#(w)\#(c)}{|D|} \cdot \sigma\left(x_{w,c}\right).$$

To make the method more flexible in terms of convergence properties, we additionally use $\lambda \in \mathbb{R}$, which is a step size parameter. In this case, retractor R returns $X_i + \lambda \nabla F(X_i)$ instead of $X_i + \nabla F(X_i)$ onto the manifold.

The whole optimization procedure is summarized in Algorithm 1.

4 Experimental Setup

4.1 Training Models

We compare our method ("RO-SGNS" in the tables) performance to two baselines: SGNS embeddings optimized via Stochastic Gradient Descent, implemented in the original "word2vec", ("SGD-SGNS" in the tables) (Mikolov et al., 2013) and embeddings obtained by SVD over SPPMI matrix ("SVD-SPPMI" in the tables) (Levy and Goldberg, 2014). We have also experimented with the blockwise alternating optimization over factors W and C, but the results are almost the same to SGD results, that is why we do not to include them into the paper. The source code of our experiments is available online[1].

The models were trained on English Wikipedia "enwik9" corpus[2], which was previously used in most papers on this topic. Like in previous studies, we counted only the words which occur more than 200 times in the training corpus (Levy and Goldberg, 2014; Mikolov et al., 2013). As a result, we obtained a vocabulary of 24292 unique tokens (set of words V_W and set of contexts V_C are equal). The size of the context window was set to 5 for all experiments, as it was done in (Levy and Goldberg, 2014; Mikolov et al., 2013). We conduct three series of experiments: for dimensionality $d = 100$, $d = 200$, and $d = 500$.

[1] https://github.com/AlexGrinch/ro_sgns
[2] http://mattmahoney.net/dc/textdata

Algorithm 1 Riemannian Optimization for SGNS

Require: Dimentionality d, initialization W_0 and C_0, step size λ, gradient function $\nabla F : \mathbb{R}^{n \times m} \to \mathbb{R}^{n \times m}$, number of iterations K

Ensure: Factor $W \in \mathbb{R}^{n \times d}$

$X_0 \leftarrow W_0 C_0^\top$ # get an initial point at the manifold

$U_0, S_0, V_0^\top \leftarrow \text{SVD}(X_0)$ # compute the first point satisfying the low-rank constraint

for $i \leftarrow 1, \dots, K$ **do**

  $U_i, S_i \leftarrow \text{QR}\left((X_{i-1} + \lambda \nabla F(X_{i-1}))V_{i-1}\right)$ # perform one step of the block power method

  $V_i, S_i^\top \leftarrow \text{QR}\left((X_{i-1} + \lambda \nabla F(X_{i-1}))^\top U_i\right)$

  $X_i \leftarrow U_i S_i V_i^\top$ # update the point at the manifold

end for

$U, \Sigma, V^\top \leftarrow \text{SVD}(X_K)$

$W \leftarrow U\sqrt{\Sigma}$ # compute word embeddings

return W

Optimization step size is chosen to be small enough to avoid huge gradient values. However, thorough choice of λ does not result in a significant difference in performance (this parameter was tuned on the training data only, the exact values used in experiments are reported below).

4.2 Evaluation

We evaluate word embeddings via the word similarity task. We use the following popular datasets for this purpose: "wordsim-353" ((Finkelstein et al., 2001); 3 datasets), "simlex-999" (Hill et al., 2016) and "men" (Bruni et al., 2014). Original "wordsim-353" dataset is a mixture of the word pairs for both word similarity and word relatedness tasks. This dataset was split (Agirre et al., 2009) into two intersecting parts: "wordsim-sim" ("ws-sim" in the tables) and "wordsim-rel" ("ws-rel" in the tables) to separate the words from different tasks. In our experiments, we use both of them on a par with the full version of "wordsim-353" ("ws-full" in the tables). Each dataset contains word pairs together with assessor-assigned similarity scores for each pair. As a quality measure, we use Spearman's correlation between these human ratings and cosine similarities for each pair. We call this quality metric *linguistic* in our paper.

5 Results of Experiments

First of all, we compare the value of SGNS objective obtained by the methods. The comparison is demonstrated in Table 1.

We see that SGD-SGNS and SVD-SPPMI methods provide quite similar results, however, the proposed method obtains significantly better SGNS values, what proves the feasibility of using Riemannian optimization framework in SGNS optimization problem. It is interesting to note that SVD-SPPMI method, which does not optimize SGNS objective directly, obtains better results than SGD-SGNS method, which aims at optimizing SGNS. This fact additionally confirms the idea described in Section 2.2.2 that the independent optimization over parameters W and C may decrease the performance.

	$d = 100$	$d = 200$	$d = 500$
SGD-SGNS	-1.68	-1.67	-1.63
SVD-SPPMI	-1.65	-1.65	-1.62
RO-SGNS	$\mathbf{-1.44}$	$\mathbf{-1.43}$	$\mathbf{-1.41}$

Table 1: Comparison of SGNS values (multiplied by 10^{-9}) obtained by the models. Larger is better.

However, the target performance measure of embedding models is the correlation between semantic similarity and human assessment (Section 4.2). Table 2 presents the comparison of the methods in terms of it. We see that our method outperforms the competitors on all datasets except for "men" dataset where it obtains slightly worse results. Moreover, it is important that the higher dimension entails higher performance gain of our method in comparison to the competitors.

To understand how our model improves or degrades the performance in comparison to the baseline, we found several words, whose neighbors in terms of cosine distance change significantly. Table 3 demonstrates neighbors of the words "five", "he" and "main" for both SVD-SPPMI and RO-SGNS models. A neighbor is marked bold if we suppose that it has similar semantic meaning to the

Dim. d	Algorithm	ws-sim	ws-rel	ws-full	simlex	men
$d = 100$	SGD-SGNS	0.719	0.570	0.662	0.288	0.645
	SVD-SPPMI	0.722	0.585	0.669	0.317	**0.686**
	RO-SGNS	**0.729**	**0.597**	**0.677**	**0.322**	0.683
$d = 200$	SGD-SGNS	0.733	0.584	0.677	0.317	0.664
	SVD-SPPMI	0.747	0.625	0.694	0.347	**0.710**
	RO-SGNS	**0.757**	**0.647**	**0.708**	**0.353**	0.701
$d = 500$	SGD-SGNS	0.738	0.600	0.688	0.350	0.712
	SVD-SPPMI	0.765	0.639	0.707	0.380	**0.737**
	RO-SGNS	**0.767**	**0.654**	**0.715**	**0.383**	0.732

Table 2: Comparison of the methods in terms of the semantic similarity task. Each entry represents the Spearman's correlation between predicted similarities and the manually assessed ones.

five				he				main			
SVD-SPPMI		RO-SGNS		SVD-SPPMI		RO-SGNS		SVD-SPPMI		RO-SGNS	
Neighbors	Dist.	Neighbors	Dist.	Neighbors	Dist.	Neighbors	Dist.	Neighbors	Dist.	Neighbors	Dist.
lb	0.748	**four**	0.999	**she**	0.918	when	0.904	**major**	0.631	**major**	0.689
kg	0.731	**three**	0.999	was	0.797	had	0.903	busiest	0.621	**important**	0.661
mm	0.670	**six**	0.997	promptly	0.742	was	0.901	**principal**	0.607	line	0.631
mk	0.651	**seven**	0.997	having	0.731	who	0.892	nearest	0.607	external	0.624
lbf	0.650	**eight**	0.996	dumbledore	0.731	**she**	0.884	connecting	0.591	**principal**	0.618
per	0.644	and	0.985	**him**	0.730	by	0.880	linking	0.588	**primary**	0.612

Table 3: Examples of the semantic neighbors obtained for words "five", "he" and "main".

usa					
SGD-SGNS		SVD-SPPMI		RO-SGNS	
Neighbors	Dist.	Neighbors	Dist.	Neighbors	Dist.
akron	0.536	**wisconsin**	0.700	**georgia**	0.707
midwest	0.535	**delaware**	0.693	**delaware**	0.706
burbank	0.534	**ohio**	0.691	**maryland**	0.705
nevada	0.534	northeast	0.690	**illinois**	0.704
arizona	0.533	cities	0.688	madison	0.703
uk	0.532	southwest	0.684	**arkansas**	0.699
youngstown	0.532	places	0.684	**dakota**	0.690
utah	0.530	counties	0.681	**tennessee**	0.689
milwaukee	0.530	**maryland**	0.680	northeast	0.687
headquartered	0.527	**dakota**	0.674	**nebraska**	0.686

Table 4: Examples of the semantic neighbors from 11th to 20th obtained for the word "usa" by all three methods. Top-10 neighbors for all three methods are exact names of states.

source word. First of all, we notice that our model produces much better neighbors of the words describing digits or numbers (see word "five" as an example). Similar situation happens for many other words, e.g. in case of "main" — the nearest neighbors contain 4 similar words for our model instead of 2 in case of SVD-SPPMI. The neighbourhood of "he" contains less semantically similar words in case of our model. However, it filters out irrelevant words, such as "promptly" and "dumbledore".

Table 4 contains the nearest words to the word "usa" from 11th to 20th. We marked names of USA states bold and did not represent top-10 nearest words as they are exactly names of states for all three models. Some non-bold words are arguably relevant as they present large USA cities ("akron", "burbank", "madison") or geographical regions of several states ("midwest", "northeast", "southwest"), but there are also some completely irrelevant words ("uk", "cities", "places") presented by first two models.

Our experiments show that the optimal number of iterations K in the optimization procedure and step size λ depend on the particular value of d. For $d = 100$, we have $K = 7, \lambda = 5 \cdot 10^{-5}$, for $d = 200$, we have $K = 8, \lambda = 5 \cdot 10^{-5}$, and for $d = 500$, we have $K = 2, \lambda = 10^{-4}$. Moreover, the best results were obtained when SVD-SPPMI embeddings were used as an initialization of Riemannian optimization process.

Figure 2 illustrates how the correlation between semantic similarity and human assessment scores changes through iterations of our method. Optimal value of K is the same for both whole testing set and its 10-fold subsets chosen for cross-validation. The idea to stop optimization procedure on some iteration is also discussed in (Lai et al., 2015).

Training of the same dimensional models ($d = 500$) on English Wikipedia corpus using SGD-SGNS, SVD-SPPMI, RO-SGNS took 20 minutes, 10 minutes and 70 minutes respectively. Our method works slower, but not significantly. Moreover, since we were not focused on the code efficiency optimization, this time can be reduced.

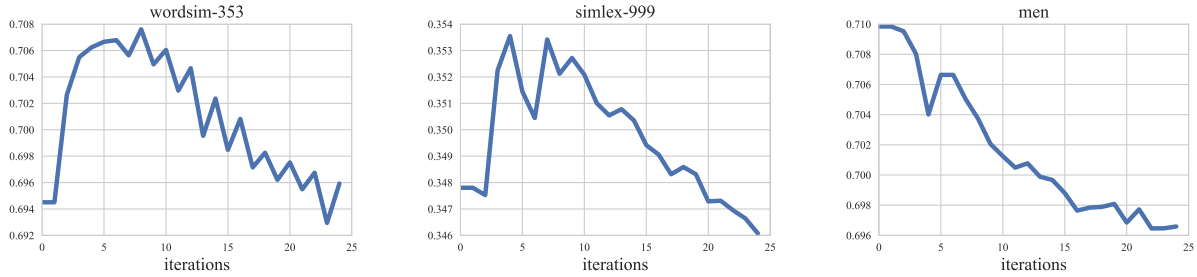


Figure 2: Illustration of why it is important to choose the optimal iteration and stop optimization procedure after it. The graphs show semantic similarity metric in dependence on the iteration of optimization procedure. The embeddings obtained by SVD-SPPMI method were used as initialization. Parameters: $d = 200$, $\lambda = 5 \cdot 10^{-5}$.

6 Related Work

6.1 Word Embeddings

Skip-Gram Negative Sampling was introduced in (Mikolov et al., 2013). The "negative sampling" approach is thoroughly described in (Goldberg and Levy, 2014), and the learning method is explained in (Rong, 2014). There are several open-source implementations of SGNS neural network, which is widely known as "word2vec". [1][2]

As shown in Section 2.2, Skip-Gram Negative Sampling optimization can be reformulated as a problem of searching for a low-rank matrix. In order to be able to use out-of-the-box SVD for this task, the authors of (Levy and Goldberg, 2014) used the surrogate version of SGNS as the objective function. There are two general assumptions made in their algorithm that distinguish it from the SGNS optimization:

1. SVD optimizes Mean Squared Error (MSE) objective instead of SGNS loss function.

2. In order to avoid infinite elements in SPMI matrix, it is transformed in ad-hoc manner (SPPMI matrix) before applying SVD.

This makes the objective not interpretable in terms of the original task (3). As mentioned in (Levy and Goldberg, 2014), SGNS objective weighs different (w, c) pairs differently, unlike the SVD, which works with the same weight for all pairs and may entail the performance fall. The comprehensive explanation of the relation between SGNS and SVD-SPPMI methods is provided in (Keerthi et al., 2015). (Lai et al., 2015; Levy et al., 2015) give a good overview of highly practical methods to improve these word embedding models.

6.2 Riemannian Optimization

An introduction to optimization over Riemannian manifolds can be found in (Udriste, 1994). The overview of retractions of high rank matrices to low-rank manifolds is provided in (Absil and Oseledets, 2015). The projector-splitting algorithm was introduced in (Lubich and Oseledets, 2014), and also was mentioned in (Absil and Oseledets, 2015) as "Lie-Trotter retraction".

Riemannian optimization is succesfully applied to various data science problems: for example, matrix completion (Vandereycken, 2013), large-scale recommender systems (Tan et al., 2014), and tensor completion (Kressner et al., 2014).

7 Conclusions

In our paper, we proposed the general two-step scheme of training SGNS word embedding model and introduced the algorithm that performs the search of a solution in the low-rank form via Riemannian optimization framework. We also demonstrated the superiority of our method by providing experimental comparison to existing state-of-the-art approaches.

Possible direction of future work is to apply more advanced optimization techniques to the Step 1 of the scheme proposed in Section 1 and to explore the Step 2 — obtaining embeddings with a given low-rank matrix.

Acknowledgments

This research was supported by the Ministry of Education and Science of the Russian Federation (grant 14.756.31.0001).

[1] Original Google word2vec: https://code.google.com/archive/p/word2vec/

[2] Gensim word2vec: https://radimrehurek.com/gensim/models/word2vec.html

References

P-A Absil and Ivan V Oseledets. 2015. Low-rank retractions: a survey and new results. *Computational Optimization and Applications* 62(1):5–29.

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL*. pages 19–27.

AH Bentbib and A Kanber. 2015. Block power method for svd decomposition. *Analele Stiintifice Ale Unversitatii Ovidius Constanta-Seria Matematica* 23(2):45–58.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)* 49(1-47).

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*. pages 406–414.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* .

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* .

S Sathiya Keerthi, Tobias Schnabel, and Rajiv Khanna. 2015. Towards a better understanding of predict and count models. *arXiv preprint arXiv:1511.02024* .

Othmar Koch and Christian Lubich. 2007. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.* 29(2):434–454.

Daniel Kressner, Michael Steinlechner, and Bart Vandereycken. 2014. Low-rank tensor completion by riemannian optimization. *BIT Numerical Mathematics* 54(2):447–468.

Siwei Lai, Kang Liu, Shi He, and Jun Zhao. 2015. How to generate a good word embedding? *arXiv preprint arXiv:1507.05523* .

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*. pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *ACL* 3:211–225.

Christian Lubich and Ivan V Oseledets. 2014. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numerical Mathematics* 54(1):171–188.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. pages 3111–3119.

Bamdev Mishra, Gilles Meyer, Silvère Bonnabel, and Rodolphe Sepulchre. 2014. Fixed-rank matrix factorizations and riemannian low-rank optimization. *Computational Statistics* 29(3-4):591–621.

A Mukherjee, K Chen, N Wang, and J Zhu. 2015. On the degrees of freedom of reduced-rank estimators in multivariate regression. *Biometrika* 102(2):457–477.

Xin Rong. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738* .

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *EMNLP*.

Mingkui Tan, Ivor W Tsang, Li Wang, Bart Vandereycken, and Sinno Jialin Pan. 2014. Riemannian pursuit for big matrix recovery. In *ICML*. volume 32, pages 1539–1547.

Constantin Udriste. 1994. *Convex functions and optimization methods on Riemannian manifolds*, volume 297. Springer Science & Business Media.

Bart Vandereycken. 2013. Low-rank matrix completion by riemannian optimization. *SIAM Journal on Optimization* 23(2):1214–1236.

Ke Wei, Jian-Feng Cai, Tony F Chan, and Shingyu Leung. 2016. Guarantees of riemannian optimization for low rank matrix recovery. *SIAM Journal on Matrix Analysis and Applications* 37(3):1198–1222.

Deep Multitask Learning for Semantic Dependency Parsing

Hao Peng* **Sam Thomson†** **Noah A. Smith***
*Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA
†School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
{hapeng,nasmith}@cs.washington.edu, sthomson@cs.cmu.edu

Abstract

We present a deep neural architecture that parses sentences into three semantic dependency graph formalisms. By using efficient, nearly arc-factored inference and a bidirectional-LSTM composed with a multi-layer perceptron, our base system is able to significantly improve the state of the art for semantic dependency parsing, without using hand-engineered features or syntax. We then explore two multitask learning approaches—one that shares parameters across formalisms, and one that uses higher-order structures to predict the graphs jointly. We find that both approaches improve performance across formalisms on average, achieving a new state of the art. Our code is open-source and available at https://github.com/Noahs-ARK/NeurboParser.

1 Introduction

Labeled directed graphs are a natural and flexible representation for semantics (Copestake et al., 2005; Baker et al., 2007; Surdeanu et al., 2008; Banarescu et al., 2013, *inter alia*). Their generality over trees, for instance, allows them to represent relational semantics while handling phenomena like coreference and coordination. Even syntactic formalisms are moving toward graphs (de Marneffe et al., 2014). However, full semantic graphs can be expensive to annotate, and efforts are fragmented across competing semantic theories, leading to a limited number of annotations in any one formalism. This makes learning to parse more difficult, especially for powerful but data-hungry machine learning techniques like neural networks.

In this work, we hypothesize that the overlap among theories and their corresponding representations can be exploited using multitask learning (Caruana, 1997), allowing us to learn from more data. We use the 2015 SemEval shared task on Broad-Coverage Semantic Dependency Parsing (**SDP**; Oepen et al., 2015) as our testbed. The shared task provides an English-language corpus with parallel annotations for three semantic graph representations, described in §2. Though the shared task was designed in part to encourage comparison between the formalisms, we are the first to treat SDP as a multitask learning problem.

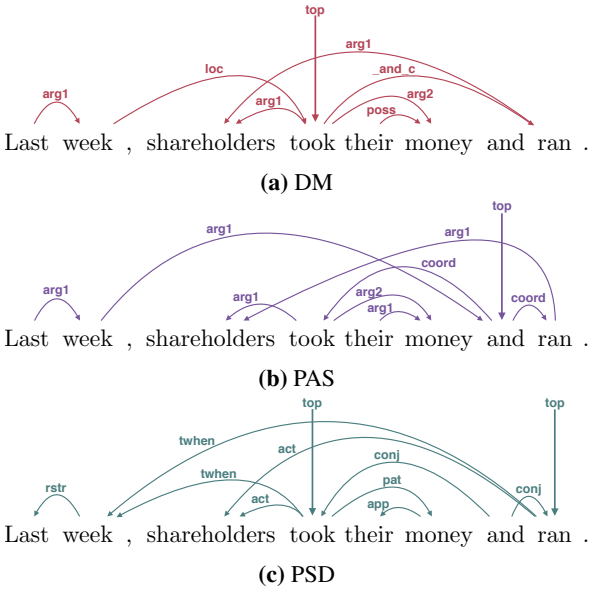


Figure 1: An example sentence annotated with the three semantic formalisms of the broad-coverage semantic dependency parsing shared tasks.

As a strong baseline, we introduce a new system that parses each formalism separately (§3). It uses a bidirectional-LSTM composed with a multi-layer perceptron to score arcs and predicates, and has efficient, nearly arc-factored inference. Experiments show it significantly improves on state-of-the-art methods (§3.4).

We then present two multitask extensions (§4.2

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2037–2048
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1186

	DM		PAS		PSD	
	id	ood	id	ood	id	ood
# labels	59	47	42	41	91	74
% trees	2.3	9.7	1.2	2.4	42.2	51.4
% projective	2.9	8.8	1.6	3.5	41.9	54.4

Table 1: Graph statistics for in-domain (WSJ, "id") and out-of-domain (Brown corpus, "ood") data. Numbers taken from Oepen et al. (2015).

and §4.3), with a parameterization and factorization that implicitly models the relationship between multiple formalisms. Experiments show that both techniques improve over our basic model, with an additional (but smaller) improvement when they are combined (§4.5). Our analysis shows that the improvement in unlabeled F_1 is greater for the two formalisms that are more structurally similar, and suggests directions for future work. Finally, we survey related work (§5), and summarize our contributions and findings (§6).

2 Broad-Coverage Semantic Dependency Parsing (SDP)

First defined in a SemEval 2014 shared task (Oepen et al., 2014), and then extended by Oepen et al. (2015), the broad-coverage semantic depency parsing (**SDP**) task is centered around three semantic formalisms whose annotations have been converted into bilexical dependencies. See Figure 1 for an example. The formalisms come from varied linguistic traditions, but all three aim to capture predicate-argument relations between content-bearing words in a sentence.

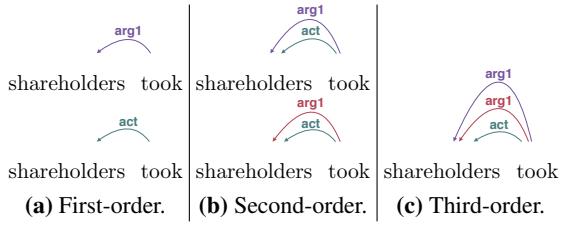
While at first glance similar to syntactic dependencies, semantic dependencies have distinct goals and characteristics, more akin to semantic role labeling (SRL; Gildea and Jurafsky, 2002) or the abstract meaning representation (AMR; Banarescu et al., 2013). They abstract over different syntactic realizations of the same or similar meaning (e.g., *"She gave me the ball."* vs. *"She gave the ball to me."*). Conversely, they attempt to distinguish between different senses even when realized in similar syntactic forms (e.g., *"I baked in the kitchen."* vs. *"I baked in the sun."*).

Structurally, they are labeled directed graphs whose vertices are tokens in the sentence. This is in contrast to AMR whose vertices are abstract concepts, with no explicit alignment to tokens, which makes parsing more difficult (Flanigan et al., 2014). Their arc labels encode broadly-applicable semantic relations rather than being tailored to any specific downstream application or ontology.[1] They are not necessarily trees, because a token may be an argument of more than one predicate (e.g., in *"John wants to eat,"* John is both the wanter and the would-be eater). Their analyses may optionally leave out non–content-bearing tokens, such as punctuation or the infinitival *"to,"* or prepositions that simply mark the type of relation holding between other words. But when restricted to content-bearing tokens (including adjectives, adverbs, etc.), the subgraph is connected. In this sense, SDP provides a *whole-sentence* analysis. This is in contrast to PropBank-style SRL, which gives an analysis of only verbal and nominal predicates (Palmer et al., 2005). Semantic dependency graphs also tend to have higher levels of nonprojectivity than syntactic trees (Oepen et al., 2014). Sentences with graphs containing cycles have been removed from the dataset by the organizers, so all remaining graphs are directed acyclic graphs. Table 1 summarizes some of the dataset's high-level statistics.

Formalisms. Following the SemEval shared tasks, we consider three formalisms. The **DM** (DELPH-IN MRS) representation comes from DeepBank (Flickinger et al., 2012), which are manually-corrected parses from the LinGO English Resource Grammar (Copestake and Flickinger, 2000). LinGO is a head-driven phrase structure grammar (HPSG; Pollard and Sag, 1994) with minimal recursion semantics (Copestake et al., 2005). The **PAS** (Predicate-Argument Structures) representation is extracted from the Enju Treebank, which consists of automatic parses from the Enju HPSG parser (Miyao, 2006). PAS annotations are also available for the Penn Chinese Treebank (Xue et al., 2005). The **PSD** (Prague Semantic Dependencies) representation is extracted from the tectogrammatical layer of the Prague Czech-English Dependency Treebank (Hajič et al., 2012). PSD annotations are also available for a Czech translation of the WSJ Corpus. In this work, we train and evaluate only on English annotations.

Of the three, PAS follows syntax most closely, and prior work has found it the easiest to predict. PSD has the largest set of labels, and parsers

[1] This may make another disambiguation step necessary to use these representations in a downstream task, but there is evidence that modeling semantic composition separately from grounding in any ontology is an effective way to achieve broad coverage (Kwiatkowski et al., 2013).



(a) First-order. **(b)** Second-order. **(c)** Third-order.

Figure 2: Examples of local structures. We refer to the number of arcs that a structure contains as its **order**.

have significantly lower performance on it (Oepen et al., 2015).

3 Single-Task SDP

Here we introduce our basic model, in which training and prediction for each formalism is kept completely separate. We also lay out basic notation, which will be reused for our multitask extensions.

3.1 Problem Formulation

The output of semantic dependency parsing is a labeled directed graph (see Figure 1). Each arc has a label from a predefined set $\mathcal{L}$, indicating the semantic relation of the child to the head. Given input sentence x, let $\mathcal{Y}(x)$ be the set of possible semantic graphs over x. The graph we seek maximizes a score function S:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}(x)} S(x, y), \tag{1}$$

We decompose S into a sum of local scores s for **local structures** (or "parts") p in the graph:

$$S(x, y) = \sum_{p \in y} s(p). \tag{2}$$

For notational simplicity, we omit the dependence of s on x. See Figure 2a for examples of local structures. s is a parameterized function, whose parameters (denoted Θ and suppressed here for clarity) will be learned from the training data (§3.3). Since we search over every possible labeled graph (i.e., considering each labeled arc for each pair of words), our approach can be considered a **graph-based** (or **all-pairs**) method. The models presented in this work all share this common graph-based approach, differing only in the set of structures they score and in the parameterization of the scoring function s. This approach also underlies state-of-the-art approaches to SDP (Martins and Almeida, 2014).

3.2 Basic Model

Our basic model is inspired by recent successes in neural arc-factored graph-based dependency parsing (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017; Kuncoro et al., 2016). It borrows heavily from the neural arc-scoring architectures in those works, but decodes with a different algorithm under slightly different constraints.

3.2.1 Basic Structures

Our basic model factors over three types of structures (p in Equation 2):

- predicate, indicating a predicate word, denoted $i{\rightarrow}\cdot$;
- unlabeled arc, representing the existence of an arc from a predicate to an argument, denoted $i{\rightarrow}j$;
- labeled arc, an arc labeled with a semantic role, denoted $i \xrightarrow{\ell} j$.

Here i and j are word indices in a given sentence, and ℓ indicates the arc label. This list corresponds to the most basic structures used by Martins and Almeida (2014). Selecting an output y corresponds precisely to selecting which instantiations of these structures are included.

To ensure the internal consistency of predictions, the following constraints are enforced during decoding:

- $i{\rightarrow}\cdot$ if and only if there exists at least one j such that $i{\rightarrow}j$;
- If $i{\rightarrow}j$, then there must be exactly one label ℓ such that $i \xrightarrow{\ell} j$. Conversely, if not $i{\rightarrow}j$, then there must not exist any $i \xrightarrow{\ell} j$;

We also enforce a **determinism** constraint (Flanigan et al., 2014): certain labels must not appear on more than one arc emanating from the same token. The set of deterministic labels is decided based on their appearance in the training set. Notably, we do not enforce that the predicted graph is connected or spanning. If not for the predicate and determinism constraints, our model would be **arc-factored**, and decoding could be done for each i, j pair independently. Our structures do overlap though, and we employ AD^3 (Martins et al., 2011) to find the highest-scoring internally consistent semantic graph. AD^3 is an approximate discrete optimization algorithm based on dual decomposition. It can be used to decode factor graphs over discrete variables when scored structures overlap, as is the case here.

3.2.2 Basic Scoring

Similarly to Kiperwasser and Goldberg (2016), our model learns representations of tokens in a sentence using a bi-directional LSTM (BiLSTM). Each different type of structure (predicate, unlabeled arc, labeled arc) then shares these same BiLSTM representations, feeding them into a multilayer perceptron (MLP) which is specific to the structure type. We present the architecture slightly differently from prior work, to make the transition to the multitask scenario (§4) smoother. In our presentation, we separate the model into a function ϕ that represents the input (corresponding to the BiLSTM and the initial layers of the MLPs), and a function ψ that represents the output (corresponding to the final layers of the MLPs), with the scores given by their inner product.[2]

Distributed input representations. Long short-term memory networks (LSTMs) are a variant of recurrent neural networks (RNNs) designed to alleviate the vanishing gradient problem in RNNs (Hochreiter and Schmidhuber, 1997). A bi-directional LSTM (BiLSTM) runs over the sequence in both directions (Schuster and Paliwal, 1997; Graves, 2012).

Given an input sentence x and its corresponding part-of-speech tag sequence, each token is mapped to a concatenation of its word embedding vector and POS tag vector. Two LSTMs are then run in opposite directions over the input vector sequence, outputting the concatenation of the two hidden vectors at each position i: $\mathbf{h}_i = \left[\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i\right]$ (we omit $\mathbf{h}_i$'s dependence on x and its own parameters). $\mathbf{h}_i$ can be thought of as an encoder that contextualizes each token conditioning on all of its context, without any Markov assumption. $\mathbf{h}$'s parameters are learned jointly with the rest of the model (§3.3); we refer the readers to Cho (2015) for technical details.

The input representation ϕ of a predicate structure depends on the representation of one word:

$$\phi(i{\rightarrow}\cdot) = \tanh\left(\mathbf{C}_{\text{pred}}\mathbf{h}_i + \mathbf{b}_{\text{pred}}\right). \quad (3a)$$

[2]For clarity, we present single-layer BiLSTMs and MLPs, while in practice we use two layers for both.

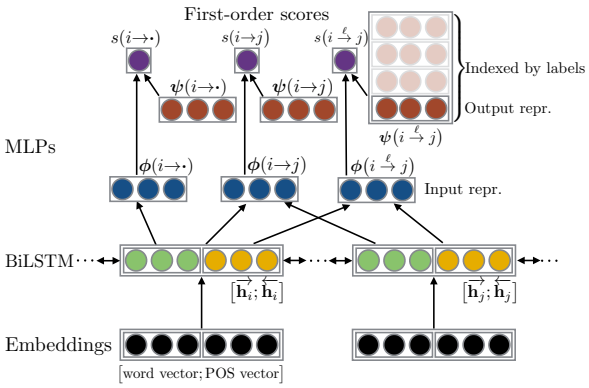


Figure 3: Illustration of the architecture of the basic model. i and j denote the indices of tokens in the given sentence. The figure depicts single-layer BiLSTM and MLPs, while in practice we use two layers for both.

For unlabeled arc and labeled arc structures, it depends on both the head and the modifier (but not the label, which is captured in the distributed output representation):

$$\phi(i{\rightarrow}j) = \tanh\left(\mathbf{C}_{\text{UA}}\left[\mathbf{h}_i; \mathbf{h}_j\right] + \mathbf{b}_{\text{UA}}\right), \quad (3b)$$

$$\phi(i \overset{\ell}{\rightarrow} j) = \tanh\left(\mathbf{C}_{\text{LA}}\left[\mathbf{h}_i; \mathbf{h}_j\right] + \mathbf{b}_{\text{LA}}\right). \quad (3c)$$

Distributed output representations. NLP researchers have found that embedding discrete output labels into a low dimensional real space is an effective way to capture commonalities among them (Srikumar and Manning, 2014; Hermann et al., 2014; FitzGerald et al., 2015, *inter alia*). In neural language models (Bengio et al., 2003; Mnih and Hinton, 2007, *inter alia*) the weights of the output layer could also be regarded as an output embedding.

We associate each first-order structure p with a d-dimensional real vector $\psi(p)$ which does not depend on particular words in p. Predicates and unlabeled arcs are each mapped to a single vector:

$$\psi(i{\rightarrow}\cdot) = \psi_{\text{pred}}, \quad (4a)$$

$$\psi(i{\rightarrow}j) = \psi_{\text{UA}}, \quad (4b)$$

and each label gets a vector:

$$\psi(i \overset{\ell}{\rightarrow} j) = \psi_{\text{LA}}(\ell). \quad (4c)$$

Scoring. Finally, we use an inner product to score first-order structures:

$$s(p) = \phi(p) \cdot \psi(p). \quad (5)$$

Figure 3 illustrates our basic model's architecture.

3.3 Learning

The parameters of the model are learned using a max-margin objective. Informally, the goal is to learn parameters for the score function so that the gold parse is scored over every incorrect parse with a margin proportional to the cost of the incorrect parse. More formally, let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ be the training set consisting of N pairs of sentence x_i and its gold parse y_i. Training is then the following ℓ_2-regularized empirical risk minimization problem:

$$\min_{\Theta} \frac{\lambda}{2}\|\Theta\|^2 + \frac{1}{N}\sum_{i=1}^{N} L(x_i, y_i; \Theta), \qquad (6)$$

where Θ is all parameters in the model, and L is the structured hinge loss:

$$L(x_i, y_i; \Theta) = \max_{y \in \mathcal{Y}(x_i)} \{S(x_i, y) + c(y, y_i)\} - S(x_i, y_i). \qquad (7)$$

c is a weighted Hamming distance that trades off between precision and recall (Taskar et al., 2004). Following Martins and Almeida (2014), we encourage recall over precision by using the costs 0.6 for false negative arc predictions and 0.4 for false positives.

3.4 Experiments

We evaluate our basic model on the English dataset from SemEval 2015 Task 18 closed track.[3] We split as in previous work (Almeida and Martins, 2015; Du et al., 2015), resulting in 33,964 training sentences from §00–19 of the WSJ corpus, 1,692 development sentences from §20, 1,410 sentences from §21 as in-domain test data, and 1,849 sentences sampled from the Brown Corpus as out-of-domain test data.

The *closed* track differs from the *open* and *gold* tracks in that it does not allow access to any syntactic analyses. In the open track, additional machine generated syntactic parses are provided, while the gold-track gives access to various gold-standard syntactic analyses. Our model is evaluated with closed track data; it does not have access to any syntactic analyses during training or test.

We refer the readers to §4.4 for implementation details, including training procedures, hyper-parameters, pruning techniques, etc..

[3] http://sdp.delph-in.net

[4] Paired bootstrap, $p < 0.05$ after Bonferroni correction.

	Model	DM	PAS	PSD	Avg.
id	Du et al., 2015	89.1	91.3	75.7	86.3
	A&M, 2015	88.2	90.9	76.4	86.0
	BASIC	**89.4**	**92.2**	**77.6**	**87.4**
ood	Du et al., 2015	81.8	87.2	73.3	81.7
	A&M, 2015	81.8	86.9	74.8	82.0
	BASIC	**84.5**	**88.3**	**75.3**	**83.6**

Table 2: Labeled parsing performance (F_1 score) on both in-domain (id) and out-of-domain (ood) test data. The last column shows the micro-average over the three tasks. Bold font indicates best performance without syntax. Underlines indicate statistical significance with Bonferroni (1936) correction compared to the best baseline system.[4]

Empirical results. As our model uses no explicit syntactic information, the most comparable models to ours are two state-of-the-art closed track systems due to Du et al. (2015) and Almeida and Martins (2015). Du et al. (2015) rely on graph-tree transformation techniques proposed by Du et al. (2014), and apply a voting ensemble to well-studied tree-oriented parsers. Closely related to ours is Almeida and Martins (2015), who used rich, hand-engineered second-order features and AD^3 for inference.

Table 2 compares our basic model to both baseline systems (labeled F_1 score) on SemEval 2015 Task 18 test data. Scores of those systems are repeated from the official evaluation results. Our basic model significantly outperforms the best published results with a 1.1% absolute improvement on the in-domain test set and 1.6% on the out-of-domain test set.

4 Multitask SDP

We introduce two extensions to our single-task model, both of which use training data for all three formalisms to improve performance on each formalism's parsing task. We describe a first-order model, where representation functions are enhanced by parameter sharing while inference is kept separate for each task (§4.2). We then introduce a model with cross-task higher-order structures that uses joint inference *across* different tasks (§4.3). Both multitask models use AD^3 for decoding, and are trained with the same margin-based objective, as in our single-task model.

4.1 Problem Formulation

We will use an additional superscript $t \in \mathcal{T}$ to distinguish the three tasks (e.g., $y^{(t)}$, $\boldsymbol{\phi}^{(t)}$), where $\mathcal{T} = \{\text{DM}, \text{PAS}, \text{PSD}\}$. Our task is now to predict three graphs $\{y^{(t)}\}_{t\in\mathcal{T}}$ for a given input sentence x. Multitask SDP can also be understood as parsing x into a single unified *multigraph* $y = \bigcup_{t\in\mathcal{T}} y^{(t)}$. Similarly to Equations 1–2, we decompose y's score $S(x, y)$ into a sum of local scores for local structures in y, and we seek a multigraph $\hat{y}$ that maximizes $S(x, y)$.

4.2 Multitask SDP with Parameter Sharing

A common approach when using BiLSTMs for multitask learning is to share the BiLSTM part of the model across tasks, while training specialized classifiers for each task (Søgaard and Goldberg, 2016). In this spirit, we let each task keep its own specialized MLPs, and explore two variants of our model that share parameters at the BiLSTM level.

The first variant consists of a set of task-specific BiLSTM encoders as well as a common one that is shared across all tasks. We denote it FREDA. FREDA uses a neural generalization of "frustratingly easy" domain adaptation (Daumé III, 2007; Kim et al., 2016), where one augments domain-specific features with a shared set of features to capture global patterns. Formally, let $\{\mathbf{h}^{(t)}\}_{t\in\mathcal{T}}$ denote the three task-specific encoders. We introduce another encoder $\widetilde{\mathbf{h}}$ that is shared across all tasks. Then a new set of input functions $\{\boldsymbol{\phi}^{(t)}\}_{t\in\mathcal{T}}$ can be defined as in Equations 3a–3c, for example:

$$\boldsymbol{\phi}^{(t)}(i \overset{\ell}{\rightarrow} j) = \tanh\big(\mathbf{C}^{(t)}_{\text{LA}}\big[\mathbf{h}^{(t)}_i; \mathbf{h}^{(t)}_j; \widetilde{\mathbf{h}}_i; \widetilde{\mathbf{h}}_j\big] + \mathbf{b}^{(t)}_{\text{LA}}\big). \quad (8)$$

The predicate and unlabeled arc versions are analogous. The output representations $\{\boldsymbol{\psi}^{(t)}\}$ remain task-specific, and the score is still the inner product between the input representation and the output representation.

The second variant, which we call SHARED, uses *only* the shared encoder $\widetilde{\mathbf{h}}$, and doesn't use task-specific encoders $\{\mathbf{h}^{(t)}\}$. It can be understood as a special case of FREDA where the dimensions of the task-specific encoders are 0.

4.3 Multitask SDP with Cross-Task Structures

In syntactic parsing, higher-order structures have commonly been used to model interactions between multiple adjacent arcs in the same dependency tree (Carreras, 2007; Smith and Eisner, 2008; Martins et al., 2009; Zhang et al., 2014, *inter alia*). Lluís et al. (2013), in contrast, used second-order structures to jointly model syntactic dependencies and semantic roles. Similarly, we use higher-order structures *across* tasks instead of *within* tasks. In this work, we look at interactions between arcs that share the same head and modifier.[5] See Figures 2b and 2c for examples of higher-order cross-task structures.

Higher-order structure scoring. Borrowing from Lei et al. (2014), we introduce a low-rank tensor scoring strategy that, given a higher-order structure p, models interactions between the first-order structures (i.e., arcs) p is made up of. This approach builds on and extends the parameter sharing techniques in §4.2. It can either follow FREDA or SHARED to get the input representations for first-order structures.

We first introduce basic tensor notation. The **order** of a tensor is the number of its dimensions. The **outer product** of two vectors forms a second-order tensor (matrix) where $[\mathbf{u} \otimes \mathbf{v}]_{i,j} = u_i v_j$. We denote the **inner product** of two tensors of the same dimensions by $\langle \cdot, \cdot \rangle$, which first takes their element-wise product, then sums all the elements in the resulting tensor.

For example, let p be a labeled third-order structure, including one labeled arc from each of the three different tasks: $p = \{p^{(t)}\}_{t\in\mathcal{T}}$. Intuitively, $s(p)$ should capture every pairwise interaction between the three input and three output representations of p. Formally, we want the score function to include a parameter for each term in the outer product of the representation vectors: $s(p) =$

$$\left\langle \mathcal{W}, \bigotimes_{t\in\mathcal{T}} \left(\boldsymbol{\phi}^{(t)}\left(p^{(t)}\right) \otimes \boldsymbol{\psi}^{(t)}\left(p^{(t)}\right)\right)\right\rangle, \quad (9)$$

where $\mathcal{W}$ is a sixth-order tensor of parameters.[6]

With typical dimensions of representation vectors, this leads to an unreasonably large number of

[5] In the future we hope to model structures over larger motifs, both across and within tasks, to potentially capture when an arc in one formalism corresponds to a path in another formalism, for example.

[6] This is, of course, not the only way to model interactions between several representations. For instance, one could concatenate them and feed them into another MLP. Our preliminary experiments in this direction suggested that it may be less effective given a similar number of parameters, but we did not run full experiments.

parameters. Following Lei et al. (2014), we upper-bound the rank of $\mathcal{W}$ by r to limit the number of parameters (r is a hyperparameter, decided empirically). Using the fact that a tensor of rank at most r can be decomposed into a sum of r rank-1 tensors (Hitchcock, 1927), we reparameterize $\mathcal{W}$ to enforce the low-rank constraint by construction:

$$\mathcal{W} = \sum_{j=1}^{r} \bigotimes_{t \in \mathcal{T}} \left(\left[\mathbf{U}_{\text{LA}}^{(t)} \right]_{j,:} \otimes \left[\mathbf{V}_{\text{LA}}^{(t)} \right]_{j,:} \right), \quad (10)$$

where $\mathbf{U}_{\text{LA}}^{(t)}, \mathbf{V}_{\text{LA}}^{(t)} \in \mathbb{R}^{r \times d}$ are now our parameters. $[\cdot]_{j,:}$ denotes the jth row of a matrix. Substituting this back into Equation 9 and rearranging, the score function $s(p)$ can then be rewritten as:

$$\sum_{j=1}^{r} \prod_{t \in \mathcal{T}} \left[\mathbf{U}_{\text{LA}}^{(t)} \boldsymbol{\phi}^{(t)} \left(p^{(t)} \right) \right]_j \left[\mathbf{V}_{\text{LA}}^{(t)} \boldsymbol{\psi}^{(t)} \left(p^{(t)} \right) \right]_j . \quad (11)$$

We refer readers to Kolda and Bader (2009) for mathematical details.

For labeled higher-order structures our parameters consist of the set of six matrices, $\{\mathbf{U}_{\text{LA}}^{(t)}\} \cup \{\mathbf{V}_{\text{LA}}^{(t)}\}$. These parameters are shared between second-order and third-order labeled structures. Labeled *second-order* structures are scored as Equation 11, but with the product extending over only the two relevant tasks. Concretely, only four of the representation functions are used rather than all six, along with the four corresponding matrices from $\{\mathbf{U}_{\text{LA}}^{(t)}\} \cup \{\mathbf{V}_{\text{LA}}^{(t)}\}$. *Unlabeled* cross-task structures are scored analogously, reusing the same representations, but with a separate set of parameter matrices $\{\mathbf{U}_{\text{UA}}^{(t)}\} \cup \{\mathbf{V}_{\text{UA}}^{(t)}\}$.

Note that we are not doing tensor factorization; we are learning $\mathbf{U}_{\text{LA}}^{(t)}, \mathbf{V}_{\text{LA}}^{(t)}, \mathbf{U}_{\text{UA}}^{(t)}$, and $\mathbf{V}_{\text{UA}}^{(t)}$ directly, and $\mathcal{W}$ is never explicitly instantiated.

Inference and learning. Given a sentence, we use AD^3 to jointly decode all three formalisms.[7] The training objective used for learning is the sum of the losses for individual tasks.

4.4 Implementation Details

Each input token is mapped to a concatenation of three real vectors: a pre-trained word vector; a randomly-initialized word vector; and a randomly-initialized POS tag vector.[8] All three are updated during training. We use 100-dimensional `GloVe` (Pennington et al., 2014) vectors trained over Wikipedia and Gigaword as pre-trained word embeddings. To deal with out-of-vocabulary words, we apply word dropout (Iyyer et al., 2015) and randomly replace a word w with a special unk-symbol with probability $\frac{\alpha}{1+\#(w)}$, where $\#(w)$ is the count of w in the training set.

Hyperparameter	Value
Pre-trained word embedding dimension	100
Randomly-initialized word embedding dimension	25
POS tag embedding dimension	25
Dimensions of representations ϕ and ψ	100
MLP layers	2
BiLSTM layers	2
BiLSTM dimensions	200
Rank of tensor r	100
α for word dropout	0.25

Table 3: Hyperparameters used in the experiments.

Models are trained for up to 30 epochs with Adam (Kingma and Ba, 2015), with $\beta_1 = \beta_2 = 0.9$, and initial learning rate $\eta_0 = 10^{-3}$. The learning rate η is annealed at a rate of 0.5 every 10 epochs (Dozat and Manning, 2017). We apply early-stopping based on the labeled F_1 score on the development set.[9] We set the maximum number of iterations of AD^3 to 500 and round decisions when it doesn't converge. We clip the ℓ_2 norm of gradients to 1 (Graves, 2013; Sutskever et al., 2014), and we do not use mini-batches. Randomly initialized parameters are sampled from a uniform distribution over $\left[-\sqrt{6/(d_\text{r}+d_\text{c})}, \sqrt{6/(d_\text{r}+d_\text{c})}\right]$, where d_r and d_c are the number of the rows and columns in the matrix, respectively. An ℓ_2 penalty of $\lambda = 10^{-6}$ is applied to all weights. Other hyperparameters are summarized in Table 3.

We use the same pruner as Martins and Almeida (2014), where a first-order feature-rich unlabeled pruning model is trained for each task, and arcs with posterior probability below 10^{-4} are discarded. We further prune labeled structures that appear less than 30 times in the training set. In the development set, about 10% of the arcs remain after pruning, with a recall of around 99%.

[7] Joint inference comes at a cost; our third-order model is able to decode roughly 5.2 sentences (i.e., 15.5 task-specific graphs) per second on a single Xeon E5-2690 2.60GHz CPU.

[8] There are minor differences in the part-of-speech data provided with the three formalisms. For the basic models, we use the POS tags provided with the respective dataset; for the multitask models, we use the (automatic) POS tags provided with DM.

[9] Micro-averaged labeled F_1 for the multitask models.

4.5 Experiments

Experimental settings. We compare four multitask variants to the basic model, as well as the two baseline systems introduced in §3.4.

- SHARED1 is a first-order model. It uses a single shared BiLSTM encoder, and keeps the inference separate for each task.
- FREDA1 is a first-order model based on "frustratingly easy" parameter sharing. It uses a shared encoder as well as task-specific ones. The inference is kept separate for each task.
- SHARED3 is a third-order model. It follows SHARED1 and uses a single shared BiLSTM encoder, but additionally employs cross-task structures and inference.
- FREDA3 is also a third-order model. It combines FREDA1 and SHARED3 by using both "frustratingly easy" parameter sharing and cross-task structures and inference.

In addition, we also examine the effects of syntax by comparing our models to the state-of-the-art open track system (Almeida and Martins, 2015).[10]

Main results overview. Table 4a compares our models to the best published results (labeled F_1 score) on SemEval 2015 Task 18 in-domain test set. Our basic model improves over all closed track entries in all formalisms. It is even with the best open track system for DM and PSD, but improves on PAS and on average, without making use of any syntax. Three of our four multitask variants further improve over our basic model; SHARED1's differences are statistically insignificant. Our best models (SHARED3, FREDA3) outperform the previous state-of-the-art closed track system by 1.7% absolute F_1, and the best open track system by 0.9%, without the use of syntax.

We observe similar trends on the out-of-domain test set (Table 4b), with the exception that, on PSD, our best-performing model's improvement over the open-track system of Almeida and Martins (2015) is not statistically significant.

The extent to which we might benefit from syntactic information remains unclear. With automatically generated syntactic parses, Almeida and Martins (2015) manage to obtain more than 1% absolute improvements over their closed track entry, which is consistent with the extensive evaluation by Zhang et al. (2016), but we leave the incorporation of syntactic trees to future work. Syntactic parsing could be treated as yet another output task, as explored in Lluís et al. (2013) and in the transition-based frameworks of Henderson et al. (2013) and Swayamdipta et al. (2016).

[10] Kanerva et al. (2015) was the winner of the gold track, which overall saw higher performance than the closed and open tracks. Since gold-standard syntactic analyses are not available in most realistic scenarios, we do not include it in this comparison.

	DM	PAS	PSD	Avg.
Du et al., 2015	89.1	91.3	75.7	86.3
A&M, 2015 (closed)	88.2	90.9	76.4	86.0
A&M, 2015 (open)†	89.4	91.7	77.6	87.1
BASIC	89.4	92.2	77.6	87.4
SHARED1	89.7	91.9	77.8	87.4
FREDA1	90.0	92.3	78.1	87.7
SHARED3	90.3	92.5	**78.5**	**88.0**
FREDA3	**90.4**	**92.7**	**78.5**	**88.0**

(a) Labeled F_1 score on the in-domain test set.

	DM	PAS	PSD	Avg.
Du et al., 2015	81.8	87.2	73.3	81.7
A&M, 2015 (closed)	81.8	86.9	74.8	82.0
A&M, 2015 (open)†	83.8	87.6	76.2	83.3
BASIC	84.5	88.3	75.3	83.6
SHARED1	84.4	88.1	75.4	83.5
FREDA1	84.9	88.3	75.8	83.9
SHARED3	**85.3**	88.4	76.1	84.1
FREDA3	**85.3**	**89.0**	**76.4**	**84.4**

(b) Labeled F_1 score on the out-of-domain test set.

Table 4: The last columns show the micro-average over the three tasks. † denotes the use of syntactic parses. Bold font indicates best performance among all systems, and underlines indicate statistical significance with Bonferroni correction compared to A&M, 2015 (open), the strongest baseline system.

Effects of structural overlap. We hypothesized that the overlap between formalisms would enable multitask learning to be effective; in this section we investigate in more detail how structural overlap affected performance. By looking at *undirected* overlap between unlabeled arcs, we discover that modeling only arcs in the same direction may have been a design mistake.

DM and PAS are more structurally similar to each other than either is to PSD. Table 5 compares the structural similarities between the three for-

	Undirected			Directed		
	DM	**PAS**	**PSD**	**DM**	**PAS**	**PSD**
DM	-	67.2	56.8	-	64.2	26.1
PAS	70.0	-	54.9	66.9	-	26.1
PSD	57.4	56.3	-	26.4	29.6	-

Table 5: Pairwise structural similarities between the three formalisms in unlabeled F_1 score. Scores from Oepen et al. (2015).

	DM		**PAS**		**PSD**	
	UF	LF	UF	LF	UF	LF
FREDA1	91.7	90.4	93.1	91.6	89.0	79.8
FREDA3	91.9	90.8	93.4	92.0	88.6	80.4

Table 6: Unlabeled (UF) and labeled (LF) parsing performance of FREDA1 and FREDA3 on the development set of SemEval 2015 Task 18.

malisms in unlabeled F_1 score (each formalism's gold-standard unlabeled graph is used as a prediction of each other formalism's gold-standard unlabeled graph). All three formalisms have more than 50% overlap when ignoring arcs' directions, but considering direction, PSD is clearly different; PSD reverses the direction about half of the time it shares an edge with another formalism. A concrete example can be found in Figure 1, where DM and PAS both have an arc from *"Last"* to *"week,"* while PSD has an arc from *"week"* to *"Last."*

We can compare FREDA3 to FREDA1 to isolate the effect of modeling higher-order structures. Table 6 shows performance on the development data in both unlabeled and labeled F_1. We can see that FREDA3's unlabeled performance improves on DM and PAS, but *degrades* on PSD. This supports our hypothesis, and suggests that in future work, a more careful selection of structures to model might lead to further improvements.

5 Related Work

We note two important strands of related work.

Graph-based parsing. Graph-based parsing was originally invented to handle non-projective syntax (McDonald et al., 2005; Koo et al., 2010; Martins et al., 2013, *inter alia*), but has been adapted to semantic parsing (Flanigan et al., 2014; Martins and Almeida, 2014; Thomson et al., 2014; Kuhlmann, 2014, *inter alia*). Local structure scoring was traditionally done with linear models over hand-engineered features, but lately, various forms of representation learning have been explored to learn feature combinations (Lei et al., 2014; Taub-Tabib et al., 2015; Pei et al., 2015, *inter alia*). Our work is perhaps closest to those who used BiLSTMs to encode inputs (Kiperwasser and Goldberg, 2016; Kuncoro et al., 2016; Wang and Chang, 2016; Dozat and Manning, 2017; Ma and Hovy, 2016).

Multitask learning in NLP. There have been many efforts in NLP to use joint learning to replace pipelines, motivated by concerns about cascading errors. Collobert and Weston (2008) proposed sharing the same word representation while solving multiple NLP tasks. Zhang and Weiss (2016) use a continuous stacking model for POS tagging and parsing. Ammar et al. (2016) and Guo et al. (2016) explored parameter sharing for multilingual parsing. Johansson (2013) and Kshirsagar et al. (2015) applied ideas from domain adaptation to multitask learning. Successes in multitask learning have been enabled by advances in representation learning as well as earlier explorations of parameter sharing (Ando and Zhang, 2005; Blitzer et al., 2006; Daumé III, 2007).

6 Conclusion

We showed two orthogonal ways to apply deep multitask learning to graph-based parsing. The first shares parameters when encoding tokens in the input with recurrent neural networks, and the second introduces interactions between output structures across formalisms. Without using syntactic parsing, these approaches outperform even state-of-the-art semantic dependency parsing systems that use syntax. Because our techniques apply to labeled directed graphs in general, they can easily be extended to incorporate more formalisms, semantic or otherwise. In future work we hope to explore cross-task scoring and inference for tasks where parallel annotations are not available. Our code is open-source and available at `https://github.com/Noahs-ARK/NeurboParser`.

Acknowledgements

We thank the Ark, Maxwell Forbes, Luheng He, Kenton Lee, Julian Michael, and Jin-ge Yao for their helpful comments on an earlier version of this draft, and the anonymous reviewers for their valuable feedback. This work was supported by NSF grant IIS-1562364 and DARPA grant FA8750-12-2-0342 funded under the DEFT program.

References

Mariana S. C. Almeida and André F. T. Martins. 2015. Lisbon: Evaluating TurboSemanticParser on multiple languages and out-of-domain data. In *Proc. of SemEval*.

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

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

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval'07 task 19: Frame semantic structure extraction. In *Proc. of SemEval*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proc. of LAW VII & ID*.

Yoshua Bengio, Rèjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *JMLR* 3:1137–1155.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*.

Carlo E. Bonferroni. 1936. Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R. Istituto Superiore di Scienze Economiche e Commerciali di Firenze* 8:3–62.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. of CoNLL*.

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

Kyunghyun Cho. 2015. Natural language understanding with distributed representation. ArXiv:1511.07916.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proc. of LREC*.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Research on Language & Computation* 3(4):281–332.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proc. of LREC*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proc. of ICLR*.

Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. In *Proc. of SemEval*.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015. Peking: Building semantic dependency graphs with a hybrid parser. In *Proc. of SemEval*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proc. of EMNLP*.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proc. of ACL*.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. DeepBank: A dynamically annotated treebank of the Wall Street Journal. In *Proc. of TLT*.

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

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer.

Alex Graves. 2013. Generating sequences with recurrent neural networks. ArXiv 1308.0850.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2016. A universal framework for inductive transfer parsing across multi-typed treebanks. In *Proc. of COLING*.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English dependency treebank 2.0. In *Proc. LREC*.

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

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proc. of ACL*.

Frank L. Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematical Physics* 6(1):164–189.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL*.

Richard Johansson. 2013. Training parsers on incompatible treebanks. In *Proc. of NAACL*.

Jenna Kanerva, Juhani Luotolahti, and Filip Ginter. 2015. Turku: Semantic dependency parsing as a sequence classification. In *Proc. of SemEval*.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proc. of COLING*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4:313–327.

Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Review* 51(3):455–500.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc. of EMNLP*.

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *Proc. of ACL*.

Marco Kuhlmann. 2014. Linköping: Cubic-time graph parsing with a simple scoring scheme. In *Proc. of SemEval*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proc. of EMNLP*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proc. of EMNLP*.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proc. of ACL*.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *TACL* 1:219–230.

Xuezhe Ma and Eduard Hovy. 2016. Neural probabilistic model for non-projective MST parsing. ArXiv 1701.00874.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proc. of SemEval*.

André F. T. Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

André F. T. Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL*.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proc. of EMNLP*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

Yusuke Miyao. 2006. From linguistic theory to syntactic analysis: Corpus-oriented grammar development and feature forest model.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proc. of ICML*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1):71–106.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proc. of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press.

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

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*.

Vivek Srikumar and Christopher D Manning. 2014. Learning distributed representations for structured output prediction. In *Proc. of NIPS*.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proc. of CoNLL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proc. of CoNLL*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*.

Hillel Taub-Tabib, Yoav Goldberg, and Amir Globerson. 2015. Template kernels for dependency parsing. In *Proc. of NAACL*.

Sam Thomson, Brendan O'Connor, Jeffrey Flanigan, David Bamman, Jesse Dodge, Swabha Swayamdipta, Nathan Schneider, Chris Dyer, and Noah A. Smith. 2014. CMU: Arc-factored, discriminative semantic dependency parsing. In *Proc. of SemEval*.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proc. of ACL*.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering* 11(2):207–238.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proc. of EMNLP*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proc. of ACL*.

Improved Word Representation Learning with Sememes

Yilin Niu[1*], **Ruobing Xie**[1*], **Zhiyuan Liu**[1,2†], **Maosong Sun**[1,2]
[1] Department of Computer Science and Technology,
State Key Lab on Intelligent Technology and Systems,
National Lab for Information Science and Technology, Tsinghua University, Beijing, China
[2] Jiangsu Collaborative Innovation Center for Language Ability,
Jiangsu Normal University, Xuzhou 221009 China

Abstract

Sememes are minimum semantic units of word meanings, and the meaning of each word sense is typically composed by several sememes. Since sememes are not explicit for each word, people manually annotate word sememes and form linguistic common-sense knowledge bases. In this paper, we present that, word sememe information can improve word representation learning (WRL), which maps words into a low-dimensional semantic space and serves as a fundamental step for many NLP tasks. The key idea is to utilize word sememes to capture exact meanings of a word within specific contexts accurately. More specifically, we follow the framework of Skip-gram and present three sememe-encoded models to learn representations of sememes, senses and words, where we apply the attention scheme to detect word senses in various contexts. We conduct experiments on two tasks including word similarity and word analogy, and our models significantly outperform baselines. The results indicate that WRL can benefit from sememes via the attention scheme, and also confirm our models being capable of correctly modeling sememe information.

1 Introduction

Sememes are defined as minimum semantic units of word meanings, and there exists a limited close set of sememes to compose the semantic meanings of an open set of concepts (i.e. word sense). However, sememes are not explicit for each word. Hence, people manually annotate word sememes and build linguistic common-sense knowledge bases.

HowNet (Dong and Dong, 2003) is one of such knowledge bases, which annotates each concept in Chinese with one or more relevant sememes. Different from WordNet (Miller, 1995), the philosophy of HowNet emphasizes the significance of `part` and `attribute` represented by sememes. HowNet has been widely utilized in word similarity computation (Liu and Li, 2002) and sentiment analysis (Xianghua et al., 2013), and in section 3.2 we will give a detailed introduction to sememes, senses and words in HowNet.

In this paper, we aim to incorporate word sememes into word representation learning (WRL) and learn improved word embeddings in a low-dimensional semantic space. WRL is a fundamental and critical step in many NLP tasks such as language modeling (Bengio et al., 2003) and neural machine translation (Sutskever et al., 2014).

There have been a lot of researches for learning word representations, among which word2vec (Mikolov et al., 2013) achieves a nice balance between effectiveness and efficiency. In word2vec, each word corresponds to one single embedding, ignoring the polysemy of most words. To address this issue, (Huang et al., 2012) introduces a multi-prototype model for WRL, conducting unsupervised word sense induction and embeddings according to context clusters. (Chen et al., 2014) further utilizes the `synset` information in WordNet to instruct word sense representation learning.

From these previous studies, we conclude that word sense disambiguation are critical for WRL, and we believe that the sememe annotation of word senses in HowNet can provide necessary semantic regularization for the both tasks. To explore its feasibility, we propose a novel **S**ememe-**E**ncoded **W**ord **R**epresentation **L**earning

[*] indicates equal contribution
[†] Corresponding author: Z. Liu (liuzy@tsinghua.edu.cn)

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2049–2058
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1187

(SE-WRL) model, which detects word senses and learns representations simultaneously. More specifically, this framework regards each word sense as a combination of its sememes, and iteratively performs word sense disambiguation according to their contexts and learn representations of sememes, senses and words by extending Skip-gram in word2vec (Mikolov et al., 2013). In this framework, an attention-based method is proposed to select appropriate word senses according to contexts automatically. To take full advantages of sememes, we propose three different learning and attention strategies for SE-WRL.

In experiments, we evaluate our framework on two tasks including word similarity and word analogy, and further conduct case studies on sememe, sense and word representations. The evaluation results show that our models outperform other baselines significantly, especially on word analogy. This indicates that our models can build better knowledge representations with the help of sememe information, and also implies the potential of our models on word sense disambiguation.

The key contributions of this work are concluded as follows: (1) To the best of our knowledge, this is the first work to utilize sememes in HowNet to improve word representation learning. (2) We successfully apply the attention scheme to detect word senses and learn representations according to contexts with the favor of the sememe annotation in HowNet. (3) We conduct extensive experiments and verify the effectiveness of incorporating word sememes for improved WRL.

2 Related Work

2.1 Word Representation

Recent years have witnessed the great thrive in word representation learning. It is simple and straightforward to represent words using one-hot representations, but it usually struggles with the data sparsity issue and the neglect of semantic relations between words.

To address these issues, (Rumelhart et al., 1988) proposes the idea of distributed representation which projects all words into a continuous low-dimensional semantic space, considering each word as a vector. Distributed word representations are powerful and have been widely utilized in many NLP tasks, including neural language models (Bengio et al., 2003; Mikolov et al., 2010), machine translation (Sutskever et al., 2014; Bahdanau et al., 2015), parsing (Chen and Manning, 2014) and text classification (Zhang et al., 2015). Word distributed representations are capable of encoding semantic meanings in vector space, serving as the fundamental and essential inputs of many NLP tasks.

There are large amounts of efforts devoted to learning better word representations. As the exponential growth of text corpora, model efficiency becomes an important issue. (Mikolov et al., 2013) proposes two models, CBOW and Skip-gram, achieving a good balance between effectiveness and efficiency. These models assume that the meanings of words can be well reflected by their contexts, and learn word representations by maximizing the predictive probabilities between words and their contexts. (Pennington et al., 2014) further utilizes matrix factorization on word affinity matrix to learn word representations. However, these models merely arrange only one vector for each word, regardless of the fact that many words have multiple senses. (Huang et al., 2012; Tian et al., 2014) utilize multi-prototype vector models to learn word representations and build distinct vectors for each word sense. (Neelakantan et al., 2015) presents an extension to Skip-gram model for learning non-parametric multiple embeddings per word. (Rothe and Schütze, 2015) also utilizes an Autoencoder to jointly learn word, sense and synset representations in the same semantic space.

This paper, for the first time, jointly learns representations of sememes, senses and words. The sememe annotation in HowNet provides useful semantic regularization for WRL. Moreover, the unified representations incorporated with sememes also provide us more explicit explanations of both word and sense embeddings.

2.2 Word Sense Disambiguation and Representation Learning

Word sense disambiguation (WSD) aims to identify word senses or meanings in a certain context computationally. There are mainly two approaches for WSD, namely the supervised methods and the knowledge-based methods. Supervised methods usually take the surrounding words or senses as features and use classifiers like SVM for word sense disambiguation (Lee et al., 2004), which are intensively limited to the time-consuming human annotation of training data.

On contrary, knowledge-based methods utilize

large external knowledge resources such as knowledge bases or dictionaries to suggest possible senses for a word. (Banerjee and Pedersen, 2002) exploits the rich hierarchy of semantic relations in WordNet (Miller, 1995) for an adapted dictionary-based WSD algorithm. (Bordes et al., 2011) introduces `synset` information in WordNet to WRL. (Chen et al., 2014) considers synsets in WordNet as different word senses, and jointly conducts word sense disambiguation and word / sense representation learning. (Guo et al., 2014) considers bilingual datasets to learn sense-specific word representations. Moreover, (Jauhar et al., 2015) proposes two approaches to learn sense-specific word representations that are grounded to ontologies. (Pilehvar and Collier, 2016) utilizes personalized PageRank to learn de-conflated semantic representations of words.

In this paper, we follow the knowledge-based approach and automatically detect word senses according to the contexts with the favor of sememe information in HowNet. To the best of our knowledge, this is the first attempt to apply attention-based models to encode sememe information for word representation learning.

3 Methodology

In this section, we present our framework Sememe-Encoded WRL (SE-WRL) that considers sememe information for word sense disambiguation and representation learning. Specifically, we learn our models on a large-scale text corpus with the semantic regularization of the sememe annotation in HowNet and obtain sememe, sense and word embeddings for evaluation tasks.

In the following sections, we first introduce HowNet and the structures of sememes, senses and words. Then we discuss the conventional WRL model Skip-gram that we utilize for the sememe-encoded framework. Finally, we propose three sememe-encoded models in details.

3.1 Sememes, Senses and Words in HowNet

In this section, we first introduce the arrangement of sememes, senses and words in HowNet. HowNet annotates precise senses to each word, and for each sense, HowNet annotates the significance of parts and attributes represented by sememes.

Fig. 1 gives an example of sememes, senses and words in HowNet. The first layer represents the **word** "apple". The word "apple" actually has two main **senses** shown on the second layer: one is a sort of juicy fruit (*apple*), and another is a famous computer brand (*Apple brand*). The third and following layers are those **sememes** explaining each sense. For instance, the first sense *Apple brand* indicates a computer brand, and thus has sememes `computer`, `bring` and `SpeBrand`.

From Fig. 1 we can find that, sememes of many senses in HowNet are annotated with various relations, such as *define* and *modifier*, and form complicated hierarchical structures. In this paper, for simplicity, we only consider all annotated sememes of each sense as a sememe set without considering their internal structure. HowNet assumes the limited annotated sememes can well represent senses and words in the real-world scenario, and thus sememes are expected to be useful for both WSD and WRL.

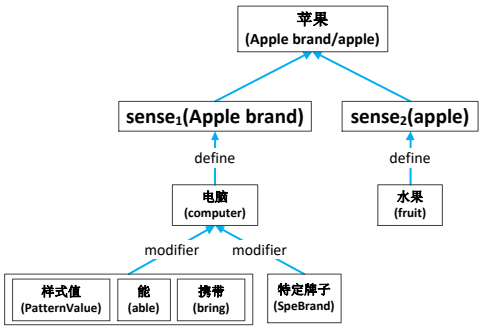


Figure 1: Examples of sememes, senses and words.

We introduce the notions utilized in the following sections as follows. We define the overall sememe, sense and word sets used in training as X, S and W respectively. For each $w \in W$, there are possible multiple senses $s_i^{(w)} \in S^{(w)}$ where $S^{(w)}$ represents the sense set of w. Each sense $s_i^{(w)}$ consists of several sememes $x_j^{(s_i)} \in X_i^{(w)}$. For each target word w in a sequential plain text, $C(w)$ represents its context word set.

3.2 Conventional Skip-gram Model

We directly utilize the widely-used model Skip-gram to implement our SE-WRL model, because Skip-gram has well balanced effectiveness as well as efficiency (Mikolov et al., 2013). The standard skip-gram model assumes that word embeddings should relate to their context words. It aims at

maximizing the predictive probability of context words conditioned on the target word w. Formally, we utilize a sliding window to select the context word set $C(w)$. For a word sequence $H = \{w_1, \cdots, w_n\}$, Skip-gram model intends to maximize:

$$L(H) = \sum_{i=K}^{n-K} \log \Pr(w_{i-K}, \cdots, w_{i+K}|w_i), \quad (1)$$

where K is the size of sliding window. $\Pr(w_{i-K}, \cdots, w_{i+K}|w_i)$ represents the predictive probability of context words conditioned on the target word w_i, formalized by the following softmax function:

$$\begin{aligned}\Pr(w_{i-K}, \cdots, w_{i+K}|w_i) &= \prod_{w_c \in C(w_i)} \Pr(w_c|w_i) \\ &= \prod_{w_c \in C(w_i)} \frac{\exp(\mathbf{w}_c^\top \cdot \mathbf{w}_i)}{\sum_{w'_i \in W} \exp(\mathbf{w}_c^\top \cdot \mathbf{w}'_i)},\end{aligned} \quad (2)$$

in which $\mathbf{w}_c$ and $\mathbf{w}_i$ stand for embeddings of context word $w_c \in C(w_i)$ and target word w_i respectively. We can also follow the strategies of hierarchical softmax and negative sampling proposed in (Mikolov et al., 2013) to accelerate the calculation of softmax.

3.3 SE-WRL Model

In this section, we introduce the SE-WRL models with three different strategies to utilize sememe information, including Simple Sememe Aggregation Model (SSA), Sememe Attention over Context Model (SAC) and Sememe Attention over Target Model (SAT).

3.3.1 Simple Sememe Aggregation Model

The Simple Sememe Aggregation Model (SSA) is a straightforward idea based on Skip-gram model. For each word, SSA considers all sememes in all senses of the word together, and represents the target word using the average of all its sememe embeddings. Formally, we have:

$$\mathbf{w} = \frac{1}{m} \sum_{s_i^{(w)} \in S^{(w)}} \sum_{x_j^{(s_i)} \in X_i^{(w)}} \mathbf{x}_j^{(s_i)}, \quad (3)$$

which means the word embedding of w is composed by the average of all its sememe embeddings. Here, m stands for the overall number of sememes belonging to w.

This model simply follows the assumption that, the semantic meaning of a word is composed of the semantic units, i.e., sememes. As compared to the conventional Skip-gram model, since sememes are shared by multiple words, this model can utilize sememe information to encode latent semantic correlations between words. In this case, similar words that share the same sememes may finally obtain similar representations.

3.3.2 Sememe Attention over Context Model

The SSA Model replaces the target word embedding with the aggregated sememe embeddings to encode sememe information into word representation learning. However, each word in SSA model still has only one single representation in different contexts, which cannot deal with polysemy of most words. It is intuitive that we should construct distinct embeddings for a target word according to specific contexts, with the favor of word sense annotation in HowNet.

To address this issue, we come up with the Sememe Attention over Context Model (SAC). SAC utilizes the attention scheme to automatically select appropriate senses for context words according to the target word. That is, SAC conducts word sense disambiguation for context words to learn better representations of target words. The structure of the SAC model is shown in Fig. 2.

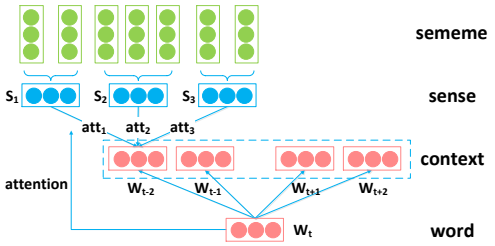


Figure 2: Sememe Attention over Context Model.

More specifically, we utilize the original word embedding for target word w, but use sememe embeddings to represent context word w_c instead of original context word embeddings. Suppose a word typically demonstrates some specific senses in one sentence. Here we employ the target word embedding as an attention to select the most appropriate senses to make up context word embeddings. We formalize the context word embedding

$\mathbf{w}_c$ as follows:

$$\mathbf{w}_c = \sum_{j=1}^{|S^{(w_c)}|} att(s_j^{(w_c)}) \cdot \mathbf{s}_j^{(w_c)}, \quad (4)$$

where $\mathbf{s}_j^{(w_c)}$ stands for the j-th sense embedding of w_c, and $att(s_j^{(w_c)})$ represents the attention score of the j-th sense with respect to the target word w, defined as follows:

$$att(s_j^{(w_c)}) = \frac{\exp(\mathbf{w} \cdot \hat{\mathbf{s}}_j^{(w_c)})}{\sum_{k=1}^{|S^{(w_c)}|} \exp(\mathbf{w} \cdot \hat{\mathbf{s}}_k^{(w_c)})}. \quad (5)$$

Note that, when calculating attention, we use the average of sememe embeddings to represent each sense $s_j^{(w_c)}$:

$$\hat{\mathbf{s}}_j^{(w_c)} = \frac{1}{|X_j^{(w_c)}|} \sum_{k=1}^{|X_j^{(w_c)}|} \mathbf{x}_k^{(s_j)}. \quad (6)$$

The attention strategy assumes that the more relevant a context word sense embedding is to the target word $\mathbf{w}$, the more this sense should be considered when building context word embeddings. With the favor of attention scheme, we can represent each context word as a particular distribution over its sense. This can be regarded as soft WSD. As shown in experiments, it will help learn better word representations.

3.3.3 Sememe Attention over Target Model

The Sememe Attention over Context Model can flexibly select appropriate senses and sememes for context words according to the target word. The process can also be applied to select appropriate senses for the target word, by taking context words as attention. Hence, we propose the Sememe Attention over Target Model (SAT) as shown in Fig. 3.

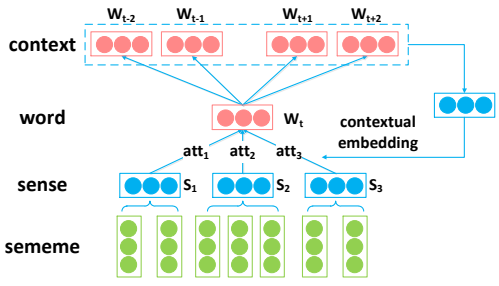


Figure 3: Sememe Attention over Target Model.

Different from SAC model, SAT learns the original word embeddings for context words, but sememe embeddings for target words. We apply context words as attention over multiple senses of the target word w to build the embedding of w, formalized as follows:

$$\mathbf{w} = \sum_{j=1}^{|S^{(w)}|} att(s_j^{(w)}) \cdot \mathbf{s}_j^{(w)}, \quad (7)$$

where $\mathbf{s}_j^{(w)}$ stands for the j-th sense embedding of w, and the context-based attention is defined as follows:

$$att(s_j^{(w)}) = \frac{\exp(\mathbf{w}'_c \cdot \hat{\mathbf{s}}_j^{(w)})}{\sum_{k=1}^{|S^{(w)}|} \exp(\mathbf{w}'_c \cdot \hat{\mathbf{s}}_k^{(w)})}, \quad (8)$$

where, similar to Eq. (6), we also use the average of sememe embeddings to represent each sense $s_j^{(w)}$. Here, $\mathbf{w}'_c$ is the context embedding, consisting of a constrained window of word embeddings in $C(w_i)$. We have:

$$\mathbf{w}'_c = \frac{1}{2K'} \sum_{k=i-K'}^{k=i+K'} \mathbf{w}_k, \quad k \neq i. \quad (9)$$

Note that, since in experiment we find the sense selection of the target word only relies on more limited context words for calculating attention, hence we select a smaller K' as compared to K.

Recall that, SAC only uses one target word as attention to select senses of context words, but SAT uses several context words together as attention to select appropriate senses of target words. Hence SAT is expected to conduct more reliable WSD and result in more accurate word representations, which will be explored in experiments.

4 Experiments

In this section, we evaluate the effectiveness of our SE-WRL[1] models on two tasks including word similarity and word analogy, which are two classical evaluation tasks mainly focusing on evaluating the quality of learned word representations. We also explore the potential of our models in word sense disambiguation with case study, showing the power of our attention-based models.

[1] https://github.com/thunlp/SE-WRL

4.1 Dataset

We use the web pages in Sogou-T[2] as the text corpus to learn WRL models. Sogou-T is provided by a Chinese commercial search engine, which contains 2.7 billion words in total.

We also utilize the sememe annotation in HowNet. The number of distinct sememes used in this paper is $1,889$. The average senses for each word are about 2.4, while the average sememes for each sense are about 1.6. Throughout the Sogou-T corpus, we find that 42.2% of words have multiple senses. This indicates the significance of WSD.

For evaluation, we choose wordsim-240 and wordsim-297[3] to evaluate the performance of word similarity computation. The two datasets both contain frequently-used Chinese word pairs with similarity scores annotated manually. We choose the Chinese Word Analogy dataset proposed by (Chen et al., 2015) to evaluate the performance of word analogy inference, that is, $\mathbf{w}(\text{“king”}) - \mathbf{w}(\text{“man”}) \simeq \mathbf{w}(\text{“queen”}) - \mathbf{w}(\text{“woman”})$.

4.2 Experimental Settings

We evaluate three SE-WRL models including S-SA, SAC and SAT on all tasks. As for baselines, we consider three conventional WRL models including Skip-gram, CBOW and GloVe. For Skip-gram and CBOW, we directly use the code released by Google (Mikolov et al., 2013). GloVe is proposed by (Pennington et al., 2014), which seeks the advantages of the WRL models based on statistics and those based on prediction. Moreover, we propose another model, Maximum Selection over Target Model (MST), for further comparison inspired by (Chen et al., 2014). It represents the current word embeddings with only the most probable sense according to the contexts, instead of viewing a word as a particular distribution over all its senses similar to that of SAT.

For a fair comparison, we train these models with the same experimental settings and with their best parameters. As for the parameter settings, we set the context window size $K = 8$ as the upper bound, and during training, the window size is dynamically selected ranging from 1 to 8 randomly. We set the dimensions of word, sense and sememe embeddings to be the same 200. For learning rate α, its initial value is 0.025 and will descend through iterations. We set the number of negative samples to be 25. We also set a lower bound of word frequency as 50, and in the training set, those words less frequent than this bound will be filtered out. For SAT, we set $K' = 2$.

[2] https://www.sogou.com/labs/resource/t.php

[3] https://github.com/Leonard-Xu/CWE/tree/master/data

4.3 Word Similarity

The task of word similarity aims to evaluate the quality of word representations by comparing the similarity ranks of word pairs computed by WRL models with the ranks given by dataset. WRL models typically compute word similarities according to their distances in the semantic space.

4.3.1 Evaluation Protocol

In experiments, we choose the cosine similarity between two word embeddings to rank word pairs. For evaluation, we compute the Spearman correlation between the ranks of models and the ranks of human judgments.

Model	Wordsim-240	Wordsim-297
CBOW	57.7	61.1
GloVe	59.8	58.7
Skip-gram	58.5	63.3
SSA	58.9	64.0
SAC	59.0	63.1
MST	59.2	62.8
SAT	**63.2**	**65.6**

Table 1: Evaluation results of word similarity computation.

4.3.2 Experiment Results

Table 1 shows the results of these models for word similarity computation. From the results we can observe that:

(1) Our SAT model outperforms other models, including all baselines, on both two test sets. This indicates that, by utilizing sememe annotation properly, our model can better capture the semantic relations of words, and learn more accurate word embeddings.

(2) The SSA model represents a word with the average of its sememe embeddings. In general, S-SA model performs slightly better than baselines, which tentatively proves that sememe information is helpful. The reason is that words which share common sememe embeddings will benefit from each other. Especially, those words with lower frequency, which cannot be learned sufficiently using conventional WRL models, in contrast, can

Model	Accuracy				Mean Rank			
	Capital	City	Relationship	All	Capital	City	Relationship	All
CBOW	49.8	85.7	**86.0**	64.2	36.98	1.23	62.64	37.62
GloVe	57.3	74.3	81.6	65.8	19.09	1.71	3.58	12.63
Skip-gram	66.8	93.7	76.8	73.4	137.19	1.07	2.95	83.51
SSA	62.3	93.7	81.6	71.9	45.74	1.06	3.33	28.52
SAC	61.6	95.4	77.9	70.8	19.08	1.02	**2.18**	12.18
MST	65.7	95.4	82.7	74.5	50.29	1.05	2.48	31.05
SAT	**83.2**	**98.9**	82.4	**85.3**	**14.42**	**1.01**	2.63	**9.48**

Table 2: Evaluation results of word analogy inference.

obtain better word embeddings from SSA simply because their sememe embeddings can be trained sufficiently through other words.

(3) The SAT model performs much better than SSA and SAC. This indicates that SAT can obtain more precise sense distribution of a word. The reason has been mentioned above that, different from SAC using only one target word as attention for WSD, SAT adopts richer contextual information as attention for WSD.

(4) SAT works better than MST, and we can conclude that a soft disambiguation over senses prevents inevitable errors when selecting only one most-probable sense. The result makes sense because, for many words, their various senses are not always entirely different from each other, but share some common elements. In some contexts, a single sense may not convey the exact meaning of this word.

4.4 Word Analogy

Word analogy inference is another widely-used task to evaluate the quality of WRL models (Mikolov et al., 2013).

4.4.1 Evaluation Protocol

The dataset proposed by (Chen et al., 2015) consists of $1,124$ analogies, which contains three analogy types: (1) capitals of countries (Capital), 677 groups; (2) states/provinces of cities (City), 175 groups; (3) family words (Relationship), 272 groups. Given an analogy group of words (w_1, w_2, w_3, w_4), WRL models usually get $\mathbf{w}_2 - \mathbf{w}_1 + \mathbf{w}_3$ equal to $\mathbf{w}_4$. Hence for word analogy inference, we suppose w_4 is missing, and WRL models will rank all candidate words according to their scores as follows:

$$R(w) = \cos(\mathbf{w}_2 - \mathbf{w}_1 + \mathbf{w}_3, \mathbf{w}), \qquad (10)$$

and select the top-ranked word as the answer.

For word analogy inference, we consider two evaluation metrics: (1) **Accuracy**. For each analogy group, a WRL model selects the top-ranked word $w = \arg\max_w R(w)$, which is judged as positive if $w = w_4$. The percentage of positive samples is regarded as the accuracy score for this WRL model. (2) **Mean Rank**. For each analogy group, a WRL model will assign a rank for the gold standard word w_4 according to the scores computed by Eq. (10). We use the mean rank of all gold standard words as the evaluation metric.

4.4.2 Experiment Results

Table 2 shows the evaluation results of these models for word analogy inference. From the table, we can observe that:

(1) The SAT model performs best among all models, and the superiority is more significant than that on word similarity computation. This indicates that SAT will enhance the modeling of implicit relations between word embeddings in the semantic space. The reason is that sememes annotated to word senses have encoded these word relations. For example, `capital` and `Cuba` are two sememes of the word "Havana", which provide explicit semantic relations between the words "Cuba" and "Havana".

(2) The SAT model does well on both classes of Capital and City, because some words in these classes have low frequencies, while their sememes occur so many times that sememe embeddings can be learned sufficiently. With these sememe embeddings, these low-frequent words can be learned more efficiently by SAT.

(3) It seems that CBOW works better than SAT on Relationship class. Whereas for the mean rank, CBOW gets the worst results, which indicates the performance of CBOW is unstable. On the contrary, although the accuracy of SAT is a bit lower than that of CBOW, SAT seldom gives an outrageous prediction. In most wrong cas-

Word: 苹果(“Apple brand/apple”) sense1: *Apple brand* (computer, PatternValue, able, bring, SpeBrand) sense2: *duct* (fruit)		
苹果 素有果中王美称（**Apple** is always famous as the king of fruits） **苹果** 电脑无法正常启动（The **Apple brand** computer can not startup normally）	*Apple brand*: 0.28 *Apple brand*: 0.87	*apple*: 0.72 *apple*: 0.13
Word: 扩散(“proliferate/metastasize”) sense1: *proliferate* (disperse) sense2: *metastasize* (disperse, disease)		
防止疫情**扩散**（Prevent epidemic from **metastasizing**） 不**扩散** 核武器条约（Treaty on the Non-**Proliferation** of Nuclear Weapons）	*proliferate*: 0.06 *proliferate*: 0.68	*metastasize*: 0.94 *metastasize*: 0.32
Word: 队伍(“contingent/troops”) sense1: *contingent* (community) sense2: *troops* (army)		
八支**队伍** 进入第二阶段团体赛（Eight **contingents** enter the second stage of team competition）	*contingent*: 0.90	*troops*: 0.10
公安基层**队伍** 组织建设（Construct the organization of public security’s **troops** in grass-roots unit）	*contingent*: 0.15	*troops*: 0.85

Table 3: Examples of sememes, senses and words in context with attention.

es, SAT predicts the word “grandfather” instead of “grandmother”, which is not completely nonsense, because in HowNet the words “grandmother”, “grandfather”, “grandma” and some other similar words share four common sememes while only one sememe of them are different. These similar sememes make the attention process less discriminative with each other. But for the wrong cases of CBOW, we find that many mistakes are about words with low frequencies, such as “stepdaughter” which occurs merely for 358 times. Considering sememes may relieve this problem.

4.5 Case study

The above experiments verify the effectiveness of our models for WRL. Here we show some examples of sememes, senses and words for case study.

4.5.1 Word Sense Disambiguation

To demonstrate the validity of Sememe Attention, we select three attention results in training set, as shown in Table 3. In this table, the first rows of three examples are word-sense-sememe structures of each word. For instance, in the third example, the word has two senses, *contingent* and *troops*; *contingent* has one sememe `community`, while *troops* has one sememe `army`. The three examples all indicate that our models can estimate appropriate distributions of senses for a word given a context.

4.5.2 Effect of Context Words for Attention

We demonstrate the effect of context words for attention in Table. 4. The word “Havana” consists of four sememes, among which two sememes `capital` and `Cuba` describe distinct attributes of the word from different aspects.

Word Sememe	哈瓦那(“Havana”) 国都(`capital`)	 古巴(`Cuba`)
古巴(“Cuba”)	0.39	0.42
俄罗斯(“Russia”)	0.39	-0.09
雪茄(“cigar”)	0.00	0.36

Table 4: Sememe weight for computing attention.

Here, we list three different context words “Cuba”, “Russia” and “cigar”. Given the context word “Cuba”, both sememes get high weights, indicating their contributions to the meaning of “Havana” in this context. The context word “Russia” is more relevant to the sememe `capital`. When the context word is “cigar”, the sememe `Cuba` has more influence, because cigar is a famous specialty of Cuba. From these examples, we can conclude that our Sememe Attention can accurately capture the word meanings in complicated contexts.

5 Conclusion and Future Work

In this paper, we propose a novel method to model sememe information for learning better word representations. Specifically, we utilize sememe information to represent various senses of each word and propose Sememe Attention to select appropriate senses in contexts automatically. We evaluate our models on word similarity and word analogy, and results show the advantages of our Sememe-Encoded WRL models. We also analyze several cases in WSD and WRL, which confirms our models are capable of selecting appropriate word senses with the favor of sememe attention.

We will explore the following research directions in future: (1) The sememe information in HowNet is annotated with hierarchical structure

and relations, which have not been considered in our framework. We will explore to utilize these annotations for better WRL. (2) We believe the idea of sememes is universal and could be well-functioned beyond languages. We will explore the effectiveness of sememe information for WRL in other languages.

Acknowledgments

This work is supported by the 973 Program (No. 2014CB340501), the National Natural Science Foundation of China (NSFC No. 61572273, 61661146007), and the Key Technologies Research and Development Program of China (No. 2014BAK04B03). This work is also funded by the Natural Science Foundation of China (NSFC) and the German Research Foundation (DFG) in Project Crossmodal Learning, NSFC 61621136008 / DFC TRR-169.

References

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

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Proceedings of CICLing*. pages 136–145.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JMLR* 3:1137–1155.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*.

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

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*. pages 1025–1035.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *Proceedings of IJCAI*. pages 1236–1242.

Zhendong Dong and Qiang Dong. 2003. Hownet-a hybrid language and knowledge resource. In *Proceedings of NLP-KE*. IEEE, pages 820–824.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *Proceedings of COLING*. pages 497–507.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*. pages 873–882.

Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of NAACL*. volume 1.

Yoong Keok Lee, Hwee Tou Ng, and Tee Kiah Chia. 2004. Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In *Proceedings of SENSEVAL-3*. pages 137–140.

Qun Liu and Sujian Li. 2002. Word similarity computing based on how-net. *CLCLP* 7(2):59–76.

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

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*. volume 14, pages 1532–43.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. In *Proceedings of EMNLP*.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *Proceedings of ACL* .

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5(3):1.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*. pages 3104–3112.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING*. pages 151–160.

Fu Xianghua, Liu Guo, Guo Yanyan, and Wang Zhiqiang. 2013. Multi-aspect sentiment analysis for chinese online social reviews based on topic modeling and hownet lexicon. *Knowledge-Based Systems* 37:186–195.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of NIPS*. pages 649–657.

Learning Character-level Compositionality with Visual Features

Frederick Liu[1], **Han Lu**[1], **Chieh Lo**[2], **Graham Neubig**[1]

[1]Language Technology Institute
[2]Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213
{fliu1,hlu2,gneubig}@cs.cmu.edu
chiehl@andrew.cmu.edu

Abstract

Previous work has modeled the compositionality of words by creating character-level models of meaning, reducing problems of sparsity for rare words. However, in many writing systems compositionality has an effect even on the character-level: the meaning of a character is derived by the sum of its parts. In this paper, we model this effect by creating embeddings for characters based on their visual characteristics, creating an image for the character and running it through a convolutional neural network to produce a visual character embedding. Experiments on a text classification task demonstrate that such model allows for better processing of instances with rare characters in languages such as Chinese, Japanese, and Korean. Additionally, qualitative analyses demonstrate that our proposed model learns to focus on the parts of characters that carry semantic content, resulting in embeddings that are coherent in visual space.

1 Introduction

Compositionality—the fact that the meaning of a complex expression is determined by its structure and the meanings of its constituents—is a hallmark of every natural language (Frege and Austin, 1980; Szabó, 2010). Recently, neural models have provided a powerful tool for learning how to compose words together into a meaning representation of whole sentences for many downstream tasks. This is done using models of various levels of sophistication, from simpler bag-of-words (Iyyer et al., 2015) and linear recurrent neural network (RNN) models (Sutskever et al., 2014; Kiros et al., 2015), to more sophisticated models using tree-structured (Socher et al., 2013) or convolutional networks (Kalchbrenner et al., 2014).

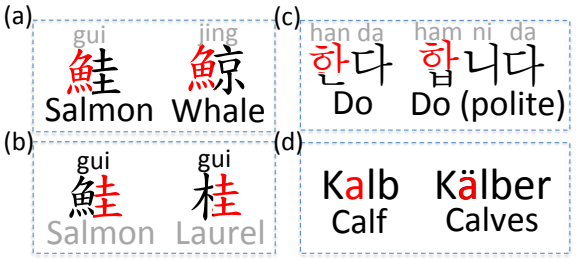


Figure 1: Examples of character-level compositionality in (a, b) Chinese, (c) Korean, and (d) German. The red part of the characters are shared, and affects the pronunciation (top) or meaning (bottom).

In fact, a growing body of evidence shows that it is essential to look below the word-level and consider compositionality within words themselves. For example, several works have proposed models that represent words by composing together the characters into a representation of the word itself (Ling et al., 2015; Zhang et al., 2015; Dhingra et al., 2016). Additionally, for languages with productive word formation (such as agglutination and compounding), models calculating morphology-sensitive word representations have been found effective (Luong et al., 2013; Botha and Blunsom, 2014). These models help to learn more robust representations for *rare words* by exploiting morphological patterns, as opposed to models that operate purely on the lexical level as the atomic units.

For many languages, compositionality stops at the character-level: characters are atomic units of meaning or pronunciation in the language, and no further decomposition can be done.[1] However, for other languages, character-level compositionality, where a character's meaning or pronunciation can

[1]In English, for example, this is largely the case.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2059–2068
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1188

Lang	Geography	Sports	Arts	Military	Economics	Transportation
Chinese	32.4k	49.8k	50.4k	3.6k	82.5k	40.4k
Japanese	18.6k	82.7k	84.1k	81.6k	80.9k	91.8k
Korean	6k	580	5.74k	840	5.78k	1.68k
Lang	**Medical**	**Education**	**Food**	**Religion**	**Agriculture**	**Electronics**
Chinese	30.3k	66.2k	554	66.9k	89.5k	80.5k
Japanese	66.5k	86.7k	20.2k	98.1k	97.4k	1.08k
Korean	16.1k	4.71k	33	2.60k	1.51k	1.03k

Table 1: By-category statistics for the Wikipedia dataset. Note that Food is the abbreviation for "Food and Culture" and Religion is the abbreviation for "Religion and Belief".

be derived from the sum of its parts, is very much a reality. Perhaps the most compelling example of compositionality of sub-character units can be found in logographic writing systems such as the Han and Kanji characters used in Chinese and Japanese, respectively.[2] As shown on the left side of Fig. 1, each part of a Chinese character (called a "radical") potentially contributes to the meaning (i.e., Fig. 1(a)) or pronunciation (i.e., Fig. 1(b)) of the overall character. This is similar to how English characters combine into the meaning or pronunciation of an English word. Even in languages with phonemic orthographies, where each character corresponds to a pronunciation instead of a meaning, there are cases where composition occurs. Fig. 1(c) and (d) show the examples of Korean and German, respectively, where morphological inflection can cause single characters to make changes where some but not all of the component parts are shared.

In this paper, we investigate the feasibility of modeling the *compositionality of characters* in a way similar to how humans do: by visually observing the character and using the features of its shape to learn a representation encoding its meaning. Our method is relatively simple, and generalizable to a wide variety of languages: we first transform each character from its Unicode representation to a rendering of its shape as an image, then calculate a representation of the image using Convolutional Neural Networks (CNNs) (Cun et al., 1990). These features then serve as inputs to a down-stream processing task and trained in an end-to-end manner, which first calculates a loss function, then back-propagates the loss back to the CNN.

[2]Other prominent examples are largely for extinct languages: Egyptian hieroglyphics, Mayan glyphs, and Sumerian cuneiform scripts (Daniels and Bright, 1996).

As demonstrated by our motivating examples in Fig. 1, in logographic languages character-level semantic or phonetic similarity is often indicated by visual cues; we conjecture that CNNs can appropriately model these visual patterns. Consequently, characters with similar visual appearances will be biased to have similar embeddings, allowing our model to handle *rare characters* effectively, just as character-level models have been effective for rare words.

To evaluate our model's ability to learn representations, particularly for rare characters, we perform experiments on a downstream task of classifying Wikipedia titles for three Asian languages: Chinese, Japanese, and Korean. We show that our proposed framework outperforms a baseline model that uses standard character embeddings for instances containing rare characters. A qualitative analysis of the characteristics of the learned embeddings of our model demonstrates that visually similar characters share similar embeddings. We also show that the learned representations are particularly effective under low-resource scenarios and *complementary* with standard character embeddings; combining the two representations through three different fusion methods (Snoek et al., 2005; Karpathy et al., 2014) leads to consistent improvements over the strongest baseline without visual features.

2 Dataset

Before delving into the details of our model, we first describe a dataset we constructed to examine the ability of our model to capture the compositional characteristics of characters. Specifically, the dataset must satisfy two desiderata: (1) it must be necessary to fully utilize each character in the input in order to achieve high accuracy, and (2) there must be enough regularity and com-

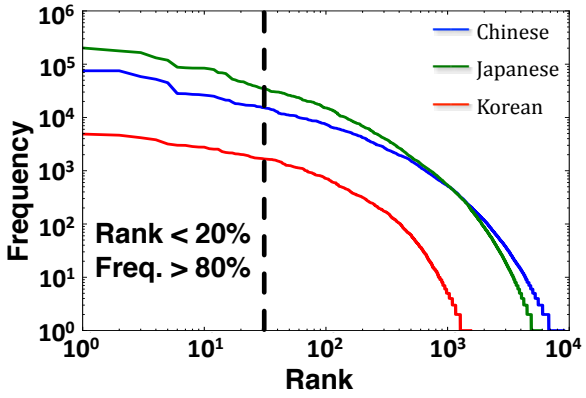


Figure 2: The character rank-frequency distribution of the corpora we considered in this paper. All three languages have a long-tail distribution.

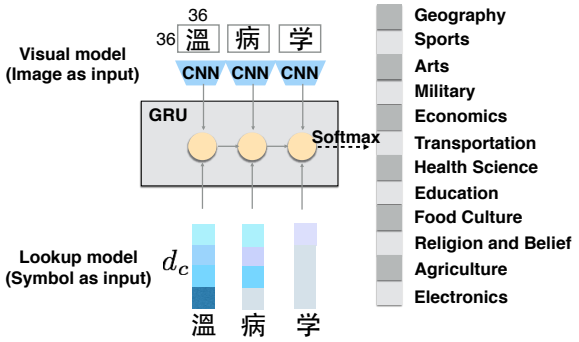


Figure 3: An illustration of two models, our proposed VISUAL model at the top and the baseline LOOKUP model at the bottom using the same RNN architecture. A string of characters (e.g. "温病学"), each converted into a 36x36 image, serves as input of our VISUAL model. d_c is the dimension of the character embedding for the LOOKUP model.

positionality in the characters of the language. To satisfy these desiderata, we create a text classification dataset where the input is a Wikipedia article title in Chinese, Japanese, or Korean, and the output is the category to which the article belongs.[3] This satisfies (1), because Wikipedia titles are short and thus each character in the title will be important to our decision about its category. It also satisfies (2), because Chinese, Japanese, and Korean have writing systems with large numbers of characters that decompose regularly as shown in Fig. 1. While this task in itself is novel, it is similar to previous work in named entity type inference using Wikipedia (Toral and Munoz, 2006; Kazama and Torisawa, 2007; Ratinov and Roth, 2009), which has proven useful for downstream named entity recognition systems.

2.1 Dataset Collection

As the labels we would like to predict, we use 12 different main categories from the Wikipedia web page: Geography, Sports, Arts, Military, Economics, Transportation, Health Science, Education, Food Culture, Religion and Belief, Agriculture and Electronics. Wikipedia has a hierarchical structure, where each of these main categories has a number of subcategories, and each subcategory has its own subcategories, etc. We traverse this hierarchical structure, adding each main category tag to all of its descendants in this subcategory tree structure. In the case that a particular article is the descendant of multiple main categories, we favor the main category that minimizes the depth of the article in the tree (e.g., if an article is two steps away from Sports and three steps away from Arts, it will receive the "Sports" label). We also perform some rudimentary filtering, removing pages that match the regular expression ".*:.*", which catches special pages such as "title:agriculture".

[3]The link to the dataset and the crawling scripts – https://github.com/frederick0329/Wikipedia_title_dataset

2.2 Statistics

For Chinese, Japanese, and Korean, respectively, the number of articles is 593k/810k/46.6k, and the average length and standard deviation of the title is 6.25±3.96/8.60±5.58/6.10±3.71. As shown in Fig. 2, the character rank-frequency distributions of all three languages follows the 80/20 rule (Newman, 2005) (i.e., top 20% ranked characters that appear more than 80% of total frequencies), demonstrating that the characters in these languages belong to a long tail distribution.

We further split the dataset into training, validation, and testing sets with a 6:2:2 ratio. The category distribution for each language can be seen in Tab. 1. Chinese has two varieties of characters, traditional and simplified, and the dataset is a mix of the two. Hence, we transform this dataset into two separate sets, one completely simplified and the other completely traditional using the Chinese text converter provided with Mac OS.

3 Model

Our overall model for the classification task follows the encoder model by Sutskever et al. (2014).

Layer#	3-layer CNN Configuration
1	Spatial Convolution (3, 3) $\rightarrow$ 32
2	ReLu
3	MaxPool (2, 2)
4	Spatial Convolution (3, 3) $\rightarrow$ 32
5	ReLu
6	MaxPool (2, 2)
7	Spatial Convolution (3, 3) $\rightarrow$ 32
8	ReLu
9	Linear (800, 128)
10	ReLu
11	Linear (128, 128)
12	ReLu

Table 2: Architecture of the CNN used in the experiments. All the convolutional layers have 32 3×3 filters.

We calculate character representations, use a RNN to combine the character representations into a sentence representation, and then add a softmax layer after that to predict the probability for each class. As shown in Fig. 2.1, the baseline model, which we call it the LOOKUP model, calculates the representation for each character by looking it up in a character embedding matrix. Our proposed model, the VISUAL model instead learns the representation of each character from its visual appearance via CNN.

LOOKUP model Given a character vocabulary C, for the LOOKUP model as in the bottom part of Fig. 2.1, the input to the network is a stream of characters $c_1, c_2, ...c_N$, where $c_n \in C$. Each character is represented by a 1-of-$|C|$ (one-hot) encoding. This one-hot vector is then multiplied by the lookup matrix $T_C \in \mathbf{R}^{|C| \times d_c}$, where d_c is the dimension of the character embedding. The randomly initialized character embeddings were optimized with classification loss.

VISUAL model The proposed method aims to learn a representation that includes image information, allowing for better parameter sharing among characters, particularly characters that are less common. Different from the LOOKUP model, each character is first transformed into a 36-by-36 image based on its Unicode encoding as shown in the upper part of Fig 2.1. We then pass the image through a CNN to get the embedding for the image. The parameters for the CNN are learned through backpropagation from the classification loss. Because we are training embeddings based on this classification loss, we expect that the CNN will focus on parts of the image that contain semantic information useful for category classification, a hypothesis that we examine in the experiments (see Section 5.5).

In more detail, the specific structure of the CNN that we utilize consists of three convolution layers where each convolution layer is followed by the max pooling and ReLU nonlinear activation layers. The configurations of each layer are listed in Tab. 2. The output vector for the image embeddings also has size d_c which is the same as the LOOKUP model.

Encoder and Classifier For both the LOOKUP and the VISUAL models, we adopt an RNN encoder using Gated Recurrent Units (GRUs) (Chung et al., 2014). Each of the GRU units processes the character embeddings sequentially. At the end of the sequence, the incremental GRU computation results in a hidden state e embedding the sentence. The encoded sentence embedding is passed through a linear layer whose output is the same size as the number of classes. We use a softmax layer to compute the posterior class probabilities:

$$P(y = j|e) = \frac{\exp(w_j^T e + b_j)}{\sum_{i=1}^{L} \exp(w_i^T e + b_i)} \quad (1)$$

To train the model, we use cross-entropy loss between predicted and true targets:

$$J = \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{L} -t_{i,j} \log(p_{i,j}) \quad (2)$$

where $t_{i,j} \in \{0, 1\}$ represents the ground truth label of the j-th class in the i-th Wikipedia page title. B is the batch size and L is the number of categories.

4 Fusion-based Models

One thing to note is that the LOOKUP and the VISUAL models have their own advantages. The LOOKUP model learns embedding that captures the semantics of each character symbol without sharing information with each other. In contrast, the proposed VISUAL model directly learns embedding from visual information, which naturally shares information between visually similar characters. This characteristic gives the VISUAL

Lookup/Visual	100%	50%	12.5%
zh_trad	0.55/0.54	0.53/0.50	0.48/0.47
zh_simp	0.55/0.54	0.53/0.52	0.48/0.46
ja	0.42/0.39	0.47/0.45	0.44/0.41
ko	0.47/0.42	0.44/0.39	0.37/0.36

Table 3: The classification results of the LOOKUP / VISUAL models for different percentages of full training size.

model the ability to generalize better to rare characters, but also has the potential disadvantage of introducing noise for characters with similar appearances but different meanings.

With the complementary nature of these two models in mind, we further combine the two embeddings to achieve better performances. We adopt three fusion schemes, early fusion, late fusion (described by Snoek et al. (2005) and Karpathy et al. (2014)), and fallback fusion, a method specific to this paper.

Early Fusion Early fusion works by concatenating the two varieties of embeddings before feeding them into the RNN. In order to ensure that the dimensions of the RNN are the same after concatenation, the concatenated vector is fed through a hidden layer to reduce the size from $2 \times d_c$ to d_c. The whole model is then fine-tuned with training data.

Late Fusion Instead of learning a joint representation like early fusion, late fusion averages the model predictions. Specifically, it takes the output of the softmax layers from both models and averages the probabilities to create a final distribution used to make the prediction.

Fallback Fusion Our final fallback fusion method hypothesizes that our VISUAL model does better with instances which contain more rare characters. First, in order to quantify the overall rareness of an instance consisting of multiple characters, we calculate the average training set frequency of the characters therein. The fallback fusion method uses the VISUAL model to predict testing instances with average character frequency below or equal to a threshold (here we use 0.0 frequency as cutoff, which means all characters in the instance do not appear in the training set), and uses the LOOKUP model to predict the rest of the instances.

5 Experiments and Results

In this section, we compare our proposed VISUAL model with the baseline LOOKUP model through three different sets of experiments. First, we examine whether our model is capable of classifying text and achieving similar performance as the baseline model. Next, we examine the hypothesis that our model will outperform the baseline model when dealing with low frequency characters. Finally, we examine the fusion methods described in Section 4.

5.1 Experimental Configurations

The dimension of the embeddings and batch size for both models are set to $d_c = 128$ and $B = 400$, respectively. We build our proposed model using Torch (Collobert et al., 2002), and use Adam (Kingma and Ba, 2014) with a learning rate $\eta = 0.001$ for stochastic optimization. The length of each instance is cut off or padded to 10 characters for batch training.

5.2 Comparison with the Baseline Model

In this experiment, we examine whether our VISUAL model achieves similar performance with the baseline LOOKUP model in classification accuracy.

The results in Tab. 3 show that the baseline model performs 1-2% better across four datasets; this is due to the fact that the LOOKUP model can directly learn character embeddings that capture the semantics of each character symbol for frequent characters. In contrast, the VISUAL model learns embeddings from visual information, which constraints characters that has similar appearance to have similar embeddings. This is an advantage for rare characters, but a disadvantage for high frequency characters because being similar in appearance does not always lead to similar semantics.

To demonstrate that this is in fact the case, besides looking at the overall classification accuracy, we also examine the performance on classifying low frequency instances which are sorted according to the average training set frequency of the characters therein. Tab. 4 and Fig. 4 both show that our model performs better in the 100 lowest frequency instances (the intersection point of the two models). More specifically, take Fig. 4(a)' as example, the solid (proposed) line is higher than the dashed (baseline) line up to 10^2, indicating that the proposed model outperforms the baseline for the

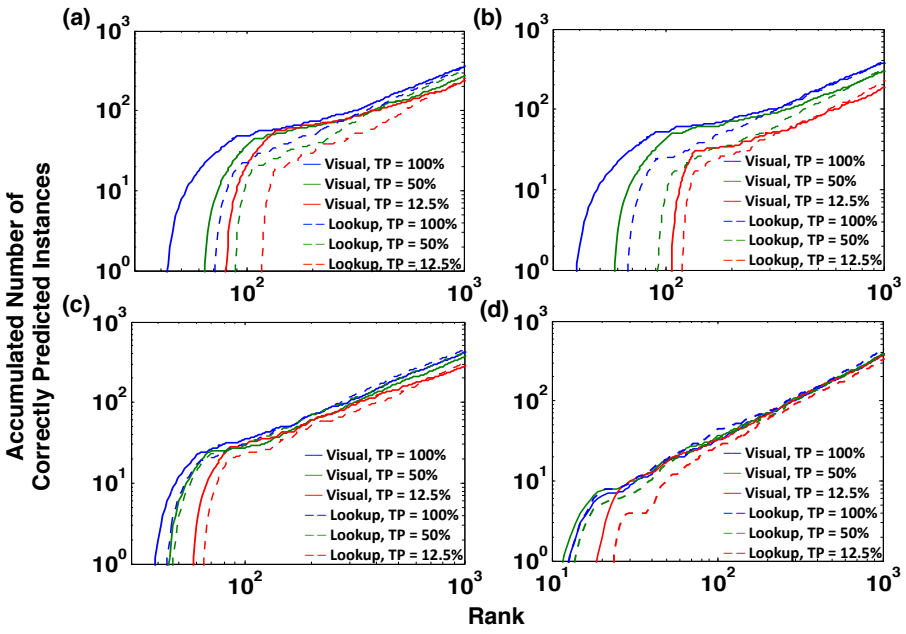


Figure 4: Experiments on different training sizes for four different datasets. More specifically, we consider three different training data size percentages (TPs) (100%, 50%, and 12.5%) and four datasets: (a) traditional Chinese, (b) simplified Chinese, (c) Japanese, and (d) Korean. We calculate the accumulated number of correctly predicted instances for the VISUAL model (solid lines) and the LOOKUP model (dashed lines). This figure is a log-log plot, where x-axis shows rarity (rarest to the left), y-axis shows cumulative correctly classified instances up to this rank; a perfect classifier will result in a diagonal line.

first 100 instances. Lines depart the x-axis when the model classifies its first instance correctly, and the LOOKUP model did not correctly classify any of the first 80 rarest instances, resulting in it crossing later than the proposed model. This confirms that the VISUAL model can share visual information among characters and help to classify low frequency instances.

For training time, visual features take significantly more time, as expected. VISUAL is 30x slower than LOOKUP, although they are equivalent at test time. For space, images of Chinese characters took 36MB to store for 8985 characters.

5.3 Experiments on Different Training Sizes

In our second experiment, we consider two smaller training sizes (i.e., 50% and 12.5% of the full training size) indicated by green and red lines in Fig. 4. We performed this experiment under the hypothesis that because the proposed method was more robust to infrequent characters, the proposed model may perform better in low-resourced scenarios. If this is the case, the intersection point of the two models will shift right because of the increase of the number of instances with low average character frequency.

Lookup/Visual	100	1000	10000
zh_trad	0.22/**0.49**	0.35/0.35	**0.40**/0.39
zh_simp	0.25/**0.53**	**0.39**/0.37	**0.41**/0.40
ja	0.30/**0.35**	**0.45**/0.41	**0.44**/0.41
ko	**0.44**/0.33	**0.44**/0.33	**0.48**/0.42

Table 4: Classification results for the LOOKUP / VISUAL of the k lowest frequency instances across four datasets. The 100 lowest frequency instances for traditional and simplified Chinese and Korean were both significant (p-value < 0.05). Those for Japanese were not (p-value = 0.13); likely because there was less variety than Chinese and more data than Korean.

As we can see in Fig. 4, the intersection point for 100% training data lies between the intersection point for 50% training data and 12.5%. This disagrees with our hypothesis; this is likely because while the number of low-frequency characters increases, smaller amounts of data also adversely impact the ability of CNN to learn useful visual features, and thus there is not a clear gain nor loss when using the proposed method.

As a more extreme test of the ability of our proposed framework to deal with the unseen char-

	zh_trad	zh_simp	ja	ko
Lookup	0.5503	0.5543	0.4914	0.4765
Visual	0.5434	0.5403	0.4775	0.4207
early	0.5520	0.5546	0.4896	0.4796
late	**0.5658**	**0.5685**	**0.5029**	**0.4869**
fall	0.5507	0.5547	0.4914	0.4766

Table 5: Experiment results for three different fusion methods across 4 datasets. The late fusion model was better (p-value < 0.001) across four datasets.

acters in the test set, we use traditional Chinese as our training data and simplified Chinese as our testing data. The model was able to achieve around 40% classification accuracy when we use the full training set, compared to 55%, which is achieved by the model trained on simplified Chinese. This result demonstrates that the model is able to transfer between similar scripts, similarly to how most Chinese speakers can guess the meaning of the text, even if it is written in the other script.

5.4 Experiment on Different Fusion Methods

Results of different fusion methods can be found in Tab. 5. The results show that late fusion gives the best performance among all the fusion schemes combining the LOOKUP model and the proposed VISUAL model. Early fusion achieves small improvements for all languages except Japanese, where it displays a slight drop. Unsurprisingly, fallback fusion performs better than the LOOKUP model and the VISUAL model alone, since it directly targets the weakness of the LOOKUP model (e.g., rare characters) and replaces the results with the VISUAL model. These results show that simple integration, no matter which schemes we use, is beneficial, demonstrating that both methods are capturing complementary information.

5.5 Visualization of Character Embeddings

Finally, we qualitatively examine what is learned by our proposed model in two ways. First, we visualize which parts of the image are most important to the VISUAL model's embedding calculation. Second, we show the 6-nearest neighbor results for characters using both the LOOKUP and the VISUAL embeddings.

Figure 5: Examples of how much each part of the character contributes to its embedding (the darker the more). Two characters are shown per radical to emphasize that characters with same radical have similar patterns.

Emphasis of the VISUAL Model In order to delve deeper into what the VISUAL model has learned, we measure a modified version of the occlusion sensitivity proposed by Zeiler and Fergus (2014) by masking the original character image in four ways, and examine the importance of each part of the character to the model's calculated representations. Specifically, we leave only the upper half, bottom half, left half, or right half of the image, and mask the remainder with white pixels since Chinese characters are usually formed by combining two radicals vertically or horizontally. We run these four images forward through the CNN part of the model and calculate the L_2 distance between the masked image embeddings with the full image embedding. The larger the distance, the more the masked part of the character contributes to the original embedding. The contribution of each part (e.g. the L_2 distance) is represented as a heat map, and then it is normalized to adjust the opacity of the character strokes for better visualization. The value of each corner of the heatmap is calculated by adding the two L_2 distances that contribute to this corner.

The visualization is shown in Fig. 5. The meaning of each Chinese character in English is shown below the Chinese character. The opacity of the character strokes represent how much the corresponding parts contribute to the original embedding (the darker the more). In general, the darker part of the character is related to its semantics. For example, "金" means gold in Chinese, which is

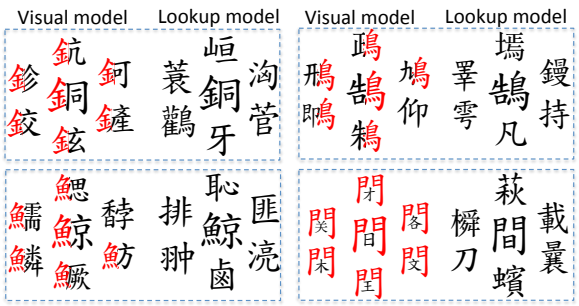


Figure 6: Visualization of the Chinese traditional characters by finding the 6-nearest neighbors of the query (i.e., center) characters. The highlighted red indicates the radical along with the meaning of the characters.

highlighted in both "鐵" (Iron) and "銅" (Bronze). We can also find similar results for other examples shown in Fig. 5. Fig. 5 also demonstrated that our model captures the compositionality of Chinese characters, both meaning of sub-character units and their structure (e.g. the semantic content tends to be structurally localized on one side of a Chinese character).

K-nearest neighbors Finally, to illustrate the difference of the learned embeddings between the two models, we display 6-nearest neighbors (L_2 distance) for selected characters in Fig. 6. As can be seen, the VISUAL embedding for characters with similar appearances are close to each other. In addition, similarity in the radical part indicates semantic similarity between the characters. For example, the characters with radical "鳥" all refer to different type of birds.

The LOOKUP embedding do not show such feature, as it learns the embedding individually for each symbol and relies heavily on the training set and the task. In fact, the characters shown in Fig. 6 for the LOOKUP model do not exhibit semantic similarity either. There are two potential explanations for this: First, the category classification task that we utilized do not rely heavily on the fine-grained semantics of each character, and thus the LOOKUP model was able to perform well without exactly capturing the semantics of each character precisely. Second, the Wikipedia dataset contains a large number of names and location and the characters therein might not have the same semantic meaning used in daily vocabulary.

6 Related Work

Methods that utilize neural networks to learn distributed representations of words or characters have been widely developed. However, word2vec (Mikolov et al., 2013), for example, requires storing an extremely large table of vectors for all word types. For example, due to the size of word types in twitter tweets, work has been done to generate vector representations of tweets at character-level (Dhingra et al., 2016).

There is also work done in understanding mathematical expressions with a convolutional network for text and layout recognition by using an attention-based neural machine translation system (Deng et al., 2016). They tested on real-world rendered mathematical expressions paired with LaTeX markup and show the system is effective at generating accurate markup. Other than that, there are several works that combine visual information with text in improving machine translation (Sutskever et al., 2014), visual question answering, caption generation (Xu et al., 2015), etc. These works extract image representations from a pre-trained CNN (Zhu et al., 2016; Wang et al., 2016).

Unrelated to images, CNNs have also been used for text classification (Kim, 2014; Zhang et al., 2015). These models look at the sequential dependencies at the word or character-level and achieve the state-of-the-art results. These works inspire us to use CNN to extract features from image and serve as the input to the RNN. Our model is able to directly back-propagate the gradient all the way through the CNN, which generates visual embeddings, in a way such that the embedding can contain both semantic and visual information.

Several techniques for reducing the rare words effects have been introduced in the literature, including spelling expansion (Habash, 2008), dictionary term expansion (Habash, 2008), proper name transliteration (Daumé and Jagarlamudi, 2011), treating words as a sequence of characters (Luong and Manning, 2016), subword units (Sennrich et al., 2015), and reading text as bytes (Gillick et al., 2015). However, most of these techniques still have no mechanism for handling low frequency characters, which are the target of this work.

Finally, there are works on improving embeddings with radicals, which explicitly splits Chinese characters into radicals based on a dictionary

of what radicals are included in which characters (Li et al., 2015; Shi et al., 2015; Yin et al., 2016). The motivation of this method is similar to ours, but is only applicable to Chinese, in contrast to the method in this paper, which works on any language for which we can render text.

7 Conclusion and Future Work

In this paper, we proposed a new framework that utilizes appearance of characters, convolutional neural networks, recurrent neural networks to learn embeddings that are compositional in the component parts of the characters. More specifically, we collected a Wikipedia dataset, which consists of short titles of three different languages and satisfies the compositionality in the characters of the language. Next, we proposed an end-to-end model that learns visual embeddings for characters using CNN and showed that the features extracted from the CNN include both visual and semantic information. Furthermore, we showed that our VISUAL model outperforms the LOOKUP baseline model in low frequency instances. Additionally, by examining the character embeddings visually, we found that our VISUAL model is able to learn visually related embeddings.

In summary, we tackled the problem of rare characters by using embeddings learned from images. In the future, we hope to further generalize this method to other tasks such as pronunciation estimation, which can take advantage of the fact that pronunciation information is encoded in parts of the characters as demonstrated in Fig. 1, or machine translation, which could benefit from a wholistic view that considers both semantics and pronunciation. We also hope to apply the model to other languages with complicated compositional writing systems, potentially including historical texts such as hieroglyphics or cuneiform.

Acknowledgments

We thank Taylor Berg-Kirkpatrick, Adhiguna Kuncoro, Chen-Hsuan Lin, Wei-Cheng Chang, Wei-Ning Hsu and the anonymous reviewers for their enlightening comments and feedbacks.

References

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *ICML*. pages 1899–1907.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Ronan Collobert, Samy Bengio, and Johnny Marithoz. 2002. Torch: A modular machine learning software library.

Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. 1990. Advances in neural information processing systems 2. pages 396–404.

Peter T Daniels and William Bright. 1996. *The world's writing systems*. Oxford University Press.

Hal Daumé and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *ACL-HLT*. pages 407–412.

Yuntian Deng, Anssi Kanervisto, and Alexander M. Rush. 2016. What you get is what you see: A visual markup decompiler. *arXiv preprint arXiv:1609.04938* .

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. *ACL* .

Gottlob Frege and John Langshaw Austin. 1980. *The foundations of arithmetic: A logico-mathematical enquiry into the concept of number*. Northwestern University Press.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103* .

Nizar Habash. 2008. Four techniques for online handling of out-of-vocabulary words in Arabic-English statistical machine translation. In *HLT-Short*. pages 57–60.

Mohit Iyyer, Varun Manjunatha, and Jordan L Boyd-Graber. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *ACL* pages 655–665.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *CVPR*. pages 1725–1732.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *EMNLP-CoNLL*. pages 698–707.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*. pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*. pages 3294–3302.

Yanran Li, Wenjie Li, Fei Sun, and Sujian Li. 2015. Component-enhanced chinese character embeddings. *EMNLP* pages 829–834.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*. pages 1520–1530.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *ACL* pages 1054–1063.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*. pages 104–113.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. pages 3111–3119.

Mej Newman. 2005. Power laws, Pareto distributions and Zipf's law. *CONTEMP PHYS* pages 323–351.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*. pages 147–155.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *ACL* pages 1715–1725.

Xinlei Shi, Junjie Zhai, Xudong Yang, Zehua Xie, and Chao Liu. 2015. Radical embedding: Delving deeper to chinese radicals. In *ACL*. pages 594–598.

Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. 2005. Early versus late fusion in semantic video analysis. In *ACM MM*. pages 399–402.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*. pages 1631–1642.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. pages 3104–3112.

Zoltán Gendler Szabó. 2010. Compositionality. *Stanford encyclopedia of philosophy* .

Antonio Toral and Rafael Munoz. 2006. A proposal to automatically build and maintain gazetteers for named entity recognition by using wikipedia. In *EACL*. pages 56–61.

Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. 2016. Cnn-rnn: A unified framework for multi-label image classification. In *CVPR*. pages 2285–2294.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Rongchao Yin, Quan Wang, Rui Li, Peng Li, and Bin Wang. 2016. Multi-granularity chinese word embedding. *EMNLP* pages 981–986.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *ECCV*. Springer, pages 818–833.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*. pages 649–657.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *CVPR*. pages 4995–5004.

A Progressive Learning Approach to Chinese SRL Using Heterogeneous Data

Qiaolin Xia[†], **Lei Sha**[†], **Baobao Chang**[†] and **Zhifang Sui**[†⋆]
[†]Key Laboratory of Computational Linguistics (Ministry of Education),
School of EECS, Peking University, 100871, Beijing, China
[⋆]Beijing Advanced Innovation Center for Imaging Technology,
Capital Normal University, Beijing, China
{xql,shalei,chbb,szf}@pku.edu.cn

Abstract

Previous studies on Chinese semantic role labeling (SRL) have concentrated on a single semantically annotated corpus. But the training data of single corpus is often limited. Whereas the other existing semantically annotated corpora for Chinese SRL are scattered across different annotation frameworks. But still, Data sparsity remains a bottleneck. This situation calls for larger training datasets, or effective approaches which can take advantage of highly heterogeneous data. In this paper, we focus mainly on the latter, that is, to improve Chinese SRL by using heterogeneous corpora together. We propose a novel progressive learning model which augments the Progressive Neural Network with Gated Recurrent Adapters. The model can accommodate heterogeneous inputs and effectively transfer knowledge between them. We also release a new corpus, Chinese SemBank, for Chinese SRL[1]. Experiments on CPB 1.0 show that our model outperforms state-of-the-art methods.

1 Introduction

Semantic role labeling (SRL) is one of the fundamental tasks in natural language processing because of its important role in information extraction (Bastianelli et al., 2013), statistical machine translation (Aziz et al., 2016; Xiong et al., 2012), and so on.

However, state-of-the-art performance of Chinese SRL is still far from satisfactory. And data sparsity has been a bottleneck which can not be ignored. For English, the most commonly used benchmark dataset PropBank (Xue and Palmer, 2003) has about 54,900 sentences. But for Chinese, there are only 10,364 sentences in Chinese PropBank 1.0 (CPB) (with about 35,700 propositions) (Xue, 2008).

Predicate given: 修改 (revise)

(a) [*ArgM-TMP*在这期间], [*Arg0*全国人大常委会] ...
(Meanwhile, the NPC Standing Committee)
广泛 征求 意见, [*ArgM-ADV*多次] [*ArgM-ADV*反复]
(widely solicit opinions, for many times, repeatedly)
[*Rel*修改] [*Arg1**pro*] 。
(revise, (omitted) .)

(b) [*agent*他们] 对 [*patient*系统]进行了[*Rel*修改] 。
(They, to, system, made, revise .)

Figure 1: Sentences from (**a**) CPB and (**b**) our heterogeneous dataset. In CPB, each predicate (e.g., 修改) has a specific set of core roles given with numbers (e.g., *Arg0*). While our dataset uses a different semantic role set, and all roles are non-predicate-specific.

To mitigate the data sparsity, models incorporating heterogeneous resources have been introduced to improve Chinese SRL performance (Wang et al., 2015; Guo et al., 2016; Li et al., 2016). The heterogeneous resources introduced by these models include other semantically annotated corpora with annotation schema different to that used in PropBank, and even of a different language. The challenge here lies in the fact that those newly introduced resources are heterogeneous in nature, without sharing the same tagging schema, semantic role set, syntactic tag set and domain. For example, Wang et al. (2015) introduced a heterogeneous dataset, Chinese NetBank, by pretraining word embeddings. Specifically, they learn an LSTM RNN model based on NetBank first, then initialize a new model with the

[1]http://www.klcl.pku.edu.cn/ShowNews.aspx?id=156

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2069–2077
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1189

pretrained embeddings obtained from NetBank, and then train it on CPB. Chinese NetBank (Yulin, 2007) is also a corpus annotated with semantic roles, but using a very different role set and annotation schema. Wang's method can inherit knowledge acquired from other resources conveniently, but only at word representation level, missing more generalized semantic meanings in higher hidden layers. Li (2016) proposed a two-pass training approach to use corpora of two languages, but a few non-common roles are ignored in the first pass. Guo et al. (2016) proposed a unified neural network model for SRL and *relation classification* (RC). It can learn two tasks at the same time, but cannot filter out harmful features learned in incompatible tasks.

Recently, *Progressive Neural Networks* (PNN) model was proposed by Rusu et al. (2016) to transfer learned reinforcement learning policies from one game to another, or from simulation to the real robot. PNN "freezes" learned parameters once starting to learn a new task, and it uses lateral connections, namely adapter, to access previously learned features.

Inspired by the PNN model, we propose a progressive learning model to Chinese semantic role labeling in this paper. Especially, we extend the model with Gated Recurrent Adapters (GRA). Since the standard PNN takes pixels as input, policies as output, it is not suitable for SRL task we focus in this context. Moreover, to handle long sentences in the corpus, we enhance adapters with internal memories, and gates to keep the gradient stable. The contributions of this paper are threefold:

1. We reconstruct PNN columns with bidirectional LSTMs to introduce heterogeneous corpora to improve Chinese SRL. The architecture can also be applied to a wider range of NLP tasks, like event extraction and relation classification, etc.

2. We further extend the model with GRA to remember and take advantage of what has been transferred, thus improve the performance on long sentences.

3. We also release a new corpus, Chinese SemBank, which was annotated with the schema different to that used in CPB. We hope that it will be helpful for future work on SRL tasks.

Subjective roles: agent(施事), co-agent(同事), experiencer(当事) , indirect experiencer(接事)
Objective roles: patient(受事), relative(系事), dative(与事) , result(结果), content(内容), target(对象)
Space roles: a point of departure(起点) , a point of arrival(终点) , path(路径), direction(方向), location(处所)
Time roles: start time(起始), end time(结束), time point(时点) , duration(时段)
Comparison roles: comparison subject(比较主体), comparison object(比较对象) , comparison range(比较范围), comparison thing(比较项目) , comparison result(比较结果)
Others: instrument(工具) , material(材料) , manner(方式) , quantity(物量) , range(范围) , reason(原因) , purpose(目的)

Table 1: Semantic roles in Chinese SemBank

We use our new corpus as a heterogeneous resource, and evaluate the proposed model on the benchmark dataset CPB 1.0. The experiment shows that our approach achieves 79.67% F1 score, significantly outperforms existing state-of-the-art systems by a large margin (Section 5).

2 Heterogeneous Corpora for Chinese SRL

In this paper, we provide a new SRL corpus Chinese SemBank (CSB) and use it as an example of heterogeneous data in our experiments. In this section, we first briefly introduce the corpus, then compare it to existing corpora.

Sentences in CSB are from various sources including online articles and news. The vision of this project is to build a very large and complete Chinese semantic corpus in the future. Currently, it only focuses on the predicate-argument structures in a sentence without annotation of the temporal relations and coreference. CBS is different with respect to commonly used dataset CPB in the following aspects:

- In terms of predicate, CSB takes wider range of predicates into account. We not only annotated common verbs, but also nominal verbs, as NomBank does, and state words. Whereas

CPB only annotate common verbs as predicates.

- In terms of semantic roles, CSB has a more fine-grained semantic role set. There are 31 roles defined in five types (as Table. 1 shows). Whereas in CPB, there are totally 23 roles, including core roles and non-core roles.

- CSB does not have any pre-defined frames for predicates because all roles are set to be non-predicate-specific. The reason for not defining frames is that frames may lead inconsistencies in labels. For example, according to Chinese verb formation theory (Sun et al., 2009), in CPB, an *agent* of a verb is often marked as its *Arg0*, but not all *Arg0* are agents. Therefore, roles are defined for predicates with similar syntactic and semantic regularities, rather than single predicate.

Two direct benefits of using stand-alone non-predicate-specific roles are: First, meanings of all semantic roles can be directly inferred from their labels. For instance, roles of things that people are telling (谈) or looking (看) are labeled as 内容/*content*, because verbs like 谈 and 看 are often followed by an object. Second, we can easily annotate sentences with new predicates without defining new frame files.

Other Corpora for Chinese SRL Other popular semantic role labeling corpora include Chinese NomBank (Xue, 2006), Peking University Chinese NetBank (Yulin, 2007). NomBank, often used as a complement to PropBank, annotates nominal predicates and semantic roles according to the similar semantic schema as PropBank does. Peking University Chinese NetBank was created by adding a semantic layer to Peking University Chinese TreeBank (Zhou et al., 1997). It only uses non-predicate-specific roles as we do. And its role set is smaller, which has 20 roles.

3 Challenges in Inheriting Knowledge from Heterogeneous Corpora

Although there are a lot of annotated corpora for Chinese SRL as we mentioned in the previous section, most of them are quite small as compared to that in English. *Data sparsity* remains a bottleneck. This situation calls for larger training dataset, or effective approaches which can take advantage of very heterogeneous datasets. In this paper, we focus on the second problem, that is, *to improve Chinese SRL by using heterogeneous corpora together within one model*.

We will consider the combination of the standard benchmark, CPB 1.0 dataset (Xue and Palmer, 2003), with the new corpus, CSB, because there are a lot of differences between them, as we discussed in Section 2. Consequently, a number of challenges arise for this task. Now we describe them as below.

Inheriting from Different Schema and Role Sets. CPB was annotated with PropBank-style frames and roles, whereas Chinese FrameNet uses its own frames and roles. And our dataset has no frame files and use different role set. Therefore, it is hard to find explicit mapping or hierarchical relationships among their role sets, or decide which system is better, especially when there are more than two resources.

Inheriting from Different Domain/Genre. The datasets mentioned above are composed of sentences from various sources, including news and stories, etc. However, it is well known that adding data in very different genre to training data may hurt parser performance (Bikel, 2004). Therefore, we also need to deal with domain adaptation problem when using heterogeneous data. In other words, the proposed approach should be robust to harmful features learned on incompatible datasets. It can also accommodate potentially different model structures and inputs in the procedure of knowledge fusion.

Inheriting from Different Syntactic Annotation. Unlikes English, previous works (Ding and Chang, 2009; Sun et al., 2009) on Chinese SRL task often use both correct segmentation and part-of-speech tagging, and even treebank gold-standard parses (Xue, 2008) as their features. But some corpora like CPB and NetBank do not share the same PoS tag set, or do not have correct PoS tagging and gold treebank parses at all, like CSB. And in real application scenarios, it is more convenient to use automatic PoS tagging instead of gold-standard tagging on large datasets, as they can be obtained quickly. So to deal with the absence of syntactic features, we adopt automatic PoS tagging when training on CSB in this work.

Some previous techniques, such as finetuning after pretraining (Wang et al., 2015; Li et al., 2016) and multi-task learning (Guo et al., 2016), have

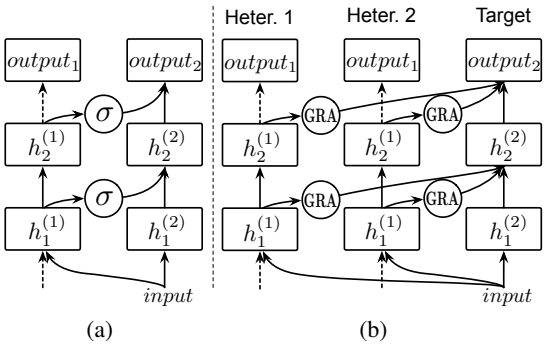


Figure 2: Depiction of the standard Progressive Neural Network architecture (a) and ours PNN GRA model (b). Our model uses Gated Recurrent Adapters (GRA), instead of sigmoid adapters to access previous knowledge in previous columns learned on heterogeneous data. If there are more than one heterogeneous resources available, more columns can be added on the left.

been used to deal with these challenges. Though they can also leverage knowledge from different domains, they have following drawbacks: finetuning cannot avoid catastrophic forgetting because learned parameters, whether embeddings or other hidden weights, will be tuned after the model has been initialized; And multi-task learning cannot ignore previously learned harmful features because some features are learned in shared layers, although it avoids forgetting by randomly selecting a task to learn at each iteration. Therefore, to solve the above-mentioned challenges, we further introduce progressive learning which we believe is more suitable for the task.

4 Progressive Learning Approach

We propose a progressive learning approach which is ideal for combining heterogeneous SRL data for multiple reasons. First, it can accommodate dissimilar inputs with different schema, syntactic information and domain, because it allow models for heterogeneous resources to be extremely different, such as different network structures, different width, and different learning rates, etc. Second, it is immune to forgetting by freezing learned weights and can leverage prior knowledge via lateral connections. Third, the lateral connections can be extended with recurrent structure and gate mechanism to handle with forgetting problem over long distance.

Our model is mainly inspired by Rusu et al. (2016). They proposed *progressive neural networks* for a wide variety of reinforcement learning tasks (e.g. Atari games and robot simulation). In their cases, inputs are pixels, outputs are learned policies. And each column, consisting of simple layers and convolutional layers, is trained to solve a particular Markov Decision Process. But in our case, inputs are sentences annotated using different syntactic tagsets and outputs are semantic role sequences. So we change the structure of columns to recurrent neural networks with LSTM, similar to the model proposed by Wang et al. (2015). Below we first introduce basic progressive neural network architecture, then describe our model, PNN with gated recurrent adapters.

4.1 Progressive Neural Networks

Fig. 2a is an illustration of the basic progressive neural network model. It starts with single column (a neural network), in which there are L hidden layers and the output for ith layer ($i \leq L$) with n_i units is $h_i^1 \in \mathbb{R}^{n_i}$. Θ^1 denotes the parameters to be learned in the first column. When switching to a second corpus, it "freezes" the parameter Θ^1 and randomly initialize a new column with parameters Θ^2 and several lateral connections between two columns so that layer h_i^2 can receive input from both h_{i-1}^2 and h_{i-1}^1. In this straightforward manner, progressive neural networks can make use of columns with any structures or to compile lateral connections in an ensemble setting. To be more general, we calculate the output of ith layer in kth column h_i^k by:

$$h_i^k = f(W_i^k h_{i-1}^k + \sum_{j<k} U_i^{(k:j)} h_{i-1}^j) \qquad (1)$$

where $W_i^k \in \mathbb{R}^{n_i^k \times n_{i-1}^k}$ is the weight matrix of layer i of column k, $U_i^{(k:j)} \in \mathbb{R}^{n_i^k \times n_{i-1}^j}$ are the lateral connections to transfer information from layer $i-1$ of column j to layer i of column k, h_0 is the input of the network. f can be any activation function, such as element-wise non-linearity. Bias term was omitted in the equation.

Adapters. With implicit assumption that there is some "overlap" between the first task and the second task, pretrain-and-finetune learning paradigm is effective, as only slight adjustment to parameters is needed to learn new features. Progressive networks also have ability to transfer knowledge from previous tasks to improve convergence

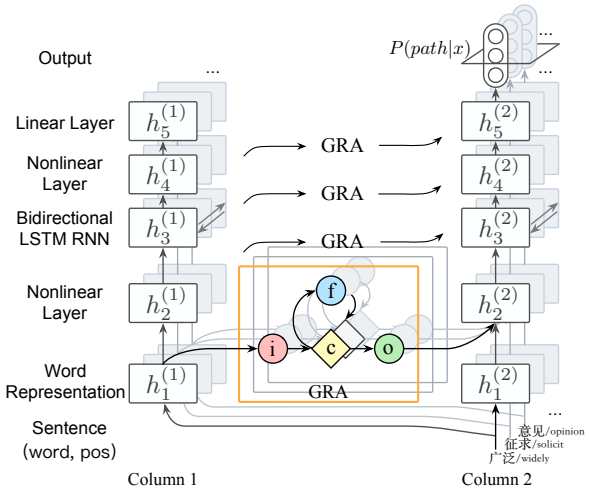


Figure 3: Each column is a stacked bidirectional LSTM RNN model. Two columns are connected by GRAs. There are three gates in each GRA: g_i, g_f, and g_o. The input gate g_i and the forget gate g_f can also be coupled as one uniform gate, that is $g_i = 1 - g_f$.

speed. On the one hand, the model reuse previously learned features from left columns via lateral connections (i.e., *adapters*). On the other hand, new features can be learned by adding more columns incrementally. Moreover, when the "overlap" between two tasks is small, lateral connections can filter out harmful features by sigmoid functions. So in practice, the output of adapters can also be calculated by

$$a_i^{(k:j)} = \sigma(A_i^{(k:j)} \alpha_{i-1}^j h_{i-1}^j) \quad (2)$$

where $A_i^{(k:j)}$ is a matrix to be learned. We treat Equation 2 as one of baseline settings in experiments.

4.2 PNN with Gated Recurrent Adapter for Chinese SRL

We reconstruct PNN with bidirectional LSTM to solve SRL problems. Our model is illustrated in Fig. 3.

First, each column in the PNN architecture is a stacked bidirectional LSTM RNN, rather than convolutional neural networks, because inputs are sentences not pixels, and bi-LSTM RNN has proved powerful for Chinese SRL (Wang et al., 2015).

Second, we enhance the adapter with recurrent structure and gate mechanism, because the simple Multi-Layer Perceptron (MLP) adapters have a limitation: their weights are learned word after word independently. For tasks like transferring reinforcement learning policies, this is enough because there are little dependencies among actions. But in NLP domain, things are different. Therefore, we add internal memory to adapters to help them remember what has been inherited from heterogeneous resource.

Third, to keep gradient stable and balance between long-term and short-term memory, we introduce gate mechanism which has been widely used in RNN models. Intuitively, we call the new adapter *Gated Recurrent Adapter* (GRA).

Formally, let $h_{i-1}^{(<k)} = [h_{i-1}^1, ..., h_{i-1}^j, ..., h_{i-1}^{k-1}]$ be the outputs of $i-1$ layers from the first column to the $(k-1)$th column. The dimensionality of them is $n_{i-1}^{(<k)} = [n_{i-1}^1, ..., n_{i-1}^{k-1}]$. $a^{(<k)}$ is the outputs of $k-1$ adapters with dimension $m^{(<k)} = [m^1, ..., m^{k-1}]$. The output vector is multiplied by a learned matrix W_a initialized by random small values before going to GRAs. Its role is to adjust for the different scales of the different inputs and reduce the dimensionality. Formally, the candidate outputs is

$$\hat{a}_t = f(W_a^j h_t^j + U_a^j a_{t-1}^j) \quad (3)$$

where a_{t-1} is the output of the adapter at the previous time-step. U_a is a weight matrix to learn. The output of an adapter a_t^j of layer i at time t can be formalized as follows,

$$g_i = \sigma(W_i^j h_t^j + U_i^j a_{t-1}^j) \quad (4)$$
$$g_f = \sigma(W_f^j h_t^j + U_f^j a_{t-1}^j) \quad (5)$$
$$g_o = \sigma(W_o^j h_t^j + U_o^j a_{t-1}^j) \quad (6)$$
$$\tilde{a}_t = g_i \odot \hat{a}_t + g_f \odot \tilde{a}_{t-1}^j \quad (7)$$
$$a_t = g_o \odot f(\tilde{a}_{t-1}) \quad (8)$$

where $h^j \in \mathbb{R}^{m_{i-1}^j \times n_{i-1}^j}$ is the outputs of previous layers, $W_f, W_o, W_a \in \mathbb{R}^{m_{i-1} \times n_{i-1}}$, $U_f, U_o, U_a \in \mathbb{R}^{m_{i-1} \times d_{i-1}}$ are parameters to learn. d_{i-1} is the dimension of the inner memory in adapters. $\tilde{a}_t$ represents the inner state of the adapter. f is an activation function, like *tanh*. The input gate and the forget gate can be coupled as a uniform gate, that is $g_i = 1 - g_f$ to alleviate the problem of information redundancy and reduce the possibility of overfitting (Greff et al., 2015).

Finally, we calculate the output of the next layer i of column k by

$$h_i^k = f(W_i^k concat[a^{(<k)}, h_{i-1}^k]) \quad (9)$$

where $W_i \in \mathbb{R}^{n_i^{(k)} \times \sum m_{i-1}^{(<k)}}$ is the parameters in ith layer.

4.3 Training Criteria

We adopt the sentence tagging approach as Wang et al. (2015) did, because words in a sentence may closely be related with each other, independently labeling each word is inappropriate. Sentence tagging approach only consider valid transition paths of tags when calculating the cost. For example, when using IOBES tagging schema, tag transition from *I-Arg0* to *B-Arg0* is invalid, and transition from *I-Arg0* to *I-Arg1* is also invalid because the type of the role changed *inside* the semantic chunk. For each task (column), the log likelihood of sentence x and its correct path y is

$$\log p(y|x,\Theta) = \log \frac{\exp \sum_t^N o_{t,y_t}}{\sum_z \exp \sum_t^{N_i} o_{t,z_t}} \quad (10)$$

where N is the number of words, $o_t \in \mathbb{R}^M$ is the output of the last layer at time t. $y_t = k$ means the tth word has the kth semantic role label. z ranges from all the valid paths of tags.

The negative log likelihood of the whole training set D is

$$J(\Theta) = \sum_{(x,y)\in D} \log p(y|x,\Theta) \quad (11)$$

We minimize $J(\Theta)$ using stochastic gradient descent to learn network parameters Θ. When testing, the best prediction of a sentence can be found using Viterbi algorithm.

5 Experiments

5.1 Experiment Settings

To compare our approach with others, we designed four experimental setups:

(1) A simple LSTM setup on CSB and CPB with automatic PoS tagging. Since CPB is about two times as large as the new corpus, we need to know whether CSB can be used for training good semantic parsers and how much information can be learned from CSB by machine. So we conduct this experiment to provide two baselines for CSB and CPB respectively. In this setup we train and evaluate a one-column LSTM model on CSB.

(2) A simple LSTM setup on CPB with pretrained word embedding on CSB (marked as bi-LSTM+CSB embedding). Previous work found that using pretrained word embeddings can improve performance (Wang et al., 2015) on Chinese SRL. So we conduct this experiment to compare with the method using embeddings trained on large-scale unlabeled data like Gigaword [2], and NetBank.

(3) A two-column finetuning setup where we pretrain the first column on CSB and finetune both two columns on CPB. Clearly, finetuning is a traditional method for continual learning scenarios. But the disadvantage of it is that learned features will be gradually forgotten when the model is adapting new tasks. To assess this empirically, we design this experiment. The model uses the same network structure as PNN does, but it does not "freeze" parameters in the first column when tuning two columns.

(4) A progressive network setup where we train column 1 on CSB, then train column 2 and adapters on CPB. We conduct this experiment to evaluate the proposed model and compare it to all previous methods. To further analyze effectiveness of the new adapter structure, we also conduct an experiment for progressive nets with GRA.

We apply grid-search technique to explore hyper-parameters including learning rates and width of layers.

Preprocessing. We follow the same data setting as previous work (Xue, 2008; Sun et al., 2009), which divided CPB dataset[3] into three parts: 648 files, from `chtb_081.fid` to `chtb_899.fid`, are the training set; 40 files, from `chtb_041.fid` to `chtb_080.fid`, are the development set; 72 files, from `chtb_001.fid` to `chtb_040.fid`, and `chtb_900.fid` to `chtb_931.fid`, are used as the test set.

We also divide shuffled CSB corpus into three sets with similar partition ratios. Currently, there are 10634 sentences in CSB. So 8900 samples are used as training set, 500 samples as development set and the rest 965 samples as test set. We use Stanford Parser[4] for PoS tagging.

5.2 Results

Performance on Chinese SemBank Table 2 gives the results of Experiment 1. We see that precision on CPB with automatic PoS tagging is

[2] https://code.google.com/p/word2vec/
[3] https://catalog.ldc.upenn.edu/LDC2005T23
[4] http://nlp.stanford.edu/software/lex-parser.shtml

Corpus	Pr.(%)	Rec.(%)	F1(%)
1. CSB	75.80	73.45	74.61
2. CPB	76.75	73.03	74.84

Table 2: Results of Chinese SRL tested on CPB and CSB with automatic PoS tagging, using standard LSTM RNN model (Experiment 1).

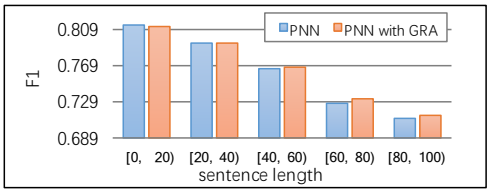


Figure 4: Performance of PNN models with and without GRAs over sentence length. For sentences shorter than 40 words, there is no big difference. But for longer sentences (≥40 words), PNN with GRA model performs significantly better.

about 0.9 percentage point higher than that on CSB, while recall is about 0.4 percentage point lower, and the gap between F1 scores on CPB and CSB is not significant, which is only about 0.3 percentage point, although the size of CSB is smaller. We can explain this by two reasons. First, CSB does not have predicate-specific roles which may lead to inconsistency, as we explained in Section 3. Thus, it might be easier to learn by machine. Second, there are underlying similarities between them: both of them annotate predicate-argument structures. So when there is sufficient training data, difference between scores on testing sets is not very likely to be huge.

Overall, the results indicated that the new annotated corpus CSB is not a bad choice for training semantic parser even when this does not involve larger training sets.

Compare to Methods without Using Heterogeneous Data Table 3 summarizes the SRL performance of previous benchmark methods and our experiments described above. Collobert and Weston only conducted their experiments on English corpus, but we notice that their approach has been implemented and tested on CPB by Wang et al. (2015), so we also put their result here for comparison. We can make several observations from these results. Our approach significantly outperforms Sha et al. (2016) by a large margin (Wilcoxon Signed Rank Test, $p < 0.05$), even without using GRA. This result can prove the ability of our model to capture underlying similarities between heterogeneous SRL resources.

Compare to Methods Using Heterogeneous Resources The results of methods using external language resources are also presented in Table 3. Not surprisingly, we see that the overall best F1 score, 79.67%, is achieved by the progressive nets with the GRAs. Furthermore, as shown in Fig. 4, PNN with GRA performs better on longer sentences, which is consistent with our expectation. Without GRA, the F1 drops 0.37% percentage point to 79.30, confirming that gated recurrent adapter structure is more suitable for our task because it can remember what has been transferred in previous time steps.

Compared to progressive learning methods, finetuning method does not perform well even with the same network structure (Two-column finetuning), but it is still better than simply pretraining word embeddings (bi-LSTM+CSB embedding). This confirms the effectiveness of multi-column learning structure which add capacity to the model by adding new columns. Therefore, as can be seen, our PNN model achieves 79.30% F1 score, outperforming finetuning by 0.88% percentage point, and pretraining embeddings by even larger margin.

To sum up, not only network structures but also learning methods (finetuning/multitask/progressive) can influence the performance of knowledge transfer. According to the results, our PNN approach is more effective than others because it is immune to forgetting and robust to harmful features, and GRA is more suitable for our task than simple adapters.

6 Related Work

6.1 Chinese Semantic Role Labeling

The concept of Semantic Role Labeling is first proposed by Gildea and Jurafsky(2002). Previous work on Chinese SRL mainly focused on how to improve SRL on single corpus. Approaches falls into two categories: feature-based machine learning approaches and neural-network-based approaches. Using feature-based method, Sun and Jurafsky (2004) did the preliminary work and achieved promising results without using any large

Method	F1(%)
Xue (2008) ME	71.90
Collobert and Weston (2008) MTL	74.05
Ding and Chang (2009) CRF	72.64
Yang et al. (2014) Multi-Predicate	75.31
Wang et al. (2015) bi-LSTM	77.09 (+0.00)
Sha et al. (2016) bi-LSTM+QOM	**77.69**
With external language resources	
Wang et al. (2015) +Gigaword embedding	77.21
Wang et al. (2015) +NetBank embedding	**77.59**
Guo et al. (2016) +Relataion Classification	75.46
With CSB corpus	
bi-LSTM+CSB embedding	77.68 (+0.59)
Two-column finetuning	78.42 (+1.33)
Two-column progressive(ours)	79.30 (+2.21)
Two-column Progressive+GRA(ours)	**79.67 (+2.58)**

Table 3: Result comparison on CPB dataset. Compared to learning with single corpus using bi-LSTM model (77.09%), learning with CSB can improve the performance by at list 0.59%. Also the best score (79.67%) was achieved by the PNN GRA model.

annotated corpus. After CPB was built by Xue and Palmer (2003), more complete and systematic research on Chinese SRL were done (Xue and Palmer, 2005; Chen et al., 2006; Ding and Chang, 2009; Yang et al., 2014).

Neural network methods do not rely on handcrafted features. For Chinese SRL, Wang et al. (2015) proposed bidirectional a LSTM RNN model. And based on their work, Sha (2016) proposed quadratic optimization method as a post-processing module and further improved the result.

6.2 Learning with Heterogeneous Data

In this paper, we mainly focus on learning with heterogeneous semantic resource for Chinese SRL. Wang et al. (2015) introduced heterogeneous data by using pretrained embeddings at initialization and achieved promising results. Guo et al. (2016) proposed a multitask learning method with a unified neural network model to learn SRL and relation classification task together and also achieved improvement.

Different from previous work, we proposed a progressive neural network model with gated recurrent adapters to leverage knowledge from heterogeneous semantic data. Compared with previous methods, this approach is more constructive, rather than destructive, because it uses lateral connections to access previously learned features which are fixed when learning new tasks. And by introducing gated recurrent adapters, we further enhance our model to deal with long sentences and achieve state-of-the-art performance on Chinese PropBank.

7 Conclusion and Future Work

In this paper, we proposed a progressive neural network model with gated recurrent adapters to leverage heterogeneous corpus for Chinese SRL. Unlike previous methods like finetuning, ours leverage prior knowledge via lateral connections. Experiments have shown that our model yields better performance on CPB than all baseline models. Moreover, we proposed novel gated recurrent adapter to handle transfer on long sentences, The experiment has proved the effectiveness of the new adapter structure.

We believe that progressive learning with heterogeneous data is a promising avenue to pursue. So in the future, we might try to combine more heterogeneous semantic data for other tasks like event extraction and relation classification, etc.

We also release the new corpus Chinese SemBank for Chinese SRL. We hope that it will be helpful in providing common benchmarks for future work on Chinese SRL tasks.

Acknowledgments

This paper is supported by NSFC project 61375074, National Key Basic Research Program of China 2014CB340504 and Beijing Advanced Innovation Center for Imaging Technology BAICIT-2016016. The contact authors of this paper are Baobao Chang and Zhifang Sui.

References

Wilker Aziz, Miguel Rios, and Lucia Specia. 2016. Shallow semantic trees for smt. In *Proc. of the 6th Workshop on Statistical Machine Translation*. Edinburgh, Scotland, pages 316–322.

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2013. Textual inference and meaning representation in human robot interaction. In *In Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*. pages 65–69.

Daniel M Bikel. 2004. *On the parameter space of generative lexicalized statistical parsing models*. Ph.D. thesis, Citeseer.

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of chinese chunking. In *Proceedings of the COLING/ACL on Main conference poster sessions*. Association for Computational Linguistics, pages 97–104.

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

Weiwei Ding and Baobao Chang. 2009. Word based chinese semantic role labeling with semantic chunking. *International Journal of Computer Processing Of Languages* 22(02n03):133–154.

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

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2015. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069* .

Jiang Guo, Wanxiang Che, Haifeng Wang, Ting Liu, and Jun Xu. 2016. A unified architecture for semantic role labeling and relation classification. In *Proc. of the 26th International Conference on Computational Linguistics (COLING)*.

Tianshi Li, Qi Li, and BaoBao Chang. 2016. Improving chinese semantic role labeling with english proposition bank. In *China National Conference on Chinese Computational Linguistics*. Springer, pages 3–11.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *CoRR* abs/1606.04671.

Lei Sha, Tingsong Jiang, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Capturing argument relationships for chinese semantic role labeling .

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantic parsing of chinese. In *Proceedings of NAACL 2004*. pages 249–256.

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009. Chinese semantic role labeling with shallow parsing. In *Proceedings of the 2009 EMNLP*. Association for Computational Linguistics, pages 1475–1483.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1626–1631.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for smt. In *In Proc. of the 50th Annual Meeting of the Association for Computational Linguistics*. pages 902–911.

Nianwen Xue. 2006. Annotating the predicate-argument structure of chinese nominalizations. In *Proceedings of the fifth international conference on Language Resources and Evaluation*. pages 1382–1387.

Nianwen Xue. 2008. Labeling chinese predicates with semantic roles. *Computational linguistics* 34(2):225–255.

Nianwen Xue and Martha Palmer. 2003. Annotating the propositions in the penn chinese treebank. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*. Association for Computational Linguistics, pages 47–54.

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for chinese verbs. In *In Proceedings of the 19th International Joint Conference on Artificial Intelligence*. pages 1160–1165.

Haitong Yang, Chengqing Zong, et al. 2014. Multi-predicate semantic role labeling. In *EMNLP*. pages 363–373.

Yuan Yulin. 2007. The fineness hierarchy of semantic roles and its application in nlp. *Journal of Chinese Information Processing* 21(4):10–20.

Qiang Zhou, Wei Zhang, and Shiwen Yu. 1997. Building a chinese treebank. *Journal of Chinese Information Processing* 4.

Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings

John Wieting **Kevin Gimpel**
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
{jwieting,kgimpel}@ttic.edu

Abstract

We consider the problem of learning general-purpose, paraphrastic sentence embeddings, revisiting the setting of Wieting et al. (2016b). While they found LSTM recurrent networks to underperform word averaging, we present several developments that together produce the opposite conclusion. These include training on sentence pairs rather than phrase pairs, averaging states to represent sequences, and regularizing aggressively. These improve LSTMs in both transfer learning and supervised settings. We also introduce a new recurrent architecture, the GATED RECURRENT AVERAGING NETWORK, that is inspired by averaging and LSTMs while outperforming them both. We analyze our learned models, finding evidence of preferences for particular parts of speech and dependency relations. [1]

1 Introduction

Modeling sentential compositionality is a fundamental aspect of natural language semantics. Researchers have proposed a broad range of compositional functional architectures (Mitchell and Lapata, 2008; Socher et al., 2011; Kalchbrenner et al., 2014) and evaluated them on a large variety of applications. Our goal is to learn a general-purpose sentence embedding function that can be used unmodified for measuring semantic textual similarity (STS) (Agirre et al., 2012) and can also serve as a useful initialization for downstream tasks. We wish to learn this embedding function such that sentences with high semantic similarity have high cosine similarity in the embedding space. In particular, we focus on the setting of Wieting et al. (2016b), in which models are trained on noisy paraphrase pairs and evaluated on both STS and supervised semantic tasks.

Surprisingly, Wieting et al. found that simple embedding functions—those based on averaging word vectors—outperform more powerful architectures based on long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). In this paper, we revisit their experimental setting and present several techniques that together improve the performance of the LSTM to be superior to word averaging.

We first change data sources: rather than train on noisy phrase pairs from the Paraphrase Database (PPDB; Ganitkevitch et al., 2013), we use noisy *sentence* pairs obtained automatically by aligning Simple English to standard English Wikipedia (Coster and Kauchak, 2011). Even though this data was intended for use by text simplification systems, we find it to be efficient and effective for learning sentence embeddings, outperforming much larger sets of examples from PPDB.

We then show how we can modify and regularize the LSTM to further improve its performance. The main modification is to simply average the hidden states instead of using the final one. For regularization, we experiment with two kinds of dropout and also with randomly scrambling the words in each input sequence. We find that these techniques help in the transfer learning setting and on two supervised semantic similarity datasets as well. Further gains are obtained on the supervised tasks by initializing with our models from the transfer setting.

Inspired by the strong performance of both averaging and LSTMs, we introduce a novel recurrent neural network architecture which we call

[1] Trained models and code are available at http://ttic.uchicago.edu/~wieting.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2078–2088
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1190

the GATED RECURRENT AVERAGING NETWORK (GRAN). The GRAN outperforms averaging and the LSTM in both the transfer and supervised learning settings, forming a promising new recurrent architecture for semantic modeling.

2 Related Work

Modeling sentential compositionality has received a great deal of attention in recent years. A comprehensive survey is beyond the scope of this paper, but we mention popular functional families: neural bag-of-words models (Kalchbrenner et al., 2014), deep averaging networks (DANs) (Iyyer et al., 2015), recursive neural networks using syntactic parses (Socher et al., 2011, 2012, 2013; İrsoy and Cardie, 2014), convolutional neural networks (Kalchbrenner et al., 2014; Kim, 2014; Hu et al., 2014), and recurrent neural networks using long short-term memory (Tai et al., 2015; Ling et al., 2015; Liu et al., 2015). Simple operations based on vector addition and multiplication typically serve as strong baselines (Mitchell and Lapata, 2008, 2010; Blacoe and Lapata, 2012).

Most work cited above uses a supervised learning framework, so the composition function is learned discriminatively for a particular task. In this paper, we are primarily interested in creating general purpose, domain independent embeddings for word sequences. Several others have pursued this goal (Socher et al., 2011; Le and Mikolov, 2014; Pham et al., 2015; Kiros et al., 2015; Hill et al., 2016; Arora et al., 2017; Pagliardini et al., 2017), though usually with the intent to extract useful features for supervised sentence tasks rather than to capture semantic similarity.

An exception is the work of Wieting et al. (2016b). We closely follow their experimental setup and directly address some outstanding questions in their experimental results. Here we briefly summarize their main findings and their attempts at explaining them. They made the surprising discovery that word averaging outperforms LSTMs by a wide margin in the transfer learning setting. They proposed several hypotheses for why this occurs. They first considered that the LSTM was unable to adapt to the differences in sequence length between phrases in training and sentences in test. This was ruled out by showing that neither model showed any strong correlation between sequence length and performance on the test data.

They next examined whether the LSTM was overfitting on the training data, but then showed that both models achieve similar values of the training objective and similar performance on *in-domain* held-out test sets. Lastly, they considered whether their hyperparameters were inadequately tuned, but extensive hyperparameter tuning did not change the story. Therefore, the reason for the performance gap, and how to correct it, was left as an open problem. This paper takes steps toward addressing that problem.

3 Models and Training

3.1 Models

Our goal is to embed a word sequence s into a fixed-length vector. We focus on three compositional models in this paper, all of which use words as the smallest unit of compositionality. We denote the tth word in s as s_t, and we denote its word embedding by x_t.

Our first two models have been well-studied in prior work, so we describe them briefly. The first, which we call AVG, simply averages the embeddings x_t of all words in s. The only parameters learned in this model are those in the word embeddings themselves, which are stored in the word embedding matrix W_w. This model was found by Wieting et al. (2016b) to perform very strongly for semantic similarity tasks.

Our second model uses a long short-term memory (LSTM) recurrent neural network (Hochreiter and Schmidhuber, 1997) to embed s. We use the LSTM variant from Gers et al. (2003) including its "peephole" connections. We consider two ways to obtain a sentence embedding from the LSTM. The first uses the final hidden vector, which we denote h_{-1}. The second, denoted LSTMAVG, averages all hidden vectors of the LSTM. In both variants, the learnable parameters include both the LSTM parameters W_c and the word embeddings W_w.

Inspired by the success of the two models above, we propose a third model, which we call the GATED RECURRENT AVERAGING NETWORK (GRAN). The GATED RECURRENT AVERAGING NETWORK combines the benefits of AVG and LSTMs. In fact it reduces to AVG if the output of the gate is all ones. We first use an LSTM to generate a hidden vector, h_t, for each word s_t in s. Then we use h_t to compute a gate that will be elementwise-multiplied with x_t, resulting in a new, gated hidden vector a_t for each step t:

$$a_t = x_t \odot \sigma(W_x x_t + W_h h_t + b) \quad (1)$$

where W_x and W_h are parameter matrices, b is a parameter vector, and σ is the elementwise logistic sigmoid function. After all a_t have been generated for a sentence, they are averaged to produce the embedding for that sentence. This model includes as learnable parameters those of the LSTM, the word embeddings, and the additional parameters in Eq. (1). For both the LSTM and GRAN models, we use W_c to denote the "compositional" parameters, i.e., all parameters other than the word embeddings.

The motivation for the GRAN is that we are contextualizing the word embeddings prior to averaging. The gate can be seen as an attention, attending to the prior context of the sentence.[2]

We also experiment with four other variations of this model, though they generally were more complex and showed inferior performance. In the first, GRAN-2, the gate is applied to h_t (rather than x_t) to produce a_t, and then these a_t are averaged as before.

GRAN-3 and GRAN-4 use two gates: one applied to x_t and one applied to a_{t-1}. We tried two different ways of computing these gates: for each gate i, $\sigma(W_{x_i}x_t + W_{h_i}h_t + b_i)$ (GRAN-3) or $\sigma(W_{x_i}x_t + W_{h_i}h_t + W_{a_i}a_{t-1} + b_i)$ (GRAN-4). The sum of these two terms comprised a_t. In this model, the last average hidden state, a_{-1}, was used as the sentence embedding after dividing it by the length of the sequence. In these models, we are additionally keeping a running average of the embeddings that is being modified by the context at every time step. In GRAN-4, this running average is also considered when producing the contextualized word embedding.

Lastly, we experimented with a fifth GRAN, GRAN-5, in which we use two gates, calculated by $\sigma(W_{x_i}x_t + W_{h_i}h_t + b_i)$ for each gate i. The first is applied to x_t and the second is applied to h_t. The output of these gates is then summed. Therefore GRAN-5 can be reduced to either word-averaging or averaging LSTM states, depending on the behavior of the gates. If the first gate is all ones and the second all zeros throughout the sequence, the model is equivalent to word-averaging. Conversely, if the first gate is all zeros and the second is all ones throughout the sequence, the model is equivalent to averaging the LSTM states. Further analysis of these models is included in Section 4.

[2]We tried a variant of this model without the gate. We obtain a_t from $f(W_x x_t + W_h h_t + b)$, where f is a nonlinearity, tuned over tanh and ReLU. The performance of the model is significantly worse than the GRAN in all experiments.

3.2 Training

We follow the training procedure of Wieting et al. (2015) and Wieting et al. (2016b), described below. The training data consists of a set S of phrase or sentence pairs $\langle s_1, s_2 \rangle$ from either the Paraphrase Database (PPDB; Ganitkevitch et al., 2013) or the aligned Wikipedia sentences (Coster and Kauchak, 2011) where s_1 and s_2 are assumed to be paraphrases. We optimize a margin-based loss:

$$\min_{W_c, W_w} \frac{1}{|S|} \Bigg(\sum_{\langle s_1, s_2 \rangle \in S} \max(0, \delta - \cos(g(s_1), g(s_2)) + \cos(g(s_1), g(t_1))) + \max(0, \delta - \cos(g(s_1), g(s_2)) + \cos(g(s_2), g(t_2))) \Bigg) + \lambda_c \left\| W_c \right\|^2 + \lambda_w \left\| W_{w_{initial}} - W_w \right\|^2 \tag{2}$$

where g is the model in use (e.g., AVG or LSTM), δ is the margin, λ_c and λ_w are regularization parameters, $W_{w_{initial}}$ is the initial word embedding matrix, and t_1 and t_2 are carefully-selected negative examples taken from a mini-batch during optimization. The intuition is that we want the two phrases to be more similar to each other ($\cos(g(s_1), g(s_2))$) than either is to their respective negative examples t_1 and t_2, by a margin of at least δ.

3.2.1 Selecting Negative Examples

To select t_1 and t_2 in Eq. (2), we simply choose the most similar phrase in some set of phrases (other than those in the given phrase pair). For simplicity we use the mini-batch for this set, but it could be a different set. That is, we choose t_1 for a given $\langle s_1, s_2 \rangle$ as follows:

$$t_1 = \operatorname*{argmax}_{t : \langle t, \cdot \rangle \in S_b \setminus \{\langle s_1, s_2 \rangle\}} \cos(g(s_1), g(t))$$

where $S_b \subseteq S$ is the current mini-batch. That is, we want to choose a negative example t_i that is similar to s_i according to the current model. The downside is that we may occasionally choose a phrase t_i that is actually a true paraphrase of s_i.

4 Experiments

Our experiments are designed to address the empirical question posed by Wieting et al. (2016b): why do LSTMs underperform AVG for transfer

learning? In Sections 4.1.2-4.2, we make progress on this question by presenting methods that bridge the gap between the two models in the transfer setting. We then apply these same techniques to improve performance in the supervised setting, described in Section 4.3. In both settings we also evaluate our novel GRAN architecture, finding it to consistently outperform both AVG and the LSTM.

4.1 Transfer Learning

4.1.1 Datasets and Tasks

We train on large sets of noisy paraphrase pairs and evaluate on a diverse set of 22 textual similarity datasets, including all datasets from every SemEval semantic textual similarity (STS) task from 2012 to 2015. We also evaluate on the SemEval 2015 Twitter task (Xu et al., 2015) and the SemEval 2014 SICK Semantic Relatedness task (Marelli et al., 2014). Given two sentences, the aim of the STS tasks is to predict their similarity on a 0-5 scale, where 0 indicates the sentences are on different topics and 5 indicates that they are completely equivalent. We report the average Pearson's r over these 22 sentence similarity tasks.

Each STS task consists of 4-6 datasets covering a wide variety of domains, including newswire, tweets, glosses, machine translation outputs, web forums, news headlines, image and video captions, among others. Further details are provided in the official task descriptions (Agirre et al., 2012, 2013, 2014, 2015).

4.1.2 Experiments with Data Sources

We first investigate how different sources of training data affect the results. We try two data sources. The first is phrase pairs from the Paraphrase Database (PPDB). PPDB comes in different sizes (S, M, L, XL, XXL, and XXXL), where each larger size subsumes all smaller ones. The pairs in PPDB are sorted by a confidence measure and so the smaller sets contain higher precision paraphrases. PPDB is derived automatically from naturally-occurring bilingual text, and versions of PPDB have been released for many languages without the need for any manual annotation (Ganitkevitch and Callison-Burch, 2014).

The second source of data is a set of sentence pairs automatically extracted from Simple English Wikipedia and English Wikipedia articles by Coster and Kauchak (2011). This data was extracted for developing text simplification systems, where each instance pairs a simple and complex sentence representing approximately the same information. Though the data was obtained for simplification, we use it as a source of training data for learning paraphrastic sentence embeddings. The dataset, which we call SimpWiki, consists of 167,689 sentence pairs.

	AVG	LSTM	LSTMAVG
PPDB	67.7	54.2	64.2
SimpWiki	68.4	59.3	67.5

Table 1: Test results on SemEval semantic textual similarity datasets (Pearson's $r \times 100$) when training on different sources of data: phrase pairs from PPDB or simple-to-standard English Wikipedia sentence pairs from Coster and Kauchak (2011).

To ensure a fair comparison, we select a sample of pairs from PPDB XL such that the number of tokens is approximately the same as the number of tokens in the SimpWiki sentences.[3]

We use PARAGRAM-SL999 embeddings (Wieting et al., 2015) to initialize the word embedding matrix (W_w) for all models. For all experiments, we fix the mini-batch size to 100, and λ_c to 0. We tune the margin δ over $\{0.4, 0.6, 0.8\}$ and λ_w over $\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 0\}$. We train AVG for 7 epochs, and the LSTM for 3, since it converges much faster and does not benefit from 7 epochs. For optimization we use Adam (Kingma and Ba, 2015) with a learning rate of 0.001. We use the 2016 STS tasks (Agirre et al., 2016) for model selection, where we average the Pearson's r over its 5 datasets. We refer to this type of model selection as *test*. For evaluation, we report the average Pearson's r over the 22 other sentence similarity tasks.

The results are shown in Table 1. We first note that, when training on PPDB, we find the same result as Wieting et al. (2016b): AVG outperforms the LSTM by more than 13 points. However, when training both on sentence pairs, the gap shrinks to about 9 points. It appears that part of the inferior performance for the LSTM in prior work was due to training on phrase pairs rather than on sentence pairs. The AVG model also benefits from training on sentences, but not nearly as much as the LSTM.[4]

[3]The PPDB data consists of 1,341,188 phrase pairs and contains 3 more tokens than the SimpWiki data.

[4]We experimented with adding EOS tags at the end of training and test sentences, SOS tags at the start of train-

Our hypothesis explaining this result is that in PPDB, the phrase pairs are short fragments of text which are not necessarily constituents or phrases in any syntactic sense. Therefore, the sentences in the STS test sets are quite different from the fragments seen during training. We hypothesize that while word-averaging is relatively unaffected by this difference, the recurrent models are much more sensitive to overall characteristics of the word sequences, and the difference between train and test matters much more.

These results also suggest that the SimpWiki data, even though it was developed for text simplification, may be useful for other researchers working on semantic textual similarity tasks.

4.1.3 Experiments with LSTM Variations

We next compare LSTM and LSTMAVG. The latter consists of averaging the hidden vectors of the LSTM rather than using the final hidden vector as in prior work (Wieting et al., 2016b). We hypothesize that the LSTM may put more emphasis on the words at the end of the sentence than those at the beginning. By averaging the hidden states, the impact of all words in the sequence is better taken into account. Averaging also makes the LSTM more like AVG, which we know to perform strongly in this setting.

The results on AVG and the LSTM models are shown in Table 1. When training on PPDB, moving from LSTM to LSTMAVG improves performance by 10 points, closing most of the gap with AVG. We also find that LSTMAVG improves by moving from PPDB to SimpWiki, though in both cases it still lags behind AVG.

4.2 Experiments with Regularization

We next experiment with various forms of regularization. Previous work (Wieting et al., 2016b,a) only used L_2 regularization. Wieting et al. (2016b) also regularized the word embeddings back to their initial values. Here we use L_2 regularization as well as several additional regularization methods we describe below.

We try two forms of dropout. The first is just standard dropout (Srivastava et al., 2014) on the word embeddings. The second is "word dropout", which drops out entire word embeddings with some probability (Iyyer et al., 2015).

We also experiment with scrambling the inputs. For a given mini-batch, we go through each sentence pair and, with some probability, we shuffle the words in each sentence in the pair. When scrambling a sentence pair, we always shuffle both sentences in the pair. We do this before selecting negative examples for the mini-batch. The motivation for scrambling is to make it more difficult for the LSTM to memorize the sequences in the training data, forcing it to focus more on the identities of the words and less on word order. Hence it will be expected to behave more like the word averaging model.[5]

We also experiment with combining scrambling and dropout. In this setting, we tune over scrambling with either word dropout or dropout.

The settings for these experiments are largely the same as those of the previous section with the exception that we tune λ_w over a smaller set of values: $\{10^{-5}, 0\}$. When using L_2 regularization, we tune λ_c over $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. When using dropout, we tune the dropout rate over $\{0.2, 0.4, 0.6\}$. When using scrambling, we tune the scrambling rate over $\{0.25, 0.5, 0.75\}$. We also include a bidirectional model ("Bi") for both LSTMAVG and the GATED RECURRENT AVERAGING NETWORK. We tune over two ways to combine the forward and backward hidden states; the first simply adds them together and the second uses a single feedforward layer with a $\tanh$ activation.

We try two approaches for model selection. The first, *test* , is the same as was done in Section 4.1.2, where we use the average Pearson's r on the 5 2016 STS datasets. The second tunes based on the average Pearson's r of all 22 datasets in our evaluation. We refer to this as *oracle*.

The results are shown in Table 2. They show that dropping entire word embeddings and scram-

ing and test sentences, adding both, and adding neither. We treated adding these tags as hyperparameters and tuned over these four settings along with the other hyperparameters in the original experiment. Interestingly, we found that adding these tags, especially EOS, had a large effect on the LSTM when training on SimpWiki, improving performance by 6 points. When training on PPDB, adding EOS tags only improved performance by 1.6 points.

The addition of the tags had a smaller effect on LSTMAVG. Adding EOS tags improved performance by 0.3 points on SimpWiki and adding SOS tags on PPDB improved performance by 0.9 points.

[5] We also tried some variations on scrambling that did not yield significant improvements: scrambling after obtaining the negative examples, partially scrambling by performing n swaps where n comes from a Poisson distribution with a tunable λ, and scrambling individual sentences with some probability instead of always scrambling both in the pair.

Model	Regularization	Oracle	2016 STS
AVG	none	68.5	68.4
	dropout	68.4	68.3
	word dropout	68.3	68.3
LSTM	none	60.6	59.3
	L_2	60.3	56.5
	dropout	58.1	55.3
	word dropout	66.2	65.3
	scrambling	66.3	65.1
	dropout, scrambling	68.4	68.4
LSTMAVG	none	67.7	67.5
	dropout, scrambling	69.2	68.6
BiLSTMAVG	dropout, scrambling	**69.4**	**68.7**

Table 2: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) when experimenting with different regularization techniques.

Model	Oracle	STS 2016
GRAN (no reg.)	68.0	68.0
GRAN	69.5	**68.9**
GRAN-2	68.8	68.1
GRAN-3	69.0	67.2
GRAN-4	68.6	68.1
GRAN-5	66.1	64.8
BiGRAN	**69.7**	68.4

Table 3: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) for the GRAN architectures. The first row, marked as (no reg.) is the GRAN without any regularization. The other rows show the result of the various GRAN models using dropout and scrambling.

bling input sequences is very effective in improving the result of the LSTM, while neither type of dropout improves AVG. Moreover, averaging the hidden states of the LSTM is the most effective modification to the LSTM in improving performance. All of these modifications can be combined to significantly improve the LSTM, finally allowing it to overtake AVG.

In Table 3, we compare the various GRAN architectures. We find that the GRAN provides a small improvement over the best LSTM configuration, possibly because of its similarity to AVG. It also outperforms the other GRAN models, despite being the simplest.

In Table 4, we show results on all individual STS evaluation datasets after using STS 2016 for model selection (unidirectional models only). The LSTMAVG and GATED RECURRENT AVERAGING NETWORK are more closely correlated in performance, in terms of Spearman's ρ and Pearson'r r, than either is to AVG. But they do differ significantly in some datasets, most notably in those comparing machine translation output with its reference. Interestingly, both the LSTMAVG and GATED RECURRENT AVERAGING NETWORK significantly outperform AVG in the datasets focused on comparing glosses like *OnWN* and *FNWN*. Upon examination, we found that these datasets, especially 2013 *OnWN*, contain examples of low similarity with high word overlap. For example, the pair ⟨*the act of preserving or protecting something.*, *the act of decreasing or reducing something.*⟩ from 2013 *OnWN* has a gold similarity score of 0.4. It appears that AVG was fooled by the high amount of word overlap in such pairs, while the other two models were better able to recognize the semantic differences.

Dataset	LSTMAVG	AVG	GRAN
MSRpar	**49.0**	45.9	47.7
MSRvid	84.3	85.1	**85.2**
SMT-eur	**51.2**	47.5	49.3
OnWN	**71.5**	71.2	71.5
SMT-news	**68.0**	58.2	58.7
STS 2012 Average	**64.8**	61.6	62.5
headline	**77.3**	76.9	76.1
OnWN	81.2	72.8	**81.4**
FNWN	53.2	50.2	**55.6**
SMT	**40.7**	38.0	40.3
STS 2013 Average	63.1	59.4	**63.4**
deft forum	**56.6**	55.6	55.7
deft news	78.0	**78.5**	77.1
headline	74.5	**75.1**	72.8
images	84.7	85.6	**85.8**
OnWN	84.9	81.4	**85.1**
tweet news	76.3	**78.7**	78.7
STS 2014 Average	75.8	75.8	**75.9**
answers-forums	71.8	70.6	**73.1**
answers-students	71.1	**75.8**	72.9
belief	75.3	76.8	**78.0**
headline	79.5	**80.3**	78.6
images	85.8	**86.0**	85.8
STS 2015 Average	76.7	**77.9**	77.7
2014 SICK	71.3	72.4	**72.9**
2015 Twitter	**52.1**	**52.1**	50.2

Table 4: Results on SemEval textual similarity datasets (Pearson's $r \times 100$). The highest score in each row is in boldface.

4.3 Supervised Text Similarity

We also investigate if these techniques can improve LSTM performance on supervised semantic textual similarity tasks. We evaluate on two supervised datasets. For the first, we start with the 20 SemEval STS datasets from 2012-2015 and then use 40% of each dataset for training, 10% for validation, and the remaining 50% for testing. There are 4,481 examples in training, 1,207 in validation, and 6,060 in the test set. The second is the SICK 2014 dataset, using its standard training, validation, and test sets. There are 4,500 sentence pairs

in the training set, 500 in the development set, and 4,927 in the test set. The SICK task is an easier learning problem since the training examples are all drawn from the same distribution, and they are mostly shorter and use simpler language. As these are supervised tasks, the sentence pairs in the training set contain manually-annotated semantic similarity scores.

We minimize the loss function[6] from Tai et al. (2015). Given a score for a sentence pair in the range $[1, K]$, where K is an integer, with sentence representations h_L and h_R, and model parameters θ, they first compute:

$$h_\times = h_L \odot h_R, \;\; h_+ = |h_L - h_R|,$$
$$h_s = \sigma\left(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}\right),$$
$$\hat{p}_\theta = \text{softmax}\left(W^{(p)} h_s + b^{(p)}\right),$$
$$\hat{y} = r^T \hat{p}_\theta,$$

where $r^T = [1\ 2\ \ldots\ K]$. They then define a sparse target distribution p that satisfies $y = r^T p$:

$$p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & i = \lfloor y \rfloor \\ 0 & \text{otherwise} \end{cases}$$

for $1 \leq i \leq K$. Then they use the following loss, the regularized KL-divergence between p and $\hat{p}_\theta$:

$$J(\theta) = \frac{1}{m} \sum_{k=1}^{m} \text{KL}\left(p^{(k)} \,\middle\|\, \hat{p}_\theta^{(k)}\right),$$

where m is the number of training pairs.

We experiment with the LSTM, LSTMAVG, and AVG models with dropout, word dropout, and scrambling tuning over the same hyperparameter as in Section 4.2. We again regularize the word embeddings back to their initial state, tuning λ_w over $\{10^{-5}, 0\}$. We used the validation set for each respective dataset for model selection.

The results are shown in Table 5. The GATED RECURRENT AVERAGING NETWORK has the best performance on both datasets. Dropout helps the word-averaging model in the STS task, unlike in the transfer learning setting. The LSTM benefits slightly from dropout, scrambling, and averaging on their own individually with the exception of word dropout on both datasets and averaging on the SICK dataset. However, when combined, these modifications are able to significantly improve the performance of the LSTM, bringing it much closer in performance to AVG. This experiment indicates that these modifications when training LSTMs are beneficial outside the transfer learning setting, and can potentially be used to improve performance for the broad range of problems that use LSTMs to model sentences.

[6]This objective function has been shown to perform very strongly on text similarity tasks, significantly better than squared or absolute error.

Model	Regularization	STS	SICK	Avg.
AVG	none	79.2	85.2	82.2
	dropout	80.7	84.5	82.6
	word dropout	79.3	81.8	80.6
LSTM	none	68.4	80.9	74.7
	dropout	69.6	81.3	75.5
	word dropout	68.0	76.4	72.2
	scrambling	74.2	84.4	79.3
	dropout, scrambling	75.0	84.2	79.6
LSTMAVG	none	69.0	79.5	74.3
	dropout	69.2	79.4	74.3
	word dropout	65.6	76.1	70.9
	scrambling	76.5	83.2	79.9
	dropout, scrambling	76.5	84.0	80.3
GRAN	none	79.7	85.2	82.5
	dropout	79.7	84.6	82.2
	word dropout	77.3	83.0	80.2
	scrambling	81.4	**85.3**	**83.4**
	dropout, scrambling	**81.6**	85.1	**83.4**

Table 5: Results from supervised training on the STS and SICK datasets (Pearson's $r \times 100$). The last column is the average result on the two datasets.

Model	STS	SICK	Avg.
GRAN	**81.6**	85.3	**83.5**
GRAN-2	77.4	85.1	81.3
GRAN-3	81.3	85.4	83.4
GRAN-4	80.1	**85.5**	82.8
GRAN-5	70.9	83.0	77.0

Table 6: Results from supervised training on the STS and SICK datasets (Pearson's $r \times 100$) for the GRAN architectures. The last column is the average result on the two datasets.

In Table 6 we compare the various GRAN architectures under the same settings as the previous experiment. We find that the GRAN still has the best overall performance.

We also experiment with initializing the supervised models using our pretrained sentence model parameters, for the AVG model (no regularization), LSTMAVG (dropout, scrambling), and GATED RECURRENT AVERAGING NETWORK (dropout, scrambling) models from Table 2 and Table 3. We both initialize and then regularize back to these initial values, referring to this setting as "universal".[7]

[7]In these experiments, we tuned λ_w over $\{10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 0\}$

#	Sentence 1	Sentence 2	LAVG	AVG	Gold
1	the lamb is looking at the camera.	a cat looking at the camera.	**3.42**	4.13	0.8
2	he also said shockey is "living the dream life of a new york athlete."	"jeremy's a good guy," barber said, adding:"jeremy is living the dream life of the new york athlete."	**3.55**	4.22	2.75
3	bloomberg chips in a billion	bloomberg gives $1.1 b to university	**3.99**	3.04	4.0
4	in other regions, the sharia is imposed.	in other areas, sharia law is being introduced by force.	**4.44**	3.72	4.75
5	three men in suits sitting at a table.	two women in the kitchen looking at a object.	3.33	**2.79**	0.0
6	we never got out of it in the first place!	where does the money come from in the first place?	4.00	**3.33**	0.8
7	two birds interacting in the grass.	two dogs play with each other outdoors.	3.44	**2.81**	0.2

Table 7: Illustrative sentence pairs from the STS datasets showing errors made by LSTMAVG and AVG. The last three columns show the gold similarity score, the similarity score of LSTMAVG, and the similarity score of AVG. Boldface indicates smaller error compared to gold scores.

Model	Regularization	STS	SICK
AVG	dropout	80.7	84.5
	dropout, universal	**82.9**	85.6
LSTMAVG	dropout, scrambling	76.5	84.0
	dropout, scrambling, universal	81.3	85.2
GRAN	dropout, scrambling	81.6	85.1
	dropout, scrambling, universal	82.7	**86.0**

Table 8: Impact of initializing and regularizing toward universal models (Pearson's $r \times 100$) in supervised training.

The results are shown in Table 8. Initializing and regularizing to the pretrained models significantly improves the performance for all three models, justifying our claim that these models serve a dual purpose: they can be used a black box semantic similarity function, and they possess rich knowledge that can be used to improve the performance of downstream tasks.

5 Analysis

5.1 Error Analysis

We analyze the predictions of AVG and the recurrent networks, represented by LSTMAVG, on the 20 STS datasets. We choose LSTMAVG as it correlates slightly less strongly with AVG than the GRAN on the results over all SemEval datasets used for evaluation. We scale the models' cosine similarities to lie within $[0, 5]$, then compare the predicted similarities of LSTMAVG and AVG to the gold similarities. We analyzed instances in which each model would tend to overestimate or underestimate the gold similarity relative to the other. These are illustrated in Table 7.

We find that AVG tends to overestimate the semantic similarity of a sentence pair, relative to LSTMAVG, when the two sentences have a lot of word or synonym overlap, but have either important differences in key semantic roles or where one sentence has significantly more content than the other. These phenomena are shown in examples 1 and 2 in Table 7. Conversely, AVG tends to underestimate similarity when there are one-word-to-multiword paraphrases between the two sentences as shown in examples 3 and 4.

LSTMAVG tends to overestimate similarity when the two inputs have similar sequences of syntactic categories, but the meanings of the sentences are different (examples 5, 6, and 7). Instances of LSTMAVG underestimating the similarity relative to AVG are relatively rare, and those that we found did not have any systematic patterns.

5.2 GRAN Gate Analysis

We also investigate what is learned by the gating function of the GATED RECURRENT AVERAGING NETWORK. We are interested to see whether its estimates of importance correlate with those of traditional syntactic and (shallow) semantic analysis.

We use the oracle trained GATED RECURRENT AVERAGING NETWORK from Table 3 and calculate the L_1 norm of the gate after embedding 10,000 sentences from English Wikipedia.[8] We also automatically tag and parse these sentences using the Stanford dependency parser (Manning et al., 2014). We then compute the average gate L_1 norms for particular part-of-speech tags, dependency arc labels, and their conjunction.

Table 9 shows the highest/lowest average norm tags and dependency labels. The network prefers nouns, especially proper nouns, as well as cardinal numbers, which is sensible as these are among the most discriminative features of a sentence.

Analyzing the dependency relations, we find

and λ_c over $\{10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 0\}$.

[8] We selected only sentences of less than or equal to 15 tokens to ensure more accurate parsing.

POS		Dep. Label	
top 10	bot. 10	top 10	bot. 10
NNP	TO	number	possessive
NNPS	WDT	nn	cop
CD	POS	num	det
NNS	DT	acomp	auxpass
VBG	WP	appos	prep
NN	IN	pobj	cc
JJ	CC	vmod	mark
UH	PRP	dobj	aux
VBN	EX	amod	expl
JJS	WRB	conj	neg

Table 9: POS tags and dependency labels with highest and lowest average GATED RECURRENT AVERAGING NETWORK gate L_1 norms. The lists are ordered from highest norm to lowest in the top 10 columns, and lowest to highest in the bottom 10 columns.

Dep. Label	Weight
xcomp	170.6
acomp	167.1
root	157.4
amod	143.1
advmod	121.6

Table 10: Average L_1 norms for adjectives (JJ) with selected dependency labels.

that nouns in the object position tend to have higher weight than nouns in the subject position. This may relate to topic and focus; the object may be more likely to be the "new" information related by the sentence, which would then make it more likely to be matched by the other sentence in the paraphrase pair.

We find that the weights of adjectives depend on their position in the sentence, as shown in Table 10. The highest norms appear when an adjective is an xcomp, acomp, or root; this typically means it is residing in an object-like position in its clause. Adjectives that modify a noun (amod) have medium weight, and those that modify another adjective or verb (advmod) have low weight.

Dep. Label	Weight
pcomp	190.0
amod	178.3
xcomp	176.8
vmod	170.6
root	161.8
auxpass	125.4
prep	121.2

Table 11: Average L_1 norms for words with the tag VBG with selected dependency labels.

Lastly, we analyze words tagged as VBG, a highly ambiguous tag that can serve many syntactic roles in a sentence. As shown in Table 11, we find that when they are used to modify a noun (amod) or in the object position of a clause (xcomp, pcomp) they have high weight. Medium weight appears when used in verb phrases (root, vmod) and low weight when used as prepositions or auxiliary verbs (prep, auxpass).

6 Conclusion

We showed how to modify and regularize LSTMs to improve their performance for learning paraphrastic sentence embeddings in both transfer and supervised settings. We also introduced a new recurrent network, the GATED RECURRENT AVERAGING NETWORK, that improves upon both AVG and LSTMs for these tasks, and we release our code and trained models.

Furthermore, we analyzed the different errors produced by AVG and the recurrent methods and found that the recurrent methods were learning composition that wasn't being captured by AVG. We also investigated the GRAN in order to better understand the compositional phenomena it was learning by analyzing the L_1 norm of its gate over various inputs.

Future work will explore additional data sources, including from aligning different translations of novels (Barzilay and McKeown, 2001), aligning new articles of the same topic (Dolan et al., 2004), or even possibly using machine translation systems to translate bilingual text into paraphrastic sentence pairs. Our new techniques, combined with the promise of new data sources, offer a great deal of potential for improved universal paraphrastic sentence embeddings.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. We thank the developers of Theano (Theano Development Team, 2016) and NVIDIA Corporation for donating GPUs used in this research.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. *Proceedings of SemEval* pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the International Conference on Learning Representations*.

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th annual meeting on Association for Computational Linguistics*.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

William Coster and David Kauchak. 2011. Simple english wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of HLT-NAACL*.

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

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8).

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

Ozan İrsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

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

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. *arXiv preprint arXiv:1703.02507* .

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1).

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016a. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016b. Towards universal paraphrastic sentence embeddings. In *Proceedings of the International Conference on Learning Representations*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL (TACL)* .

Wei Xu, Chris Callison-Burch, and William B Dolan. 2015. SemEval-2015 task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*.

Ontology-Aware Token Embeddings for Prepositional Phrase Attachment

Pradeep Dasigi[1] **Waleed Ammar**[2] **Chris Dyer**[1,3] **Eduard Hovy**[1]
[1]Language Technologies Institute, Carnegie Mellon University, Pittsburgh PA, USA
[2]Allen Institute for Artificial Intelligence, Seattle WA, USA
[3]DeepMind, London, UK
pdasigi@cs.cmu.edu, wammar@allenai.org,
cdyer@cs.cmu.edu, hovy@cmu.edu

Abstract

Type-level word embeddings use the same set of parameters to represent all instances of a word regardless of its context, ignoring the inherent lexical ambiguity in language. Instead, we embed semantic concepts (or synsets) as defined in WordNet and represent a word token in a particular context by estimating a distribution over relevant semantic concepts. We use the new, context-sensitive embeddings in a model for predicting prepositional phrase (PP) attachments and jointly learn the concept embeddings and model parameters. We show that using context-sensitive embeddings improves the accuracy of the PP attachment model by 5.4% absolute points, which amounts to a 34.4% relative reduction in errors.

1 Introduction

Type-level word embeddings map a word type (i.e., a surface form) to a dense vector of real numbers such that similar word types have similar embeddings. When pre-trained on a large corpus of unlabeled text, they provide an effective mechanism for generalizing statistical models to words which do not appear in the labeled training data for a downstream task.

In accordance with standard terminology, we make the following distinction between types and tokens in this paper: By word types, we mean the surface form of the word, whereas by tokens we mean the instantiation of the surface form in a context. For example, the same word type *'pool'* occurs as two different tokens in the sentences *"He sat by the pool,"* and *"He played a game of pool."*

Most word embedding models define a single vector for each word type. However, a fundamental flaw in this design is their inability to distinguish between different meanings and abstractions of the same word. In the two sentences shown above, the word *'pool'* has different meanings, but the same representation is typically used for both of them. Similarly, the fact that *'pool'* and *'lake'* are both kinds of water bodies is not explicitly incorporated in most type-level embeddings. Furthermore, it has become a standard practice to tune pre-trained word embeddings as model parameters during training for an NLP task (e.g., Chen and Manning, 2014; Lample et al., 2016), potentially allowing the parameters of a frequent word in the labeled training data to drift away from related but rare words in the embedding space.

Previous work partially addresses these problems by estimating concept embeddings in WordNet (e.g., Rothe and Schütze, 2015), or improving word representations using information from knowledge graphs (e.g., Faruqui et al., 2015). However, it is still not clear how to use a lexical ontology to derive context-sensitive token embeddings.

In this work, we represent a word token in a given context by estimating a context-sensitive probability distribution over relevant concepts in WordNet (Miller, 1995) and use the expected value (i.e., weighted sum) of the concept embeddings as the token representation (see §2). We take a task-centric approach towards doing this, and learn the token representations jointly with the task-specific parameters. In addition to providing context-sensitive token embeddings, the proposed method implicitly regularizes the embeddings of related words by forcing related words to share similar concept embeddings. As a result, the representation of a rare word which does not appear in the training data for a downstream task benefits from all the updates to related words which share one or more concept embeddings.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2089–2098
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1191

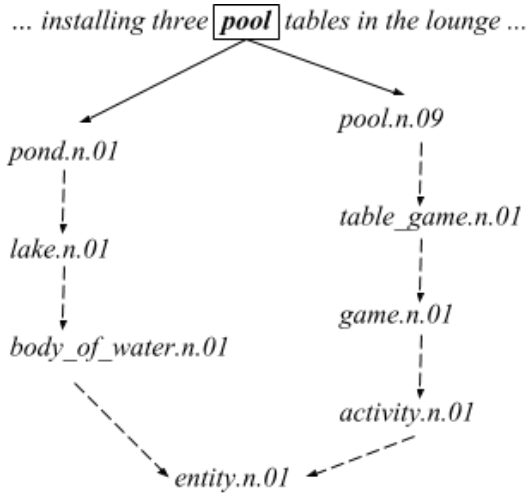


Figure 1: An example grounding for the word *'pool'*. Solid arrows represent possible senses and dashed arrows represent hypernym relations. Note that the same set of concepts are used to ground the word *'pool'* regardless of its context. Other WordNet senses for *'pool'* were removed from the figure for simplicity.

Our approach to context-sensitive embeddings assumes the availability of a lexical ontology. While this work relies on WordNet, and we exploit the order of senses given by WordNet, our model is, in principle applicable to any ontology, with appropriate modifications. In this work, we do not assume the inputs are sense tagged. We use the proposed embeddings to predict prepositional phrase (PP) attachments (see §3), a challenging problem which emphasizes the selectional preferences between words in the PP and each of the candidate head words. Our empirical results and detailed analysis (see §4) show that the proposed embeddings effectively use WordNet to improve the accuracy of PP attachment predictions.

2 WordNet-Grounded Context-Sensitive Token Embeddings

In this section, we focus on defining our context-sensitive token embeddings. We first describe our grounding of word types using WordNet concepts. Then, we describe our model of context-sensitive token-level embeddings as a weighted sum of WordNet concept embeddings.

2.1 WordNet Grounding

We use WordNet to map each word type to a set of synsets, including possible generalizations or abstractions. Among the labeled relations defined in WordNet between different synsets, we focus on the hypernymy relation to help model generalization and selectional preferences between words, which is especially important for predicting PP attachments (Resnik, 1993). To ground a word type, we identify the set of (direct and indirect) hypernyms of the WordNet senses of that word. A simplified grounding of the word 'pool' is illustrated in Figure 1. This grounding is key to our model of token embeddings, to be described in the following subsections.

2.2 Context-Sensitive Token Embeddings

Our goal is to define a context-sensitive model of token embeddings which can be used as a drop-in replacement for traditional type-level word embeddings.

Notation. Let $\mathit{Senses}(w)$ be the list of synsets defined as possible word senses of a given word type w in WordNet, and $\mathit{Hypernyms}(s)$ be the list of hypernyms for a synset s.[1] For example, according to Figure 1:

$$\mathit{Senses}(\text{pool}) = [\text{pond.n.01, pool.n.09}], \text{and}$$
$$\mathit{Hypernyms}(\text{pond.n.01}) = [\text{pond.n.01, lake.n.01,}$$
$$\text{body\_of\_water.n.01, entity.n.01}]$$

Each WordNet synset s is associated with a set of parameters $\mathbf{v}_s \in \mathbb{R}^n$ which represent its embedding. This parameterization is similar to that of Rothe and Schütze (2015).

Embedding model. Given a sequence of tokens $\boldsymbol{t}$ and their corresponding word types $\boldsymbol{w}$, let $\mathbf{u}_i \in \mathbb{R}^n$ be the embedding of the word token t_i at position i. Unlike most embedding models, the token embeddings $\mathbf{u}_i$ are not parameters. Rather, $\mathbf{u}_i$ is computed as the expected value of concept embeddings used to ground the word type w_i corresponding to the token t_i:

$$\mathbf{u}_i = \sum_{s \in \mathit{Senses}(w_i)} \sum_{s' \in \mathit{Hypernyms}(s)} p(s, s' \mid \boldsymbol{t}, \boldsymbol{w}, i)\, \mathbf{v}_{s'} \quad (1)$$

such that

$$\sum_{s \in \mathit{Senses}(w_i)} \sum_{s' \in \mathit{Hypernyms}(s)} p(s, s' \mid \boldsymbol{t}, \boldsymbol{w}, i) = 1$$

[1] For notational convenience, we assume that $s \in \mathit{Hypernyms}(s)$.

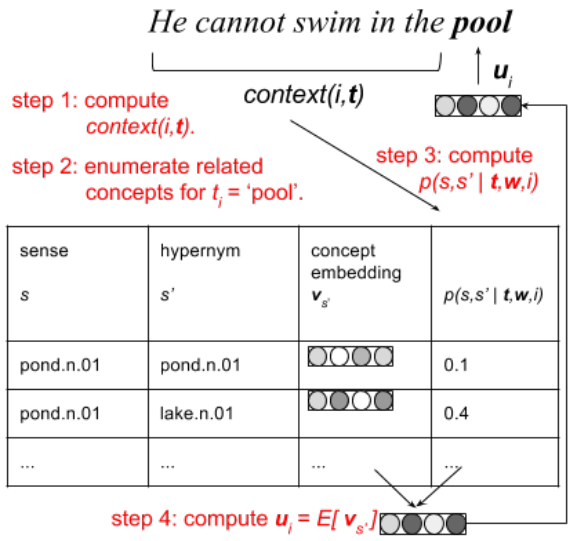


Figure 2: Steps for computing the context-sensitive token embedding for the word *'pool'*, as described in §2.2.

The distribution which governs the expectation over synset embeddings factorizes into two components:

$$p(s, s' \mid \boldsymbol{t}, \boldsymbol{w}, i) \propto \lambda_{w_i} \exp^{-\lambda_{w_i} \, rank(s, w_i)} \times MLP([\mathbf{v}_{s'}; context(i, \boldsymbol{t})]) \quad (2)$$

The first component, $\lambda_{w_i} \exp^{-\lambda_{w_i} \, rank(s,w_i)}$, is a sense prior which reflects the prominence of each word sense for a given word type. Here, we exploit[2] the fact that WordNet senses are ordered in descending order of their frequencies, obtained from sense tagged corpora, and parameterize the sense prior like an exponential distribution. $rank(s, w_i)$ denotes the rank of sense s for the word type w_i, thus $rank(s, w_i) = 0$ corresponds to s being the first sense of w_i. The scalar parameter (λ_{w_i}) controls the decay of the probability mass, which is learned along with the other parameters in the model. Note that sense priors are defined for each word type (w_i), and are shared across all tokens which have the same word type.

$MLP([\mathbf{v}_{s'}; context(i, \boldsymbol{t})])$, the second component, is what makes the token representations context-sensitive. It scores each concept in the WordNet grounding of w_i by feeding the concatenation of the concept embedding and a dense vector that summarizes the textual context into a multilayer perceptron (*MLP*) with two $tanh$ layers followed by a $softmax$ layer. This component is inspired by the soft attention often used in neural machine translation (Bahdanau et al., 2014).[3] The definition of the *context* function is dependent on the encoder used to encode the context. We describe a specific instantiation of this function in §3.

To summarize, Figure 2 illustrates how to compute the embedding of a word token $t_i = \textit{'pool'}$ in a given context:

1. compute a summary of the context $context(i, \boldsymbol{t})$,
2. enumerate related concepts for t_i,
3. compute $p(s, s' \mid \boldsymbol{t}, \boldsymbol{w}, i)$ for each pair (s, s'), and
4. compute $\mathbf{u}_i = \mathbb{E}[\mathbf{v}_{s'}]$.

In the following section, we describe our model for predicting PP attachments, including our definition for *context*.

3 PP Attachment

Disambiguating PP attachments is an important and challenging NLP problem. Since modeling hypernymy and selectional preferences is critical for successful prediction of PP attachments (Resnik, 1993), it is a good fit for evaluating our WordNet-grounded context-sensitive embeddings.

Figure 3, reproduced from Belinkov et al. (2014), illustrates an example of the PP attachment prediction problem. The accuracy of a competitive English dependency parser at predicting the head word of an ambiguous prepositional phrase is 88.5%, significantly lower than the overall unlabeled attachment accuracy of the same parser (94.2%).[4]

This section formally defines the problem of PP attachment disambiguation, describes our baseline model, then shows how to integrate the token-level embeddings in the model.

3.1 Problem Definition

We follow Belinkov et al. (2014)'s definition of the PP attachment problem. Given a preposition p and

[2]Note that for ontologies where such information is not available, our method is still applicable but without this component. We show the effect of using a uniform sense prior in §4.2.

[3]Although soft attention mechanism is typically used to explicitly represent the importance of each item in a sequence, it can also be applied to non-sequential items.

[4]See Table 2 in §4 for detailed results.

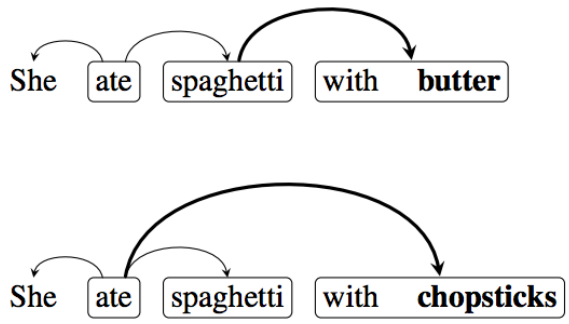


Figure 3: Two sentences illustrating the importance of lexicalization in PP attachment decisions. In the top sentence, the PP *'with butter'* attaches to the noun *'spaghetti'*. In the bottom sentence, the PP *'with chopsticks'* attaches to the verb *'ate'*. **Note:** This figure and caption have been reproduced from Belinkov et al. (2014).

its direct dependent d in the prepositional phrase (PP), our goal is to predict the correct head word for the PP among an ordered list of candidate head words $\boldsymbol{h}$. Each example in the train, validation, and test sets consists of an input tuple $\langle \boldsymbol{h}, p, d \rangle$ and an output index k to identify the correct head among the candidates in $\boldsymbol{h}$. Note that the order of words that form each $\langle \boldsymbol{h}, p, d \rangle$ is the same as that in the corresponding original sentence.

3.2 Model Definition

Both our proposed and baseline models for PP attachment use bidirectional RNN with LSTM cells (bi-LSTM) to encode the sequence $\boldsymbol{t} = \langle h_1, \ldots, h_K, p, d \rangle$.

We score each candidate head by feeding the concatenation of the output bi-LSTM vectors for the head h_k, the preposition p and the direct dependent d through an MLP, with a fully connected *tanh* layer to obtain a non-linear projection of the concatenation, followed by a fully-connected *softmax* layer:

$$p(h_k \text{is head}) = \mathit{MLP}_{\mathit{attach}}([\mathit{lstm\_out}(h_k); \mathit{lstm\_out}(p); \mathit{lstm\_out}(d)]) \quad (3)$$

To train the model, we use cross-entropy loss at the output layer for each candidate head in the training set. At test time, we predict the candidate head with the highest probability according to the model in Eq. 3, i.e.,

$$\hat{k} = \arg\max_k p(h_k \text{is head} = 1). \quad (4)$$

This model is inspired by the Head-Prep-Child-Ternary model of Belinkov et al. (2014). The main difference is that we replace the input features for each token with the output bi-RNN vectors.

We now describe the difference between the proposed and the baseline models. Generally, let $\mathit{lstm\_in}(t_i)$ and $\mathit{lstm\_out}(t_i)$ represent the input and output vectors of the bi-LSTM for each token $t_i \in \boldsymbol{t}$ in the sequence. The outputs at each timestep are obtained by concatenating those of the forward and backward LSTMs.

Baseline model. In the baseline model, we use type-level word embeddings to represent the input vector $\mathit{lstm\_in}(t_i)$ for a token t_i in the sequence. The word embedding parameters are initialized with pre-trained vectors, then tuned along with the parameters of the bi-LSTM and $\mathit{MLP}_{\mathit{attach}}$. We call this model **LSTM-PP**.

Proposed model. In the proposed model, we use token level word embedding as described in §2 as the input to the bi-LSTM, i.e., $\mathit{lstm\_in}(t_i) = \mathbf{u}_i$. The context used for the attention component is simply the hidden state from the previous timestep. However, since we use a bi-LSTM, the model essentially has two RNNs, and accordingly we have two context vectors, and associated attentions. That is, $\mathit{context}_f(i, \boldsymbol{t}) = \mathit{lstm\_in}(t_{i-1})$ for the forward RNN and $\mathit{context}_b(i, \boldsymbol{t}) = \mathit{lstm\_in}(t_{i+1})$ for the backward RNN. Consequently, each token gets two representations, one from each RNN. The synset embedding parameters are initialized with pre-trained vectors and tuned along with the sense decay (λ_w) and MLP parameters from Eq. 2, the parameters of the bi-LSTM and those of $\mathit{MLP}_{\mathit{attach}}$. We call this model **OntoLSTM-PP**.

4 Experiments

Dataset and evaluation. We used the English PP attachment dataset created and made available by Belinkov et al. (2014). The training and test splits contain 33,359 and 1951 labeled examples respectively. As explained in §3.1, the input for each example is 1) an ordered list of candidate head words, 2) the preposition, and 3) the direct

dependent of the preposition. The head words are either nouns or verbs and the dependent is always a noun. All examples in this dataset have at least two candidate head words. As discussed in Belinkov et al. (2014), this dataset is a more realistic PP attachment task than the RRR dataset (Ratnaparkhi et al., 1994). The RRR dataset is a binary classification task with exactly two head word candidates in all examples. The context for each example in the RRR dataset is also limited which defeats the purpose of our context-sensitive embeddings.

Model specifications and hyperparameters. For efficient implementation, we use mini-batch updates with the same number of senses and hypernyms for all examples, padding zeros and truncating senses and hypernyms as needed. For each word type, we use a maximum of S senses and H indirect hypernyms from WordNet. In our initial experiments on a held-out development set (10% of the training data), we found that values greater than $S = 3$ and $H = 5$ did not improve performance. We also used the development set to tune the number of layers in MLP_{attach} separately for the *OntoLSTM-PP* and *LSTM-PP*, and the number of layers in the attention MLP in *OntoLSTM-PP*. When a synset has multiple hypernym paths, we use the shortest one. Finally, words types which do not appear in WordNet are assumed to have one unique sense per word type with no hypernyms. Since the POS tag for each word is included in the dataset, we exclude WordNet synsets which are incompatible with the POS tag. The synset embedding parameters are initialized using the synset vectors obtained by running AutoExtend (Rothe and Schütze, 2015) on 100-dimensional GloVe (Pennington et al., 2014) vectors for WordNet 3.1. We refer to this embedding as *GloVe-extended*. Representation for the OOV word types in LSTM-PP and OOV synset types in *OntoLSTM-PP* were randomly drawn from a uniform 100-d distribution. Initial sense prior parameters (λ_w) were also drawn from a uniform 1-d distribution.

Baselines. In our experiments, we compare our proposed model, *OntoLSTM-PP* with three baselines – *LSTM-PP* initialized with GloVe embedding, *LSTM-PP* initialized with GloVe vectors retrofitted to WordNet using the approach of Faruqui et al. (2015) (henceforth referred to as *GloVe-retro*), and finally the best performing standalone PP attachment system from Belinkov et al. (2014), referred to as *HPCD (full)* in the paper. *HPCD (full)* is a neural network model that learns to compose the vector representations of each of the candidate heads with those of the preposition and the dependent, and predict attachments. The input representations are enriched using syntactic context information, POS, WordNet and VerbNet (Kipper et al., 2008) information and the distance of the head word from the PP is explicitly encoded in composition architecture. In contrast, we do not use syntactic context, VerbNet and distance information, and do not explicitly encode POS information.

4.1 PP Attachment Results

Table 1 shows that our proposed token level embedding scheme *OntoLSTM-PP* outperforms the better variant of our baseline *LSTM-PP* (with GloVe-retro intialization) by an absolute accuracy difference of 4.9%, or a relative error reduction of 32%. *OntoLSTM-PP* also outperforms *HPCD (full)*, the previous best result on this dataset.

Initializing the word embeddings with *GloVe-retro* (which uses WordNet as described in Faruqui et al. (2015)) instead of GloVe amounts to a small improvement, compared to the improvements obtained using *OntoLSTM-PP*. This result illustrates that our approach of dynamically choosing a context sensitive distribution over synsets is a more effective way of making use of WordNet.

Effect on dependency parsing. Following Belinkov et al. (2014), we used RBG parser (Lei et al., 2014), and modified it by adding a binary feature indicating the PP attachment predictions from our model.

We compare four ways to compute the additional binary features: 1) the predictions of the best standalone system *HPCD (full)* in Belinkov et al. (2014), 2) the predictions of our baseline model *LSTM-PP*, 3) the predictions of our improved model *OntoLSTM-PP*, and 4) the gold labels *Oracle PP*.

Table 2 shows the effect of using the PP attachment predictions as features within a dependency parser. We note there is a relatively small difference in unlabeled attachment accuracy for all dependencies (not only PP attachments), even when gold PP attachments are used as additional features to the parser. However, when gold PP attachment are used, we note a large potential improve-

System	Initialization	Embedding	Resources	Test Acc.
HPCD (full)	Syntactic-SG	Type	WordNet, VerbNet	88.7
LSTM-PP	GloVe	Type	-	84.3
LSTM-PP	GloVe-retro	Type	WordNet	84.8
OntoLSTM-PP	GloVe-extended	Token	WordNet	**89.7**

Table 1: Results on Belinkov et al. (2014)'s PPA test set. *HPCD (full)* is from the original paper, and it uses syntactic SkipGram. *GloVe-retro* is GloVe vectors retrofitted (Faruqui et al., 2015) to WordNet 3.1, and *GloVe-extended* refers to the synset embeddings obtained by running AutoExtend (Rothe and Schütze, 2015) on GloVe.

System	Full UAS	PPA Acc.
RBG	94.17	88.51
RBG + HPCD (full)	94.19	89.59
RBG + LSTM-PP	94.14	86.35
RBG + OntoLSTM-PP	94.30	90.11
RBG + Oracle PP	94.60	98.97

Table 2: Results from RBG dependency parser with features coming from various PP attachment predictors and oracle attachments.

ment of 10.46 points in PP attachment accuracies (between the PPA accuracy for *RBG* and *RBG + Oracle PP*), which confirms that adding PP predictions as features is an effective approach. Our proposed model *RBG + OntoLSTM-PP* recovers 15% of this potential improvement, while *RBG + HPCD (full)* recovers 10%, which illustrates that PP attachment remains a difficult problem with plenty of room for improvements even when using a dedicated model to predict PP attachments and using its predictions in a dependency parser.

We also note that, although we use the same predictions of the *HPCD (full)* model in Belinkov et al. (2014)[5], we report different results than Belinkov et al. (2014). For example, the unlabeled attachment score (UAS) of the baselines *RBG* and *RBG + HPCD (full)* are 94.17 and 94.19, respectively, in Table 2, compared to 93.96 and 94.05, respectively, in Belinkov et al. (2014). This is due to the use of different versions of the RBG parser.[6]

4.2 Analysis

In this subsection, we analyze different aspects of our model in order to develop a better understanding of its behavior.

Effect of context sensitivity and sense priors. We now show some results that indicate the relative strengths of two components of our context-sensitive token embedding model. The second row in Table 3 shows the test accuracy of a system trained without sense priors (that is, making $p(s|w_i)$ from Eq. 1 a uniform distribution), and the third row shows the effect of making the token representations context-insensitive by giving a similar attention score to all related concepts, essentially making them type level representations, but still grounded in WordNet. As it can be seen, removing context sensitivity has an adverse effect on the results. This illustrates the importance of the sense priors and the attention mechanism.

It is interesting that, even without sense priors and attention, the results with WordNet grounding is still higher than that of the two LSTM-PP systems in Table 1. This result illustrates the regularization behavior of sharing concept embeddings across multiple words, which is especially important for rare words.

Effect of training data size. Since *OntoLSTM-PP* uses external information, the gap between the model and *LSTM-PP* is expected to be more pronounced when the training data sizes are smaller. To test this hypothesis, we trained the two models with different amounts of training data and measured their accuracies on the test set. The plot is shown in Figure 4. As expected, the gap tends to be larger at smaller data sizes. Surprisingly, even with 2000 sentences in the training data set, *OntoLSTM-PP* outperforms *LSTM-PP* trained with the full data set. When both the models are trained with the full dataset, *LSTM-PP* reaches a training accuracy of 95.3%, whereas *OntoLSTM-PP* reaches 93.5%. The fact that *LSTM-PP* is overfitting the training data more, indicates the regular-

[5]The authors kindly provided their predictions for 1942 test examples (out of 1951 examples in the full test set). In Table 2, we use the same subset of 1942 test examples and will include a link to the subset in the final draft.

[6]We use the latest commit (SHA: e07f74) on the GitHub repository of the RGB parser.

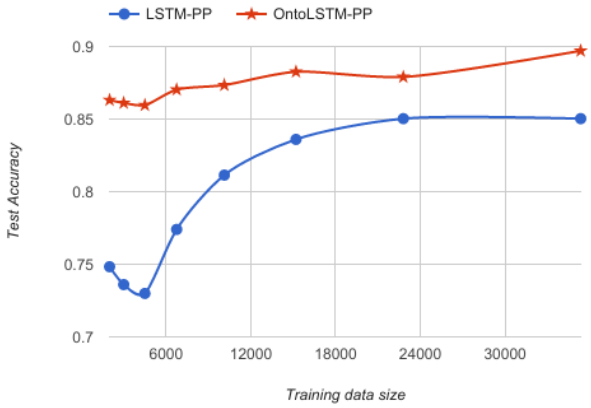


Figure 4: Effect of training data size on test accuracies of OntoLSTM-PP and LSTM-PP.

Model	PPA Acc.
full	89.7
- sense priors	88.4
- attention	87.5

Table 3: Effect of removing sense priors and context sensitivity (attention) from the model.

ization capability of *OntoLSTM-PP*.

Qualitative analysis. To better understand the effect of WordNet grounding, we took a sample of 100 sentences from the test set whose PP attachments were correctly predicted by OntoLSTM-PP but not by LSTM-PP. A common pattern observed was that those sentences contained words not seen frequently in the training data. Figure 5 shows two such cases. In both cases, the weights assigned by OntoLSTM-PP to infrequent words are also shown. The word types *soapsuds* and *buoyancy* do not occur in the training data, but OntoLSTM-PP was able to leverage the parameters learned for the synsets that contributed to their token representations. Another important observation is that the word type *buoyancy* has four senses in WordNet (we consider the first three), none of which is the metaphorical sense that is applicable to *markets* as shown in the example here. Selecting a combination of relevant hypernyms from various senses may have helped OntoLSTM-PP make the right prediction. This shows the value of using hypernymy information from WordNet. Moreover, this indicates the strength of the hybrid nature of the model, that lets it augment ontological information with distributional information.

Parameter space. We note that the vocabulary sizes in OntoLSTM-PP and LSTM-PP are comparable as the synset types are shared across word types. In our experiments with the full PP attachment dataset, we learned embeddings for 18k synset types with OntoLSTM-PP and 11k word types with LSTM-PP. Since the biggest contribution to the parameter space comes from the embedding layer, the complexities of both the models are comparable.

5 Related Work

This work is related to various lines of research within the NLP community: dealing with synonymy and homonymy in word representations both in the context of distributed embeddings and more traditional vector spaces; hybrid models of distributional and knowledge based semantics; and selectional preferences and their relation with syntactic and semantic relations.

The need for going beyond a single vector per word-type has been well established for a while, and many efforts were focused on building multi-prototype vector space models of meaning (Reisinger and Mooney, 2010; Huang et al., 2012; Chen et al., 2014; Jauhar et al., 2015; Neelakantan et al., 2015; Arora et al., 2016, etc.). However, the target of all these approaches is obtaining multi-sense word vector spaces, either by incorporating sense tagged information or other kinds of external context. The number of vectors learned is still fixed, based on the preset number of senses. In contrast, our focus is on learning a context dependent distribution over those concept representations. Other work not necessarily related to multi-sense vectors, but still related to our work includes Belanger and Kakade (2015)'s work which proposed a Gaussian linear dynamical system for estimating token-level word embeddings, and Vilnis and McCallum (2015)'s work which proposed mapping each word type to a density instead of a point in a space to account for uncertainty in meaning. These approaches do not make use of lexical ontologies and are not amenable for joint training with a downstream NLP task.

Related to the idea of concept embeddings is Rothe and Schütze (2015) who estimated WordNet synset representations, given pre-trained type-level word embeddings. In contrast, our work focuses on estimating token-level word embeddings as context sensitive distributions of concept em-

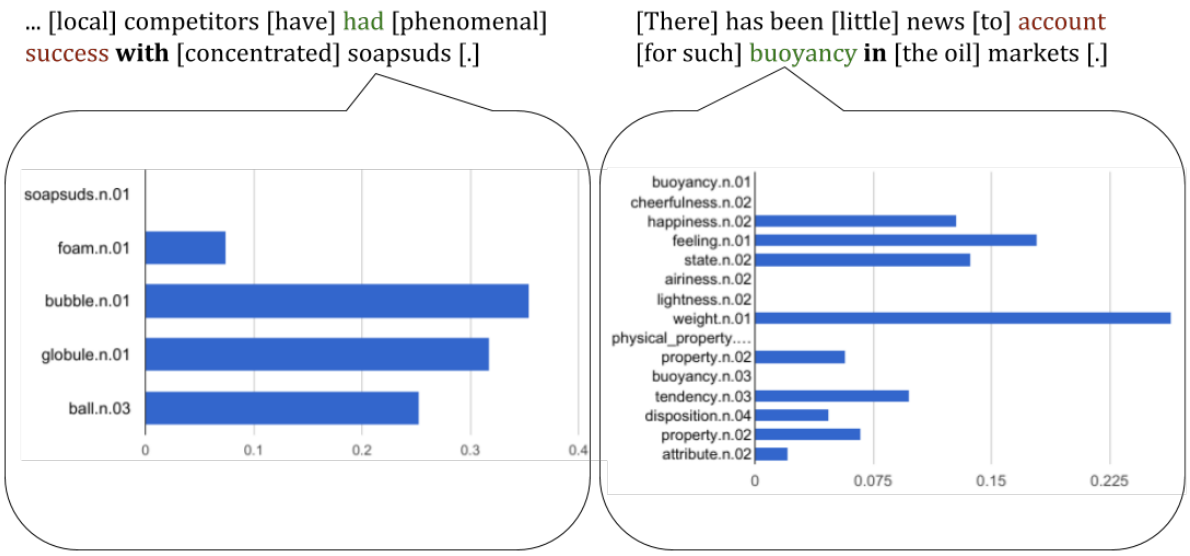


Figure 5: Two examples from the test set where OntoLSTM-PP predicts the head correctly and LSTM-PP does not, along with weights by OntoLSTM-PP to synsets that contribute to token representations of infrequent word types. The prepositions are shown in bold, LSTM-PP's predictions in red and OntoLSTM-PP's predictions in green. Words that are not candidate heads or dependents are shown in brackets.

beddings.

There is a large body of work that tried to improve word embeddings using external resources. Yu and Dredze (2014) extended the CBOW model (Mikolov et al., 2013) by adding an extra term in the training objective for generating words conditioned on similar words according to a lexicon. Jauhar et al. (2015) extended the skipgram model (Mikolov et al., 2013) by representing word senses as latent variables in the generation process, and used a structured prior based on the ontology. Faruqui et al. (2015) used belief propagation to update pre-trained word embeddings on a graph that encodes lexical relationships in the ontology. Similarly, Johansson and Pina (2015) improved word embeddings by representing each sense of the word in a way that reflects the topology of the semantic network they belong to, and then representing the words as convex combinations of their senses. In contrast to previous work that was aimed at improving *type level* word representations, we propose an approach for obtaining *context-sensitive* embeddings at the *token level*, while jointly optimizing the model parameters for the NLP task of interest.

Resnik (1993) showed the applicability of semantic classes and selectional preferences to resolving syntactic ambiguity. Zapirain et al. (2013) applied models of selectional preferences automatically learned from WordNet and distributional information, to the problem of semantic role labeling. Resnik (1993); Brill and Resnik (1994); Agirre (2008) and others have used WordNet information towards improving prepositional phrase attachment predictions.

6 Conclusion

In this paper, we proposed a grounding of lexical items which acknowledges the semantic ambiguity of word types using WordNet and a method to learn a context-sensitive distribution over their representations. We also showed how to integrate the proposed representation with recurrent neural networks for disambiguating prepositional phrase attachments, showing that the proposed WordNet-grounded context-sensitive token embeddings outperforms standard type-level embeddings for predicting PP attachments. We provided a detailed qualitative and quantitative analysis of the proposed model.

Implementation and code availability. The models are implemented using Keras (Chollet, 2015), and the functionality is available at `https://github.com/pdasigi/onto-lstm` in the form of Keras layers to make it easier to use the proposed embedding model in other NLP problems.

Future work. This approach may be extended to other NLP tasks that can benefit from using encoders that can access WordNet information. WordNet also has some drawbacks, and may not always have sufficient coverage given the task at hand. As we have shown in §4.2, our model can deal with missing WordNet information by augmenting it with distributional information. Moreover, the methods described in this paper can be extended to other kinds of structured knowledge sources like Freebase which may be more suitable for tasks like question answering.

Acknowledgements

The first author is supported by a fellowship from the Allen Institute for Artificial Intelligence. We would like to thank Matt Gardner, Jayant Krishnamurthy, Julia Hockenmaier, Oren Etzioni, Hector Liu, Filip Ilievski, and anonymous reviewers for their comments.

References

Eneko Agirre. 2008. Improving parsing and pp attachment performance with sense information. In *ACL*. Citeseer.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. Linear algebraic structure of word senses, with applications to polysemy. *arXiv preprint arXiv:1601.03764* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

David Belanger and Sham M. Kakade. 2015. A linear dynamical system model for text. In *ICML*.

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. 2014. Exploring compositional architectures and word vector representations for prepositional phrase attachment. *Transactions of the Association for Computational Linguistics* 2:561–572.

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 1198–1204.

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

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*. pages 1025–1035.

François Chollet. 2015. Keras. https://github.com/fchollet/keras.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, pages 873–882.

Sujay Kumar Jauhar, Chris Dyer, and Eduard H. Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *NAACL*.

Richard Johansson and Luis Nieto Pina. 2015. Embedding a semantic network in a word space. In *In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics–Human Language Technologies*. Citeseer.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation* 42(1):21–40.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*.

Tao Lei, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *ACL*. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR* abs/1310.4546.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the workshop on Human Language Technology*.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *HLT-ACL*.

Philip Resnik. 1993. Semantic classes and syntactic ambiguity. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *ACL*.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *ICLR*.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *ACL*.

Benat Zapirain, Eneko Agirre, Lluis Marquez, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics* .

Identifying 1950s American Jazz Musicians: Fine-Grained IsA Extraction via Modifier Composition

Ellie Pavlick*
University of Pennsylvania
3330 Walnut Street
Philadelphia, Pennsylvania 19104
epavlick@seas.upenn.edu

Marius Paşca
Google Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
mars@google.com

Abstract

We present a method for populating fine-grained classes (e.g., *"1950s American jazz musicians"*) with instances (e.g., *Charles Mingus*). While state-of-the-art methods tend to treat class labels as single lexical units, the proposed method considers each of the individual modifiers in the class label relative to the head. An evaluation on the task of reconstructing Wikipedia category pages demonstrates a >10 point increase in AUC, over a strong baseline relying on widely-used Hearst patterns.

1 Introduction

The majority of approaches (Snow et al., 2006; Shwartz et al., 2016) for extracting IsA relations from text rely on lexical patterns as the primary signal of whether an instance belongs to a class. For example, observing a pattern like *"X such as Y"* is a strong indication that *Y* (e.g., *"Charles Mingus"*) is an instance of class *X* (e.g., *"musician"*) (Hearst, 1992).

Methods based on these "Hearst patterns" assume that class labels can be treated as atomic lexicalized units. This assumption has several significant weakness. First, in order to recognize an instance of a class, these pattern-based methods require that the entire class label be observed verbatim in text. The requirement is reasonable for class labels containing a single word, but in practice, there are many possible fine-grained classes: not only *"musicians"* but also *"1950s American jazz musicians"*. The probability that a given label will appear in its entirety within one of the expected patterns is very low, even in large amounts of text. Second, when class labels are treated as though they cannot be decomposed, every class label must be modeled independently, even those containing overlapping words (*"American jazz musician"*, *"French jazz musician"*). As a result, the number of meaning representations to be learned is exponential in the length of the class label, and quickly becomes intractable. Thus, compositional models of taxonomic relations are necessary for better language understanding.

1950s American jazz musicians
...seminal musicians *such as* **Charles Mingus** and George Russell...
...A virtuoso bassist and composer, **Mingus** irrevocably *changed the face of* jazz...
...**Mingus** truly *was a product of* America in all its historic complexities...
...**Mingus** *dominated the scene back in* the 1950s and 1960s...

Figure 1: We extract instances of fine-grained classes by considering each of the modifiers in the class label individually. This allows us to extract instances even when the full class label never appears in text.

We introduce a compositional approach for reasoning about fine-grained class labels. Our approach is based on the notion from formal semantics, in which modifiers (*"1950s"*) correspond to properties that differentiate instances of a subclass (*"1950s musicians"*) from instances of the superclass (*"musicians"*) (Heim and Kratzer, 1998). Our method consists of two stages: interpreting each modifier relative to the head (*"musicians active during 1950s"*), and using the interpretations to identify instances of the class from text (Figure 1). Our main contributions are: 1) a compositional method for IsA extraction, which in-

*Contributed during an internship at Google.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2099–2109
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1192

volves a novel application of noun-phrase paraphrasing methods to the task of semantic taxonomy induction and 2) the operationalization of a formal semantics framework to address two aspects of semantics that are often kept separate in NLP: assigning intrinsic "meaning" to a phrase, and reasoning about that phrase in a truth-theoretic context.

2 Related Work

Noun Phrase Interpretation. Compound noun phrases (*"jazz musician"*) communicate implicit semantic relations between modifiers and the head. Many efforts to provide semantic interpretations of such phrases rely on matching the compound to pre-defined patterns or semantic ontologies (Fares et al., 2015; Ó Séaghdha and Copestake, 2007; Tratz and Hovy, 2010; Surtani and Paul, 2015; Choi et al., 2015). Recently, interpretations may take the form of arbitrary natural language predicates (Hendrickx et al., 2013). Most approaches are supervised, comparing unseen noun compounds to the most similar phrase seen in training (Wijaya and Gianfortoni, 2011; Nulty and Costello, 2013; Van de Cruys et al., 2013). Other unsupervised approaches apply information extraction techniques to paraphrase noun compounds (Kim and Nakov, 2011; Xavier and Strube de Lima, 2014; Paşca, 2015). They focus exclusively on providing good paraphrases for an input noun compound. To our knowledge, ours is the first attempt to use these interpretations for the downstream task of IsA relation extraction.

IsA Relation Extraction. Most efforts to acquire taxonomic relations from text build on the seminal work of Hearst (1992), which observes that certain textual patterns–e.g., *"X and other Y"*–are high-precision indicators of whether X is a member of class Y. Recent work focuses on learning such patterns automatically from corpora (Snow et al., 2006; Shwartz et al., 2016). These IsA extraction techniques provide a key step for the more general task of knowledge base population. The "universal schema" approach (Riedel et al., 2013; Kirschnick et al., 2016; Verga et al., 2017), which infers relations using matrix factorization, often includes Hearst patterns as input features. Graphical (Bansal et al., 2014) and joint inference models (Movshovitz-Attias and Cohen, 2015) typically require Hearst patterns to define an inventory of possible classes. A separate line of work avoids Hearst patterns by instead exploiting semi-structured data from HTML markup (Wang and Cohen, 2009; Dalvi et al., 2012; Pasupat and Liang, 2014). These approaches all share the limitation that, in practice, in order for a class to be populated with instances, the entire class label has to have been observed verbatim in text. This requirement limits the ability to handle arbitrarily fine-grained classes. Our work addresses this limitation by modeling fine-grained class labels compositionally. Thus the proposed method can combine evidence from multiple sentences, and can perform IsA extraction without requiring any example instances of a given class.[1]

Taxonomy Construction. Previous work on the construction of a taxonomy of IsA relations (Flati et al., 2014; de Melo and Weikum, 2010; Kozareva and Hovy, 2010; Ponzetto and Strube, 2007; Ponzetto and Navigli, 2009) considers that task to be different than extracting a flat set of IsA relations from text in practice. Challenges specific to taxonomy construction include overall concept positioning and how to discover whether concepts are unrelated, subordinated or parallel to each other (Kozareva and Hovy, 2010); the need to refine and enrich the taxonomy (Flati et al., 2014); the difficulty in adding relevant IsA relations towards the top of the taxonomy (Ponzetto and Navigli, 2009); eliminating cycles and inconsistencies (Ponzetto and Navigli, 2009; Kozareva and Hovy, 2010). For practical purposes, these challenges are irrelevant when extracting flat IsA relations. Whereas Flati et al. (2014); Bizer et al. (2009); de Melo and Weikum (2010); Nastase and Strube (2013); Ponzetto and Strube (2007); Ponzetto and Navigli (2009); Hoffart et al. (2013) rely on data within human-curated resources, our work operates over unstructured text. Resources constructed in Bizer et al. (2009); Nastase and Strube (2013); Hoffart et al. (2013) contain not just a taxonomy of IsA relations,

[1] Pasupat and Liang (2014) also focuses on zero-shot IsA extraction, but exploits HTML document structure, rather than reasoning compositionally.

but also relation types other than IsA.

3 Modifiers as Functions

Formalization. In formal semantics, modification is modeled as function application. Specifically, let MH be a class label consisting of a head H, which we assume to be a common noun, preceded by a modifier M. We use $[\![\cdot]\!]$ to represent the "interpretation function" that maps a linguistic expression to its denotation in the world. The interpretation of a common noun is the set of entities[2] in the universe $\mathcal{U}$. They are denoted by the noun (Heim and Kratzer, 1998):

$$[\![H]\!] = \{e \in \mathcal{U} \mid e \text{ is a } H\} \quad (1)$$

The interpretation of a modifier M is a function that maps between sets of entities. That is, modifiers select a subset[3] of the input set:

$$[\![M]\!](H) = \{e \in H \mid e \text{ satisfies } M\} \quad (2)$$

This formalization leaves open how one decides whether or not "e satisfies M". This nontrivial, as the meaning of a modifier can vary depending on the class it is modifying: if e is a *"good student"*, e is not necessarily a *"good person"*, making it difficult to model whether "e satisfies *good*" in general. We therefore reframe the above equation, so that the decision of whether "e satisfies M" is made by calling a binary function ϕ_M, parameterized by the class H within which e is being considered:

$$[\![M]\!](H) = \{e \in H \mid \phi_M(H, e)\} \quad (3)$$

Conceptually, ϕ_M captures the core "meaning" of the modifier M, which is the set of properties that differentiate members of the output class MH from members of the more general input class H. This formal semantics framework has two important consequences. First, the modifier has an intrinsic "meaning". The properties entailed by the modifier are independent of the particular state of the world. This makes it possible to make inferences about *"1950s musician"* even if no 1950s musician have been observed. Second, the modifier is a function that can be applied in a truth-theoretic setting. That is, applying *"1950s"* to the set of *"musicians"* returns exactly the set of *"1950s musicians"*.

Computational Approaches. While the notion of modifiers as functions has been incorporated into computational models previously, prior work focuses on either assigning an intrinsic meaning to M or on operationalizing M in a truth-theoretic sense, but not on doing both simultaneously. For example, Young et al. (2014) focuses exclusively on the subset selection aspect of modification. That is, given a set of instances H and a modifier M, their method could return the subset MH. However, their method does not model the meaning of the modifier itself, so that, e.g., if there were no red cars in their model of the world, the phrase *"red cars"* would have no meaning. In contrast, Baroni and Zamparelli (2010) models the meaning of modifiers explicitly as functions that map between vector-space representations of nouns. However, their model focuses on similarity between class labels–e.g., to say that *"important routes"* is similar to *"major roads"*–and it is not obvious how the method could be operationalized in order to identify instances of those classes. A contribution of our work is to model the semantics of M intrinsically, but in a way that permits application in the model theoretic setting. We learn an explicit model of the "meaning" of a modifier M relative to a head H, represented as a distribution over properties that differentiate the members of the class MH from those of the class H. We then use this representation to identify the subset of instances of H, which constitute the subclass MH.

4 Learning Modifier Interpretations

4.1 Setup

For each modifier M, we would like to learn the function ϕ_M from Eq. 3. Doing so makes it possible, given H and an instance $e \in H$, to decide whether e has the properties required to be an instance of MH. In general, there is no systematic way to determine the implied relation between M and H, as modifiers can arguably express any semantic relation, given the right context (Weiskopf,

[2] We use "entities" and "instances" interchangeably;"entities" is standard terminology in linguistics.

[3] As does virtually all previous work in information extraction, we assume that modifiers are *subsective*, acknowledging the limitations (Kamp and Partee, 1995).

2007). We therefore model the semantic relation between M and H as a distribution over properties that could potentially define the subclass $MH \subseteq H$. We will refer to this distribution as a "property profile" for M relative to H. We make the assumption that relations between M and H that are discussed more often are more likely to capture the important properties of the subclass MH. This assumption is not perfect (Section 4.4) but has given good results for paraphrasing noun phrases (Nakov and Hearst, 2013; Paşca, 2015). Our method for learning property profiles is based on the unsupervised method proposed by Paşca (2015), which uses query logs as a source of common sense knowledge, and rewrites noun compounds by matching MH (*"American musicians"*) to queries of the form "$H(.*)M$" (*"musicians from America"*).

4.2 Inputs

We assume two inputs: 1) an IsA repository, $\mathcal{O}$, containing $\langle e, C\rangle$ tuples where C is a category and e is an instance of C, and 2) a fact repository, $\mathcal{D}$, containing $\langle s, p, o, w\rangle$ tuples where s and o are noun phrases, p is a predicate, and w is a confidence that p expresses a true relation between s and o. Both $\mathcal{O}$ and $\mathcal{D}$ are extracted from a sample of around 1 billion Web documents in English. The supplementary material gives additional details.

We instantiate $\mathcal{O}$ with an IsA repository constructed by applying Hearst patterns to the Web documents. Instances are represented as automatically-disambiguated entity mentions[4] which, when possible, are resolved to Wikipedia pages. Classes are represented as (non-disambiguated) natural language strings. We instantiate $\mathcal{D}$ with a large repository of facts extracted using in-house implementations of ReVerb (Fader et al., 2011) and OLLIE (Mausam et al., 2012). The predicates are extracted as natural language strings. Subjects and objects may be either disambiguated entity references or natural language strings. Every tuple is included in both the forward and the reverse direction. E.g. $\langle$*jazz, perform at, venue*$\rangle$ also appears as $\langle$*venue*, $\leftarrow$*perform at, jazz*$\rangle$, where $\leftarrow$ is a special character signifying inverted predicates. These inverted predicates simplify the following definitions. In total, $\mathcal{O}$ contains 1.1M tuples and $\mathcal{D}$ contains 30M tuples.

4.3 Building Property Profiles

Properties. Let I be a function that takes as input a noun phrase MH and returns a property profile for M relative to H. We define a "property" to be a tuple of a subject, predicate and object in which the subject position[5] is a wildcard, e.g. $\langle *, born\ in, America\rangle$. Any instance that fills the wildcard slot then "has" the property. We expand adjectival modifiers to encompass nominalized forms using a nominalization dictionary extracted from WordNet (Miller, 1995). If MH is *"American musician"* and we require a tuple to have the form $\langle H, p, M, w\rangle$, we will include tuples in which the third element is either *"American"* or *"America"*.

Relating M to H Directly. We first build property profiles by taking the predicate and object from any tuple in $\mathcal{D}$ in which the subject is the head and the object is the modifier:

$$I_1(MH) = \{\langle\langle p, M\rangle, w\rangle \mid \langle H, p, M, w\rangle \in \mathcal{D}\} \quad (4)$$

Relating M to an Instance of H. We also consider an extension in which, rather than requiring the subject to be the class label H, we require the subject to be an instance of H.

$$I_2(MH) = \{\langle\langle p, M\rangle, w\rangle \mid \langle e, H\rangle \in \mathcal{O} \wedge \langle e, p, M, w\rangle \in \mathcal{D}\} \quad (5)$$

Modifier Expansion. In practice, when building property profiles, we do not require that the object of the fact tuple match the modifier exactly, as suggested in Eq. 4 and 5. Instead, we follow Paşca (2015) and take advantage of facts involving distributionally similar modifiers. Specifically, rather than looking only at tuples in $\mathcal{D}$ in which the object matches M, we consider all tuples, but discount the weight proportionally to the similarity between M and the object of the tuple.

[4] "Entity mentions" may be individuals, like *"Barack Obama"*, but may also be concepts like *"jazz"*.

[5] Inverse predicates capture properties in which the wildcard is conceptually the object of the relation, but occupies the subject slot in the tuple. For example, $\langle$*venue*, $\leftarrow$*perform at, jazz*$\rangle$ captures that a *"jazz venue"* is a *"venue"* e such that *"jazz performed at e"*.

Good Property Profiles				Bad Property Profiles	
rice dish	French violinist	Led Zeppelin song	still life painter	child actor	risk manager
* serve with rice	* live in France	Led Zeppelin write *	* known for still life	* have child	* take risk
* include rice	* born in France	Led Zeppelin play *	* paint still life	* expect child	* be at risk
* consist of rice	* speak French	Led Zeppelin have *	still life be by *	* play child	* be aware of risk

Table 1: Example property profiles learned by observing predicates that relate instances of class H to modifier M (I_2). Results are similar when using the class label H directly (I_1). We spell out inverted predicates (Section 4.2) so wildcards (*) may appear as subjects or objects.

Thus, I_1 is computed as below:

$$I_1(MH) = \{\langle\langle p, M\rangle, w \times sim(M, N)\rangle \mid \langle H, p, N, w\rangle \in \mathcal{D}\} \quad (6)$$

where $sim(M, N)$ is the cosine similarity between M and N. I_2 is computed analogously. We compute sim using a vector space built from Web documents following Lin and Wu (2009); Pantel et al. (2009). We retain the 100 most similar phrases for each of ∼10M phrases, and consider all other similarities to be 0.

4.4 Analysis of Property Profiles

Table 1 provides examples of good and bad property profiles for several MHs. In general, frequent relations between M and H capture relevant properties of MH, but it is not always the case. To illustrate, the most frequently discussed relation between *"child"* and *"actor"* is that actors have children, but this property is not indicative of the meaning of *"child actor"*. Qualitatively, the top-ranked interpretations learned by using the head noun directly (I_1, Eq. 4) are very similar to those learned using instances of the head (I_2, Eq. 5). However, I_2 returns many more properties (10 on average per MH) than I_1 (just over 1 on average). Anecdotally, we see that I_2 captures more specific relations than does I_1. For example, for *"jazz musicians"*, both methods return *"* write jazz"* and *"* compose jazz"*, but I_2 additionally returns properties like *"* be major creative influence in jazz"*. We compare I_1 and I_2 quantitatively in Section 6. Importantly, we do see that both I_1 and I_2 are capable of learning head-specific property profiles for a modifier. Table 2 provides examples.

Class Label	Property Profile
American company	* based in America
American composer	* born in America
American novel	* written in America
jazz album	* features jazz
jazz composer	* writes jazz
jazz venue	jazz performed at *

Table 2: Head-specific property profiles learned by relating instances of H to the modifier M (I_2). Results are similar using I_1.

5 Class-Instance Identification

Instance finding. After finding properties that relate a modifier to a head, we turn to the task of identifying instances of fine-grained classes. That is, for a given modifier M, we want to instantiate the function ϕ_M from Eq. 3. In practice, rather than being a binary function that decides whether or not e is in class MH, our instantiation, $\hat{\phi}_M$, will return a real-valued score expressing the confidence that e is a member of MH. For notational convenience, let $\mathcal{D}(\langle s, p, o\rangle) = w$, if $\langle s, p, o, w\rangle \in \mathcal{D}$ and 0 otherwise. We define $\hat{\phi}_M$ as follows:

$$\hat{\phi}_M(H, e) = \sum_{\langle\langle p,o\rangle,\omega\rangle \in I(MH)} \omega \times \mathcal{D}(\langle e, p, o\rangle) \quad (7)$$

Applying M to H, then, is as in Eq. 3 except that instead of a discrete set, it returns a scored list of candidate instances:

$$[\![M]\!](H) = \{\langle e, \hat{\phi}_M(H, e)\rangle \mid \langle e, H\rangle \in \mathcal{O}\} \quad (8)$$

Ultimately, we need to identify instances of arbitrary class labels, which may contain multiple modifiers. Given a class label $C = M_1 \ldots M_k H$ that contains a head H preceded by modifiers $M_1 \ldots M_k$, we generate a list of candidate instances by finding all instances of H that have some property to support every modifier:

$$\bigcap_{i=1}^{k} \{\langle e, s(e)\rangle \mid \langle e, w\rangle \in [\![M_i]\!](H) \wedge w > 0\} \quad (9)$$

where $s(e)$ is the mean[6] of the scores assigned by each separate $\hat{\phi}_{M_i}$. From here on, we use **Mods** to refer to our method that generates lists of instances for a class using Eq. 8 and 9. When $\hat{\phi}_M$ (Eq. 7) is implemented using I_1, we use the name $\textbf{Mods}_H$ (for "heads"). When it is implemented using I_2, we use the name $\textbf{Mods}_I$ (for "instances").

Weakly Supervised Reranking. Eq. 8 uses a naive ranking in which the weight for $e \in MH$ is the product of how often e has been observed with some property and the weight of that property for the class MH. Thus, instances of H with overall higher counts in $\mathcal{D}$ receive high weights for every MH. We therefore train a simple logistic regression model to predict the likelihood that e belongs to MH. We use a small set of features[7], including the raw weight as computed in Eq. 7. For training, we sample $\langle e, C \rangle$ pairs from our IsA repository $\mathcal{O}$ as positive examples and random pairs that were not extracted by any Hearst pattern as negative examples. We frame the task as a binary prediction of whether $e \in C$, and use the model's confidence as the value of $\hat{\phi}_M$ in place of the function in Eq. 7.

6 Evaluation

6.1 Experimental Setup

Evaluation Sets. We evaluate our models on their ability to return correct instances for arbitrary class labels. As a source of evaluation data, we use Wikipedia category pages (e.g., http://en.wikipedia.org/wiki/Category:Pakistani_film_actresses). These are pages in which the title is the name of the category (*"pakistani film actresses"*) and the body is a manually curated list of links to other pages that fall under the category. We measure the precision and recall of each method for discovering the instances listed on these pages given the page title (henceforth "class label").

We collect the titles of all Wikipedia category pages, removing those in which the last word is capitalized or which contain fewer than three words. These heuristics are intended to retain compositional titles in which the head is a single common noun. We also remove any titles that contain links to sub-categories. This is to favor fine-grained classes (*"pakistani film actresses"*) over coarse-grained ones (*"film actresses"*). We perform heuristic modifier chunking in order to group together multiword modifiers (e.g., *"puerto rican"*); for details, see supplementary material. From the resulting list of class labels, we draw two samples of 100 labels each, enforcing that no H appear as the head of more than three class labels per sample. The first sample is chosen uniformly at random (denoted **UniformSet**). The second (**WeightedSet**) is weighted so that the probability of drawing $M_1 \ldots M_k H$ is proportional to the total number of class labels in which H appears as the head. These different evaluation sets[8] are intended to evaluate performance on the head versus the tail of class label distribution, since information retrieval methods often perform differently on different parts of the distribution. On average, there are 17 instances per category in UniformSet and 19 in WeightedSet. Table 3 gives examples of class labels.

Evaluation Set: Examples of Class Labels
UniformSet: 2008 california wildfires · australian army chaplains · australian boy bands · canadian military nurses · canberra urban places · cellular automaton rules · chinese rice dishes · coldplay concert tours · daniel libeskind designs · economic stimulus programs · german film critics · invasive amphibian species · latin political phrases · log flume rides · malayalam short stories · pakistani film actresses · puerto rican sculptors · string theory books
WeightedSet: ancient greek physicists · art deco sculptors · audio engineering schools · ballet training methods · bally pinball machines · british rhythmic gymnasts · calgary flames owners · canadian rock climbers · canon l-series lenses · emi classics artists · free password managers · georgetown university publications · grapefruit league venues · liz claiborne subsidiaries · miss usa 2000 delegates · new zealand illustrators · russian art critics

Table 3: Examples of class labels from evaluation sets.

Baselines. We implement two baselines using our IsA repository ($\mathcal{O}$ as defined in Section 4.1). Our simplest baseline ignores modifiers altogether, and simply assumes that any instance of H is an instance of MH, regardless of M. In this case the confidence value for

[6] Also tried minimum, but mean gave better results.
[7] Feature templates in supplementary material.
[8] Available at http://www.seas.upenn.edu/~nlp/resources/finegrained-class-eval.gz

$\langle e, MH \rangle$ is equivalent to that for $\langle e, H \rangle$. We refer to this baseline simply as **Baseline**. Our second, stronger baseline uses the IsA repository directly to identify instances of the fine-grained class $C = M_1 \dots M_k H$. That is, we consider e to be an instance of the class if $\langle e, C \rangle \in \mathcal{O}$, meaning the entire class label appeared in a source sentence matching some Hearst pattern. We refer to this baseline as **Hearst**. The weight used to rank the candidate instances is the confidence value assigned by the Hearst pattern extraction (Section 4.2).

Compositional Models. As a baseline compositional model, we augment the Hearst baseline via set intersection. Specifically, for a class $C = M_1 \dots M_k H$, if each of the $M_i H$ appears in $\mathcal{O}$ independently, we take the instances of C to be the intersection of the instances of each of the $M_i H$. We assign the weight of an instance e to be the sum of the weights associated with each independent modifier. We refer to this method as **Hearst**$\cap$. It is roughly equivalent to (Paşca, 2014). We contrast it with our proposed model, which recognizes instances of a fine-grained class by 1) assigning a meaning to each modifier in the form of a property profile and 2) checking whether a candidate instance exhibits these properties. We refer to the versions of our method as **Mods**$_H$ and **Mods**$_I$, as described in Section 5. When relevant, we use "raw" to refer to the version in which instances are ranked using raw weights and "RR" to refer to the version in which instances are ranked using logistic regression (Section 5). We also try using the proposed methods to extend rather than replace the Hearst baseline. We combine predictions by merging the ranked lists produced by each system: i.e. the score of an instance is the inverse of the sum of its ranks in each of the input lists. If an instance does not appear at all in an input list, its rank in that list is set to a large constant value. We refer to these combination systems as **Hearst+Mods**$_H$ and **Hearst+Mods**$_I$.

6.2 Results

Precision and Coverage. We first compare the methods in terms of their coverage, the number of class labels for which the method is able to find some instance, and their precision, to what extent the method is able to correctly rank true instances of the class above non-instances. We report total coverage, the number of labels for which the method returns any instance, and correct coverage, the number of labels for which the method returns a correct instance. For precision, we compute the average precision (AP) for each class label. AP ranges from 0 to 1, where 1 indicates that all positive instances were ranked above all negative instances. We report mean average precision (MAP), which is the mean of the APs across all the class labels. MAP is only computed over class labels for which the method returns something, meaning methods are not punished for returning empty lists.

Table 4 gives examples of instances returned for several class labels and Table 5 shows the precision and coverage for each of the methods. Figure 2 illustrates how the single mean AP score (as reported in Table 5) can misrepresent the relative precision of different methods. In combination, Table 5 and Figure 2 demonstrate that the proposed methods extract instances about as well as the baseline, whenever the baseline can extract anything at all; i.e. the proposed method does not cause a precision drop on classes covered by the baseline. In addition, there are many classes for which the baseline is not able to extract any instances, but the proposed method is. None of the methods can extract some of the gold instances, such as *"Dictator perpetuo"* and *"Furor Teutonicus"* of the gold class *"latin political phrases"*.

Table 5 also reveals that the reranking model (RR) consistently increases MAP for the proposed methods. Therefore, going forward, we only report results using the reranking model (i.e. Mods$_H$ and Mods$_I$ will refer to Mods$_H$ RR and Mods$_I$ RR, respectively).

Manual Re-Annotation. It possible that true instances of a class are missing from our Wikipedia reference set, and thus that our precision scores underestimate the actual precision of the systems. We therefore manually verify the top 10 predictions of each of the systems for a random sample of 25 class labels. We choose class labels for which Hearst was able to return at least one instance, in order to ensure reliable precision

Flemish still life painters: Clara Peeters · Willem Kalf · Jan Davidsz de Heem · *Pieter Claesz* · ~~Peter Paul Rubens~~ · Frans Snyders · Jan Brueghel the Elder · ~~Hans Memling~~ · ~~Pieter Bruegel the Elder~~ · ~~Caravaggio~~ · Abraham Brueghel
Pakistani cricket captains: Salman Butt · Shahid Afridi · Javed Miandad · Azhar Ali · ~~Greg Chappell~~ · Younis Khan · Wasim Akram · Imran Khan · Mohammad Hafeez · Rameez Raja · Abdul Hafeez Kardar · Waqar Younis · Sarfraz Ahmed
Thai buddhist temples: *Wat Buddhapadipa* · Wat Chayamangkalaram · *Wat Mongkolratanaram* · ~~Angkor Wat~~ · ~~Preah Vihear Temple~~ · Wat Phra Kaew · Wat Rong Khun · Wat Mahathat Yuwaratrangsarit · ~~Vat Phou~~ · Tiger Temple · Sanctuary of Truth · Wat Chalong · ~~Swayambhunath~~ · ~~Mahabodhi Temple~~ · Tiger Cave Temple · Harmandir Sahib

Table 4: Instances extracted for several fine-grained classes from Wikipedia. Lists shown are from Mods_I. Instances in italics were also returned by Hearst∩. Strikethrough denotes incorrect.

	UniformSet		WeightedSet	
	Coverage	MAP	Coverage	MAP
Baseline	95 / 70	0.01	98 / 74	0.01
Hearst	9 / 9	0.63	8 / 8	0.80
Hearst∩	13 / 12	0.62	9 / 9	0.80
Mods_H raw	56 / 32	0.23	50 / 30	0.16
Mods_H RR	56 / 32	0.29	50 / 30	0.25
Mods_I raw	62 / 36	0.18	59 / 38	0.20
Mods_I RR	62 / 36	0.24	59 / 38	0.23

Table 5: Coverage and precision for populating Wikipedia category pages with instances. "Coverage" is the number of class labels (out of 100) for which at least one instance was returned, followed by the number for which at least one correct instance was returned. "MAP" is mean average precision. MAP does not punish methods for returning empty lists, thus favoring the baseline (see Figure 2).

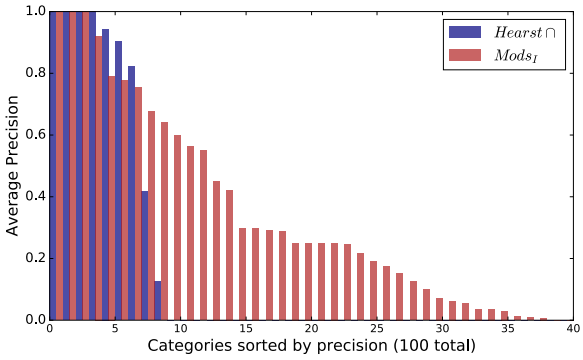


Figure 2: Distribution of AP over 100 class labels in WeightedSet. The proposed method (red) and the baseline method (blue) achieve high AP for the same number of classes, but Mods_I additionally finds instances for classes for which the baseline returns nothing.

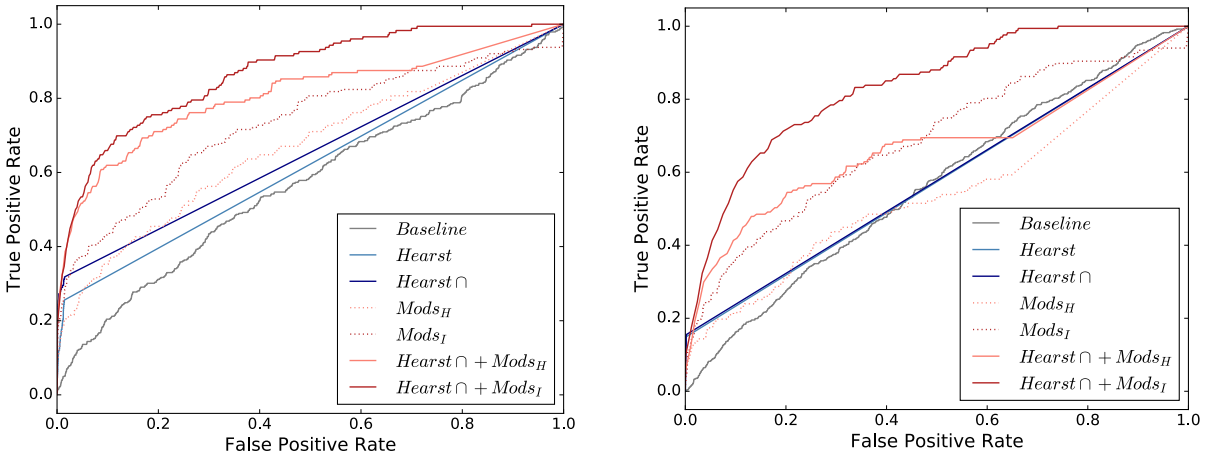
estimates. For each of these labels, we manually check the top 10 instances proposed by each method to determine whether each belongs to the class. Table 6 shows the precision scores for each method computed against the original Wikipedia list of instances and against our manually-augmented list of gold instances. The overall ordering of the systems does not change, but the precision scores increase notably after re-annotation. We continue to evaluate against the Wikipedia lists, but acknowledge that reported precision is likely an underestimate of true precision.

	Wikipedia	Gold
Hearst	0.56	0.79
Hearst∩	0.53	0.78
Mods_H	0.23	0.39
Mods_I	0.24	0.42
Hearst+Mods_H	0.43	0.63
Hearst+Mods_I	0.43	0.63

Table 6: P@10 before vs. after re-annotation; Wikipedia underestimates true precision.

	UniformSet		WeightedSet	
	AUC	Recall	AUC	Recall
Baseline	0.55	0.23	0.53	0.28
Hearst	0.56	0.03	0.52	0.02
Hearst∩	0.57	0.04	0.53	0.02
Mods_H	0.68	0.08	0.60	0.06
Mods_I	0.71	0.09	0.65	0.09
Hearst∩+Mods_H	0.70	0.09	0.61	0.08
Hearst∩+Mods_I	0.73	0.10	0.66	0.10

Table 7: Recall of instances on Wikipedia category pages, measured against the full set of instances from all pages in sample. AUC captures tradeoff between true and false positives.



(a) Uniform random sample (UniformSet).

(b) Weighted random sample (WeightedSet).

Figure 3: ROC curves for selected methods (Hearst in blue, proposed in red). Given a ranked list of instances, ROC curves plot true positives vs. false positives retained by setting various cutoffs. The curve becomes linear once all remaining instances have the same score (e.g., 0), as this makes it impossible to add true positives without also including all remaining false positives.

Precision-Recall Analysis. We next look at the precision-recall tradeoff in terms of the area under the curve (AUC) when each method attempts to rank the complete list of candidate instances. We take the union of all of the instances proposed by all of the methods (including the Baseline method which, given a class label $M_0 \ldots M_k H$, proposes every instance of the head H as a candidate). Then, for each method, we rank this full set of candidates such that any instance returned by the method is given the score the method assigns, and every other instance is scored as 0. Table 7 reports the AUC and recall. Figure 3 plots the full ROC curves. The requirement by Hearst that class labels appear in full in a single sentence results in very low recall, which translates into very low AUC when considering the full set of candidate instances. In comparison, the proposed compositional methods make use of a larger set of sentences, and provide non-zero scores for many more candidates, resulting in a >10 point increase in AUC on both UniformSet and WeightedSet (Table 7).

7 Conclusion

We have presented an approach to IsA extraction that takes advantage of the compositionality of natural language. Existing approaches often treat class labels as atomic units that must be observed in full in order to be populated with instances. As a result, current methods are not able to handle the infinite number of classes describable in natural language, most of which never appear in text. Our method reasons about each modifier in the label individually, in terms of the properties that it implies about the instances. This approach allows us to harness information that is spread across multiple sentences, significantly increasing the number of fine-grained classes that we are able to populate.

Acknowledgments

The paper incorporates suggestions on an earlier version from Susanne Riehemann. Ryan Doherty offered support in refining and accessing the fact repository used in the evaluation.

References

M. Bansal, D. Burkett, G. de Melo, and D. Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*. Baltimore, Maryland, pages 1041–1051.

M. Baroni and R. Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*. Cambridge, Massachusetts, pages 1183–1193.

C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. 2009. DBpedia - a crystallization point for the Web of data. *Journal of Web Semantics* 7(3):154–165.

E. Choi, T. Kwiatkowski, and L. Zettlemoyer. 2015. Scalable semantic parsing with partial ontologies. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*. Beijing, China, pages 1311–1320.

B. Dalvi, W. Cohen, and J. Callan. 2012. Websets: Extracting sets of entities from the Web using unsupervised information extraction. In *Proceedings of the 5th ACM Conference on Web Search and Data Mining (WSDM-12)*. Seattle, Washington, pages 243–252.

G. de Melo and G. Weikum. 2010. MENTA: Inducing multilingual taxonomies from Wikipedia. In *Proceedings of the 19th International Conference on Information and Knowledge Management (CIKM-10)*. Toronto, Canada, pages 1099–1108.

A. Fader, S. Soderland, and O. Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*. Edinburgh, Scotland, pages 1535–1545.

M. Fares, S. Oepen, and E. Velldal. 2015. Identifying compounds: On the role of syntax. In *International Workshop on Treebanks and Linguistic Theories (TLT-14)*. Warsaw, Poland, pages 273–283.

T. Flati, D. Vannella, T. Pasini, and R. Navigli. 2014. Two is bigger (and better) than one: the Wikipedia Bitaxonomy project. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*. Baltimore, Maryland, pages 945–955.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics (COLING-92)*. pages 539–545.

I. Heim and A. Kratzer. 1998. *Semantics in Generative Grammar*, volume 13. Blackwell Oxford.

I. Hendrickx, Z. Kozareva, P. Nakov, D. Ó Séaghdha, S. Szpakowicz, and T. Veale. 2013. SemEval-2013 task 4: Free paraphrases of noun compounds. In *Proceedings of Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval-13)*. pages 138–143.

J. Hoffart, F. Suchanek, K. Berberich, and G. Weikum. 2013. YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence Journal. Special Issue on Artificial Intelligence, Wikipedia and Semi-Structured Resources* 194:28–61.

H. Kamp and B. Partee. 1995. Prototype theory and compositionality. *Cognition* 57(2):129–191.

N. Kim and P. Nakov. 2011. Large-scale noun compound interpretation using bootstrapping and the Web as a corpus. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*. Edinburgh, Scotland, pages 648–658.

J. Kirschnick, H. Hemsen, and V. Markl. 2016. Jedi: Joint entity and relation detection using type inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16) - System Demonstrations*. Berlin, Germany, pages 61–66.

Z. Kozareva and E. Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*. Cambridge, Massachusetts, pages 1110–1118.

D. Lin and X. Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP-09)*. Singapore, pages 1030–1038.

Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-12)*. Jeju Island, Korea, pages 523–534.

G. Miller. 1995. WordNet: a lexical database. *Communications of the ACM* 38(11):39–41.

D. Movshovitz-Attias and W. Cohen. 2015. Kb-lda: Jointly learning a knowledge base of hierarchy, relations, and facts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP-15)*. Beijing, China, pages 1449–1459.

P. Nakov and M. Hearst. 2013. Semantic interpretation of noun compounds using verbal and other paraphrases. *ACM Transactions on Speech and Language Processing* 10(3):1–51.

V. Nastase and M. Strube. 2013. Transforming Wikipedia into a large scale multilingual concept network. *Artificial Intelligence* 194:62–85.

P. Nulty and F. Costello. 2013. General and specific paraphrases of semantic relations between nouns. *Natural Language Engineering* 19(03):357–384.

D. Ó Séaghdha and A. Copestake. 2007. Co-occurrence contexts for noun compound interpretation. In *Proceedings of the Workshop on a Broader Perspective on Multiword Expressions*. Prague, Czech Republic, pages 57–64.

M. Paşca. 2014. Acquisition of open-domain classes via intersective semantics. In *Proceedings of the 23rd World Wide Web Conference (WWW-14)*. Seoul, Korea, pages 551–562.

M. Paşca. 2015. Interpreting compound noun phrases using web search queries. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-15)*. Denver, Colorado, pages 335–344.

P. Pantel, E. Crestan, A. Borkovsky, A. Popescu, and V. Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*. Singapore, pages 938–947.

P. Pasupat and P. Liang. 2014. Zero-shot entity extraction from Web pages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*. Baltimore, Maryland, pages 391–401.

S. Ponzetto and R. Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating Wikipedia. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*. Pasadena, California, pages 2083–2088.

S. Ponzetto and M. Strube. 2007. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI-07)*. Vancouver, British Columbia, pages 1440–1447.

S. Riedel, L. Yao, A. McCallum, and B. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Association for Computational Linguistics (NAACL-HLT-13)*. Atlanta, Georgia, pages 74–84.

V. Shwartz, Y. Goldberg, and I. Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*. Berlin, Germany, pages 2389–2398.

R. Snow, D. Jurafsky, and A. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*. Sydney, Australia, pages 801–808.

N. Surtani and S. Paul. 2015. A vsm-based statistical model for the semantic relation interpretation of noun-modifier pairs. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-15)* pages 636–645.

S. Tratz and E. Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*. Uppsala, Sweden, pages 678–687.

T. Van de Cruys, S. Afantenos, and P. Muller. 2013. MELODI: A supervised distributional approach for free paraphrasing of noun compounds. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval-13)*. Atlanta, Georgia, pages 144–147.

P. Verga, A. Neelakantan, and A. McCallum. 2017. Generalizing to unseen entities and entity pairs with row-less universal schema. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL-17)*. Valencia, Spain, pages 613–622.

R. Wang and W. Cohen. 2009. Automatic set instance extraction using the Web. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP-09)*. Singapore, pages 441–449.

D. Weiskopf. 2007. Compound nominals, context, and compositionality. *Synthese* 156(1):161–204.

D. Wijaya and P. Gianfortoni. 2011. Nut case: What does it mean?: Understanding semantic relationship between nouns in noun compounds through paraphrasing and ranking the paraphrases. In *Proceedings of the 1st International Workshop on Search and Mining Entity-Relationship Data (SMER-11)*. Glasgow, United Kingdom, pages 9–14.

C. Xavier and V. Strube de Lima. 2014. Boosting open information extraction with noun-based relations. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-14)*. Reykjavik, Iceland, pages 96–100.

P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics (TACL)* 2:67–78.

Parsing to 1-Endpoint-Crossing, Pagenumber-2 Graphs

Junjie Cao*, **Sheng Huang***, **Weiwei Sun** and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{junjie.cao,huangsheng,ws,wanxiaojun}@pku.edu.cn

Abstract

We study the Maximum Subgraph problem in deep dependency parsing. We consider two restrictions to deep dependency graphs: (a) 1-endpoint-crossing and (b) pagenumber-2. Our main contribution is an exact algorithm that obtains maximum subgraphs satisfying both restrictions simultaneously in time $O(n^5)$. Moreover, ignoring one linguistically-rare structure descreases the complexity to $O(n^4)$. We also extend our quartic-time algorithm into a practical parser with a discriminative disambiguation model and evaluate its performance on four linguistic data sets used in semantic dependency parsing.

1 Introduction

Dependency parsing has long been studied as a central issue in developing syntactic or semantic analysis. Recently, some linguistic projects grounded on deep grammar formalisms, including CCG, LFG, and HPSG, draw attentions to rich syntactic and semantic dependency annotations that are not limited to trees (Hockenmaier and Steedman, 2007; Sun et al., 2014; Ivanova et al., 2012). Parsing for these *deep* dependency representations can be viewed as the search for Maximum Subgraphs (Kuhlmann and Jonsson, 2015). This is a natural extension of the Maximum Spanning Tree (MST) perspective (McDonald et al., 2005) for dependency tree parisng.

One main challenge of the Maximum Subgraph perspective is to design tracTable algorithms for certain graph classes that have good empirical coverage for linguistic annotations. Unfortunately, no previously defined class simultaneously has high coverage and low-degree polynomial parsing algorithms. For example, noncrossing dependency graphs can be found in time $O(n^3)$, but cover only 48.23% of sentences in CCGBank (Kuhlmann and Jonsson, 2015).

We study two well-motivated restrictions to deep dependency graphs: (a) 1-endpoint-crossing (1EC hereafter; Pitler et al., 2013) and (b) pagenumber is less than or equal to 2 (P2 hereafter; Kuhlmann and Jonsson, 2015). We will show that if the output dependency graphs are restricted to satisfy both restrictions, the Maximum Subgraph problem can be solved using dynamic programming in time $O(n^5)$. Moreover, if we ignore one linguistically-rare sub-problem, we can reduce the time complexity to $O(n^4)$. Though this new algorithm is a degenerated one, it has the same empirical coverage for various deep dependency annotations. We evaluate the coverage of our algorithms on four linguistic data sets: CCGBank, DeepBank, Enju HPSGBank and Prague Dependency TreeBank. They cover 95.68%, 97.67%, 97.28% and 97.53% of dependency graphs in the four corpora. The relatively satisfactory coverage makes it possible to parse with high accuracy.

Based on the quartic-time algorithm, we implement a parser with a discriminative disambiguation model. Our new parser can be taken as a graph-based parser which is complementary to transition-based (Henderson et al., 2013; Zhang et al., 2016) and factorization-based (Martins and Almeida, 2014; Du et al., 2015a) systems. We evaluate our parser on four data sets: those used in SemEval 2014 Task 8 (Oepen et al., 2014), and the dependency graphs extracted from CCGbank (Hockenmaier and Steedman, 2007). Evaluations indicate that our parser produces very accurate deep dependency analysis. It reaches state-of-the-art results on average produced by a transition-based system of Zhang et al.

The first two authors contribute equally.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2110–2120
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1193

(2016) and factorization-based systems (Martins and Almeida, 2014; Du et al., 2015a).

The implementation of our parser is available at http://www.icst.pku.edu.cn/lcwm/grass.

2 Background

Dependency parsing is the task of mapping a natural language sentence into a dependency graph. Previous work on dependency parsing mainly focused on tree-shaped representations. Recently, it is shown that data-driven parsing techniques are also applicable to generate more flexible deep dependency graphs (Du et al., 2014; Martins and Almeida, 2014; Du et al., 2015b,a; Zhang et al., 2016; Sun et al., 2017). Parsing for deep dependency representations can be viewed as the search for Maximum Subgraphs for a certain graph class $\mathcal{G}$ (Kuhlmann and Jonsson, 2015), a generalization of the MST perspective for tree parsing. In particular, we have the following optimization problem:

Given an arc-weighted graph $G = (V, A)$, find a subgraph $G' = (V, A' \subseteq A)$ with maximum total weight such that G' belongs to $\mathcal{G}$.

The choice of $\mathcal{G}$ determines the computational complexity of dependency parsing. For example, if $\mathcal{G}$ is the set of projective trees, the problem can be solved in time $O(|V|^3)$, and if $\mathcal{G}$ is the set of noncrossing dependency graphs, the complexity is $O(|V|^3)$. Unfortunately, no previously defined class simultaneously has high coverage on deep dependency annotations and low-degree polynomial decoding algorithms for practical parsing. In this paper, we study well-motivated restrictions: 1EC (Pitler et al., 2013) and P2 (Kuhlmann and Jonsson, 2015). We will show that relatively satisfactory coverage and parsing complexity can be obtained for graphs that satisfy both restrictions.

3 The 1EC, P2 Graphs

3.1 The 1EC Restriction

Pitler et al. (2013) introduced a very nice property for modelling non-projective dependency trees, i.e. 1EC. This property not only covers a large amount of tree annotations in natural language treebanks, but also allows the corresponding MST problem to bo solved in time of $O(n^4)$. The formal description of the 1EC property is adopted from (Pitler et al., 2013).

Definition 1. *Edges e_1 and e_2 cross if e_1 and e_2 have distinct endpoints and exactly one of the endpoints of e_1 lies between the endpoints of e_2.*

Definition 2. *A dependency graph is 1-Endpoint-Crossing if for any edge e, all edges that cross e share an endpoint p.*

Given a sentence $s = w_0 w_1 \cdots w_{n-1}$ of length n, the vertices, i.e. words, are indexed with integers, an arc from w_i to w_j as $a_{(i,j)}$, and the common endpoint, namely pencil point, of all edges crossed with $a_{(i,j)}$ or $a_{(j,i)}$ as $pt(i, j)$. We denote an edge as $e_{(i,j)}$, if we do not consider its direction.

3.2 The P2 Restriction

The term *pagenumber* is referred to as *planar* by some other authors, e.g. (Titov et al., 2009; Gómez-Rodríguez and Nivre, 2010; Pitler et al., 2013). We give the definition of related concepts as follows.

Definition 3. *A book is a particular kind of topological space that consists of a single line called the spine, together with a collection of one or more half-planes, called the pages, each having the spine as its boundary.*

Definition 4. *A book embedding of a finite graph G onto a book B satisfies three conditions: (1) every vertex of G is drawn as a point on the spine of B; (2) every edge of G is drawn as a curve that lies within a single page of B; (3) every page of B does not have any edge crossings.*

Empirically, a deep dependency graph is not very dense and can typically be embedded onto a very *thin* book. To measure the thickness of a graph, we can use its *pagenumber*.

Definition 5. *The book pagenumber of G is the minimum number of pages required for a book embedding of G.*

For sake of concision, we say a graph is "pagenumber-k", meaning that the pagenumber is at most k.

Theorem 1. *The pagenumber of* 1EC *graph may be greater than 2.*

Proof. The graph in Figure 1 gives an instance which is 1EC but the pagenumber of which is 3. There is a cycle, namely $a \to c \to e \to b \to d \to a$, consisting of odd number of edges. □

Pitler et al. (2013) proved that 1EC trees are a subclass of graphs whose pagenumber is at most 2. This property provides the foundation to the

PN≤ 2	1EC	EnjuBank	DeepBank	PCEDT	CCGBank
Yes	Both	32236 (99.53%)	32287 (99.69%)	31866 (98.39%)	38848 (98.09%)
Both	Yes	31507 (97.28%)	31634 (97.67%)	31589 (97.53%)	37913 (95.73%)
Yes	Yes	31507 (97.28%)	31634 (97.67%)	31589 (97.53%)	37894 (95.68%)
No	Yes	0 (0.0%)	0 (0.0%)	0 (0.0%)	19 (0.05%)
Yes	No	729 (2.25%)	653 (2.02%)	277 (0.86%)	954 (2.41%)
Sentences		32389	32389	32389	39604

Table 1: Coverage in terms of complete graphs under various structural restrictions. Column "PN≤ 2" indicates whether the restriction "P2" is satisfied; Column "1EC" indicates whether the restriction "1EC" is satisfied.

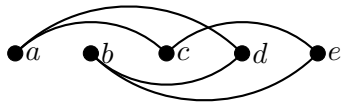


Figure 1: A 1EC graph whose pagenumber is 3.

success in designing dynamic programming algorithms for trees. Theorem 1 indicates that when we consider more general graph, the case is more complicated. In this paper, we study graphs that are constrained to be both 1EC and P2. We call them 1EC/P2 graphs.

3.3 Coverage on Linguistic Data

To show that the two restrictions above are well-motivated for describing linguistic data, we evaluate their empirical coverage on four deep dependency corpora (as defined in Section 5.2). These corpora are also used for training and evaluating our data-driven parsers. The coverage is evaluated using sentences in the training sets.

Table 1 shows the results. We can see that 1EC is also an empirical well-motivated restriction when it comes to deep dependency structures. The P2 property has an even better coverage. Unfortunately, it is a NP-hard problem to find optimal P2 graphs (Kuhlmann and Jonsson, 2015). Though theoretically a 1EC graph is not necessarily P2, the empirical evaluation demonstrates the high overlap of them on linguistic annotations. In particular, almost all 1EC deep dependency graphs are P2. The percentages of graphs satisfying both restrictions vary between 95.68% for CCGBank and 97.67% for DeepBank. The relatively satisfactory coverage enables accurate practical parsing.

4 The Algorithm

This section contains the main contribution of this paper: a polynomial time exact algorithm for solving the Maximum Subgraph problem for the class of 1EC/P2 graphs.

Theorem 2. *Take* 1EC/P2 *graphs as target subgraphs, the maximum subgraph problem can be solved in time* $O(|V|^5)$.

For sake of formal concision, we introduce the algorithm of which the goal is to calculate the maximum score of a subgraph. Extracting corresponding optimal graphs can be done in a number of ways. For example, we can maintain an auxiliary arc table which is populated parallel to the procedure of obtaining maximum scores.

Our algorithm is highly related to the following property: Every subgraph of a 1EC/P2 graph is also a 1EC/P2 graph. We therefore focus on maximal 1EC/P2 graphs, a particular type of 1EC/P2 graphs defined as follows.

Definition 6. *A maximal* 1EC/P2 *graph is a* 1EC/P2 *graph that cannot be extended by including one more edge.*

Our algorithm is a bottom-up dynamic programming algorithm. It defines different structures corresponding to different sub-problems, and visits all structures from bottom to top, finding the best combination of smaller structures to form a new structure. The key design is to make sure that it can produce all maximal 1EC/P2 graphs. During the search for maximal 1EC/P2 graphs, we can freely delete bad edges whose scores are negative. In particular, we figure out some edges, in each construction step, which can be created without violating either 1EC or P2 restriction. Assume the arc weight associated with $a_{(i,j)}$ is $w[i,j]$. Then we define a function SELECT(i,j) according to the comparison of 0 and $w[i,j]$ as well as $w[j,i]$. If $w[i,j] \geq 0$ (or $w[j,i] \geq 0$), we then *select* $a_{(i,j)}$ (or $a_{(j,i)}$) and add it to currently the best solution of a sub-problem. SELECT(i,j) returns $\max(\max(0, w[i,j]) + \max(0, w[j,i]))$. If we allow at most one arc between two nodes, SELECT(i,j) returns $\max(0, w[i,j], w[j,i])$.

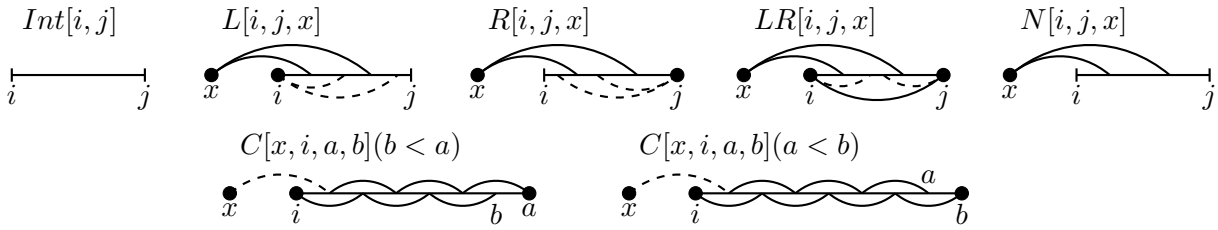


Figure 2: Graphic representations of sub-problems.

The graphical illustration of our algorithm uses undirected graphs[1]. In other words, we use $e_{(i,j)}$ to include the discussion about both $a_{(i,j)}$ and $a_{(j,i)}$.

4.1 Sub-problems

We consider six sub-problems when we construct a maximum dependency graph on a given (closed) interval $[i, k] \subseteq V$ of vertices. When we focus on the nodes strictly inside this interval, and we use an open interval (i, k) to exclude i and j. See Figure 2 for graphical visualization. The first five are adapted in concord with Pitler et al. (2013)'s solution for trees, and we introduce a new sub-problem, namely **C**. Because graphs allow for loops as well as disconnectedness, the sub-problems are simplified to some extent, while a special case of **LR** is now prominent. **C** is thus introduced to represent the special case. The sub-problems are explained as follows.

Int $Int[i, j]$ represents a partial analysis associated with an interval from i to j inclusively. $Int[i, j]$ may or may not contain edge $e_{(i,j)}$. To parse a given sentence is equivalent to solve the problem $Int[0, n-1]$.

L $L[i, j, x]$ represents a partial analysis associated with an interval from i to j inclusively as well as an external vertex x. $\forall p \in (i, j), pt(x, p) = i$. $L[i, j, x]$ can contain $e_{(i,j)}$ but disallows $e_{(x,i)}$ or $e_{(x,j)}$.

R $R[i, j, x]$ represents a partial analysis associated with an interval from i to j inclusively as well as an external vertex x. $\forall p \in (i, j), pt(x, p) = j$. $R[i, j, x]$ can contain $e_{(i,j)}$ but disallows $e_{(x,i)}$ or $e_{(x,j)}$.

LR $LR[i, j, x]$ represents a partial analysis associated with an interval from i to j inclusively as well as an external vertex x. $\forall p \in (i, j), pt(x, p) = i$ or j. $LR[i, j, x]$ must allow $e_{(i,j)}$ but disallows $e_{(x,i)}$ or $e_{(x,j)}$.

N $N[i, j, x]$ represents a partial analysis associated with an interval from i to j inclusively and an external vertex x. $\forall p \in (i, j), pt(x, p) \notin [i, j]$. $N[i, j, x]$ can contain $e_{(i,j)}$ but disallows $e_{(x,i)}$ or $e_{(x,j)}$.

C $C[x, i, a, b](a \neq b, a > i, b > i)$ represents a partial analysis associated with an interval from i to $max\{a, b\}$ inclusively and an external vertex x. Intuitively, **C** depicts a class of graphs constructed by upper- and lower-plane edges arranged in a staggered pattern. a stands for the last endpoint in the upper plane, and b the last endpoint in the lower plane.

We give a definition of **C**. There exists in $C[x, i, a, b]$ a series $\{s_1, \cdots, s_m\}$ that fulfills the following constraints:

1. $s_1 = i < s_2 < ... < s_m = \max\{a, b\}$.
2. $\exists e_{(x,s_2)}$.
3. $\forall k \in [1, m-2], \exists e_{(s_k, s_{k+2})}$.
4. $\forall k \in [1, m-2], \nexists e_{(l,r)} (s_k, s_{k+2}) \subset (l, r) \subset (s_1, s_m)$[2].
5. $\forall k \in [2, m-3]$, $e_{(s_k, s_{k+2})}$ crosses only with $e_{(s_{k-1}, s_{k+1})}$ and $e_{(s_{k+1}, s_{k+3})}$; $e_{(s_1, s_3)}$ crosses only with $e_{(s_2, s_4)}$ and $e_{(x,s_2)}$; $e_{(s_{m-2}, s_m)}$ crosses only with $e_{(s_{m-3}, s_{m-1})}$.
6. $e_{(x,s_{m-1})}, e_{(s_1,s_m)}, e_{(x,s_1)}, e_{(x,s_m)}$ are disallowed.
7. While $a < b$, the series can be written as $\{s_1 = i, \cdots, s_{m-1} = a, s_m = b\} (m \geq 5)$. While $b < a$, the series is $\{s_1, \cdots, s_{m-1} =$

[1] The *single-head* property does not hold. We currently do not consider other constraints of directions. So prediction of the direction of one edge does not affect prediction of other edges as well as their directions. The directions can be assigned *locally*, and our parser builds directed rather than undirected graphs in this way. Undirected graphs are only used to conveniently illustrate our algorithms. All experimental results in Section 5.2 consider directed dependencies in a standard way. We use the official evaluation tool provided by SDP2014 shared task. The numberic results reported in this paper are directly comparable to results in other papers.

[2] By "$(x, y) \subset (z, w)$," we mean $x \geq z, y < w$ or $x > z, y \leq w$.

$b, s_m = a\}(m \geq 4)$. We denote the two cases using the signs **C1** and **C2** respectively.

The distinction between **C1** and **C2** is whether there is one more edge below than above.

4.2 Decomposing an Int Sub-problem

Consider an $Int[i,j]$ sub-problem. Assume that $k(k \in (i,j))$ is the farthest vertex that is linked with i, and $l = pt(i,k)$. When $j - i > 1$, there must be such a k given that we consider maximal 1EC/P2 graphs. There are three cases.

Case 1: $l = j$**.** Vertex k divides the interval $[i,j]$ into two parts: $[i,k]$ and $[k,j]$. First notice that the edges linking (i,k) and j can only cross with $e_{(i,k)}$. Thus i or k can be the pencil points of those edges, which entails that interval $[i,k]$ is an **LR** in respect to external vertex j. Because there exist no edge from i to any node in (k,j), interval $[k,j]$ is an **Int**. The problem is eventually decomposed to: $LR[i,k,j] + Int[k,j] + \text{SELECT}[i,j]$.

Case 2: $l \in (k,j)$**.** In this case, we can freely add $e_{(i,l)}$ without violating either 1EC or P2 conditions. Therefore Case 2 does not lead to any maximal 1EC/P2 graph. Our algorithm does not need to explicitly handle this case, given that they can be derived from solutions to other cases.

Case 3: $l \in (i,k)$**.** Now assume that there is an edge from i to a vertex in (l,k). Consider the farthest vertex that is linked with l, say $p(p \in (k,j)$. We can freely add $e_{(i,p)}$ without violating the 1EC and P2 restrictions. Similar to Case 2, we do not explicitly deal with this case.

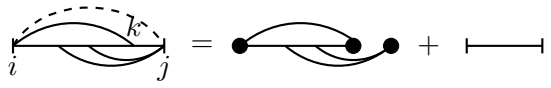
If there is no edge from i to any vertex in (l,k), then $[i,l]$, $[l,k]$, $[k,j]$ are **R**, **Int**, **L** respectively. Three external edges are $e_{(i,k)}$, $e_{(l,j)}$, and $e_{(i,j)}$. The decomposition is: $R[i,l,k]+Int[l,k]+L[k,j,l] + \text{SELECT}[l,j] + \text{SELECT}[i,j]$.

4.3 Decomposing an L Sub-problem

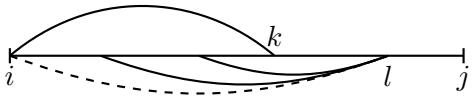
If there is no edge from x to any node in (i,j), the graph is reduced to $Int[i,j]$. If there is one, let k be the vertex farthest from i and adjacent to x. There are two different cases, as shown in Figure 4.

1. If there exists an edge from x to some node in (i,k), intervals $[i,k]$, $[k,j]$ are classified as **L**, **N** respectively. Two edges external to the interval: $e_{(x,k)}$, $e_{(i,j)}$. The decomposition is $L[i,k,x]+N[k,j,i]+\text{SELECT}[x,k]+\text{SELECT}[i,j]$.

Case 1: $l = j$



Case 2: $l \in (k,j)$



Case 3: $l \in (i,k)$

Does such a dashed edge exist?

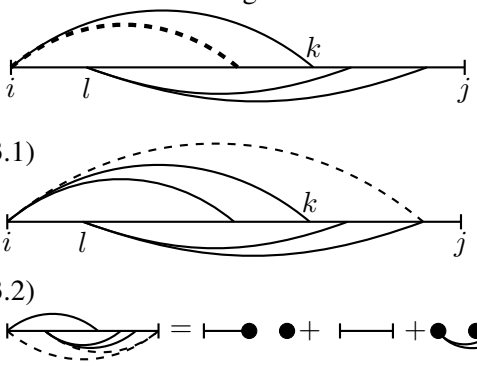


Figure 3: Decomposition for $Int[i,j]$, with $pt(i,k) = l$.

Does such a dashed edge exist?

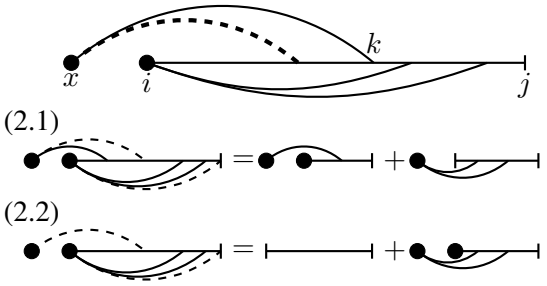


Figure 4: Decomposition for $L[i,j,x]$.

2. Otherwise, Intervals $[i,k]$, $[k,j]$ are classified as **Int**, **L** respectively. Two edges external to the interval: $e_{(x,k)}$, $e_{(i,j)}$. The decomposition is $Int[i,k] + L[k,j,i] + \text{SELECT}[x,k] + \text{SELECT}[i,j]$.

4.4 Decomposing an R Sub-problem

If there is no edge from x to (i,j), then the graph is reduced to $Int[i,j]$. If there is one, let k be the farthest vertex from j and adjacent to x. There are two different cases:

1. If there exist an edge from x to (k,j), Intervals $[i,k]$, $[k,j]$ are classified as **N**, **R** respectively. Two edges external to the interval: $e_{(x,k)}$, $e_{(i,j)}$. The decomposition

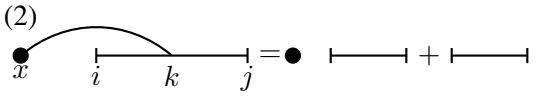


Figure 5: Decomposition for $N[i, j, x]$.

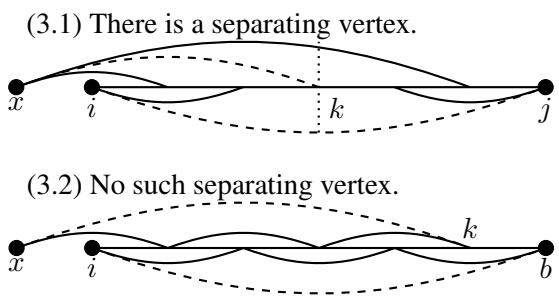


Figure 6: Decomposition for $LR[i, j, x]$.

is $N[i, k, j] + R[k, j, x] + \text{SELECT}[x, k] + \text{SELECT}[i, j]$.

2. Otherwise, Intervals $[i, k]$, $[k, j]$ are classified as **R**, **Int** respectively. Two edges external to the interval are $e_{(x,k)}$, $e_{(i,j)}$. The decomposition is $R[i, k, j] + Int[k, j] + \text{SELECT}[x, k] + \text{SELECT}[i, j]$.

The decomposition is similar to **L**, we thus do not give a graphical representation to save space.

4.5 Decomposing an N Sub-problem

If there is no edge from x to (i, j), then the graph is reduced to $Int[i, j]$. If there is one, let k be the farthest vertex from i and adjacent to x. By definition, $N[i, j, x]$ does not allow for $e_{(x,i)}$ or $e_{(x,j)}$. Thus $k \neq i$ or j. Intervals $[i, k]$, $[k, j]$ are classified as **N**, **Int** respectively. Two edges external to the interval are $e_{(x,k)}$, $e_{(i,j)}$. The decomposition is $N[i, k, x] + Int[k, j] + \text{SELECT}[x, k] + \text{SELECT}[i, j]$.

4.6 Decomposing an LR Sub-problem

If the pencil point of all edges from x to (i, j) is i, then the model is the same as $L[i, j, x]$. Similary, if the pencil point is j, then the model is the same as $R[i, j, x]$.

If some of the edges from x to (i, j) share a pencil point i, and the others share j, there are two different cases.

1. If there is a k which satisfies that within $[i, j]$, only $e_{(i,j)}$ crosses over k (i.e., $[i, j]$ can be divided along dashed line k into two), then, k divides $[i, j]$ into $[i, k]$ and $[k, j]$. Because k is not allowed to be pencil point, the two subintervals must be an **L** and an **R** in terms of external x, respectively. In addition, there are two edges, namely $e_{(x,k)}$ and $e_{(i,j)}$ not included by the subintervals. The problem is thus decomposed as $L[i, k, x] + R[k, j, x] + \text{SELECT}[x, k] + \text{SELECT}[i, j]$.

2. If there is no such k in concord with the condition in (1), it comes a much more difficult case for which we introduce sub-problem **C**. Here we put forward the conclusion:

Lemma 1. *Assume that $k(k \in (i, j))$ is the vertex that is adjacent to x and farthest from i. The decomposition for the second case is $C[x, i, k, j] + \text{SELECT}[x, k] + \text{SELECT}[i, j]$.*

Proof. The distinction between Case 1 and 2 implies the following property, which is essential, $\forall t \in (i, j), \exists e_{(pl,pr)}$ such that $t \in (pl, pr) \subset [i, j]$.

We can recursively generate a series of length n—$\{e_{(sl_k,sr_k)}\}$—in $LR[i, j, x]$ as follows.

$k = 1$ Let $sl_k = i, sr_k = \max\{p | p \in (i + 1, j) \text{ and } \exists e_{(i,p)}\}$;

$k > 1$ For sr_{k-1}, we denote all edges that cover it as $e_{(pl_1,pr_1)}, \cdots, e_{(pl_s,pr_s)}$. Note that there is at least one such edge. For any two edges in them, viz $e_{(pl_u,pr_u)}$ and $e_{(pl_v,pr_v)}$, $(pl_u, pr_u) \subset (pl_v, pr_v)$ or $(pl_v, pr_v) \subset (pl_u, pr_u)$. Otherwise, the P2 property no longer holds due to the interaction among $e_{(sl_{k-1},sr_{k-1})}$, $e_{(pl_u,pr_u)}$ and $e_{(pl_v,pr_v)}$. Assume (pl_w, pr_w) is the largest one, then we let $sl_k = pl_w, sr_k = pr_w$. When $sr_k = j$, recursion ends.

We are going to prove that if we delete two edges $e_{(x,sr_{n-1})}$ and $e_{(i,j)}$ from $LR[i, j, x]$, the series $\{sl_1, sl_2, sl_3, ..., sl_{n-2}, sl_{n-1}, sl_n, sr_{n-1}, sr_n\}$ satisfies each and all the conditions of **C1**.

Condition 1. Because $e_{(sl_n,sr_n)}$ covers sr_{n-1}, Condition 1 holds for $k = m-3, m-2$. Consider $k \leq m-4 = n-2$. Assume that $s_{k+1} < s_k$, then we have $e_{(s_{k+1},sr_{k+1})}$ is larger than $e_{(s_k,sr_{k+1})}$. This is impossible because we select the largest edge in every step.

Condition 2. The **LR** sub-problem we discussed now cannot be reduced to **L** nor **R**, so there must be two edges from x that respectively cross edges linked to i and j. We are going to prove that

the two edges must be $e_{(x,s_2)}$ and $e_{(x,sr_{n-1})}$. Assume that there is $e_{(x,p)}$, where $p \in (i,j)$, $p \neq s_2$ and $p \neq sr_{n-1}$. If $p \in (i,s_2)$, then $e_{(s_1,s_3)}$ crosses with $e_{(x,p)}$ and $e_{(s_2,s_4)}$ simultaneously. 1EC is violated. If $p \in (s_2, sr_{n-1})$, $e_{(x,p)}$ necessarily crosses with some edge $e_{(s_k,s_{k+2})}$. Furthermore, $i < s_k < s_{k+2} < j$. Thus 1EC is violated. If $p \in (sr_{n-1}, j)$, the situation is similar to $p \in (i, s_2)$.

Condition 3. $\forall k \in [1, n-2]$, $e_{(sl_k,sr_k)}$ and $e_{(sl_{k+1},sr_{k+1})}$ cross, $e_{(sl_{k+1},sr_{k+1})}$ and $e_{(sl_{k+2},sr_{k+2})}$ cross, so $sr_k \leq sl_{k+2}$. Otherwise the interaction of the three edges results in the violation of P2. If $sr_k < sl_{k+2}$, $e_{(sl_k,sr_k)}$ and $e_{(sl_{k+2},sr_{k+2})}$ share no common endpoint, violating 1EC. Therefore, $sr_k = sl_{k+2} = s_{k+2}$, and Condition 3 is satisfied.

We also reach proposition that $pt(s_k, s_{k+2}) = s_{k+1}$.

Condition 4. This condition is easy to verify because (s_k, s_{k+2}) is the largest with respective to sr_k.

Condition 5. Assume, that there is $e_{(pl,pr)}$ which intersects with $e_{(s_k,s_{k+2})}$, and at the same time satisfy the conditions: $e_{(pl,pr)} \notin \{e_{(s_t,s_{t+2})} | t \in [1, m-2]\} \cup \{e_{(x,s_2)}, e_{(x,sr_{n-1})}\}$. Since $pt(s_k, s_{k+2}) = s_{k+1}$, $pl = s_{k+1}$ or $pr = s_{k+1}$.

If $pl = s_{k+1}$, then $pl < sl_{k+2} < pr$, and in turn $k < m-2$. In addition, according to Condition 4, $(pl, pr) \subset (s_{k+1}, s_{k+3})$. So $pr < s_{k+3}$. If $k = m-3$ then $e_{(x,s_{n-1})}$ crosses with $e_{(pl,pr)}$ and $e_{(i,j)}$ simultaneously. 1EC is violated. If $k < m-3$ then $e_{(s_{k+2},s_{k+4})}$ cross with $e_{(pl,pr)}$, and $pr < s_{k+3} = pt(e_{(s_{k+2},s_{k+4})})$. Again 1EC is violated. If $pr = s_{k+1}$ The symmetry of our proof entails the violation of 1ec.

All in all, the assumption does not hold and thus satisfies Condition 5.

Condition 6. $e_{(x,s_1)}, e_{(x,s_m)}$ are disallowed due to definition of an **LR** problem. $e_{(x,s_{m-1})}, e_{(s_1,s_m)}$ are disallowed due to the decomposition.

Condition 7. Due to the existence of $e_{(x,s_2)}$ and $e_{(x,sr_{n-1})}$, there must be two edges: $e_{(x,p_1)}$ and $e_{(x,p_2)}$ that cross $e_{(i,s_2)}$ and $e_{(sr_{n-1},j)}$ respectively. There must be an odd number of edges in the series $\{e_{(sl_k,sr_k)}\}$, otherwise P2 is violated as the case shown in Figure 1. In summary, the last condition is satisfied and we have a **C1** structure in this **LR** sub-problem.

□

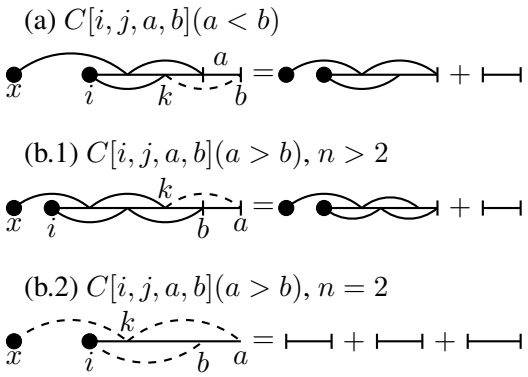


Figure 7: Decomposition for $C[x,i,a,b]$.

4.7 Decomposing a C Sub-problem

We illustrate the decomposition using the graphical representations shown in Figure 7. When $a < b$, since a is the *upper-plane* endpoint farthest to the right, and b is the *lower-plane* counterpart, in this case a precedes b (i.e., a is to the left of b).

Let $C[x,i,a,k]$ be a **C** in which the lower-plane endpoint k precedes a. Add $e_{(k,b)}$ gives a new **C** sub-problem with lower-plane endpoint preceded by the upper-plane one. The decomposition is then $C[x,i,a,k] + Int[a,b] + \text{SELECT}[k,b]$.

When $a > b$ and $n > 2$, the lower-plane endpoint b precedes a. In analogy, the case can be obtained by adding $e_{(k,a)}$ to $C[x,i,k,b]$. The decomposition: $C[x,i,k,b] + Int[b,a] + \text{SELECT}[k,a]$.

When $n = 2$, we reach the most fundamental case. Only 4 vertices are in the series, namely i,k,b,a. Moreover, there are three edges: $e_{(x,k)}$, $e_{(i,b)}$, $e_{(k,a)}$, and the interval [i,a] is divided by k,b into three parts. The decomposition is $Int[i,k] + Int[k,b] + Int[b,a] + \text{SELECT}[x,k] + \text{SELECT}[i,b] + \text{SELECT}[k,a]$.

4.8 Discussion

4.8.1 Soundness and Completeness

The algorithm is sound and complete with respective to 1EC/P2 graphs. We present our algorithms by detailing the decomposition rules. The completeness is obvious because we can decompose any 1EC/P2 graph from an **Int**, use our rules to reduce it into smaller sub-problems, and repeat this procedure. The decomposition rules are also construction rules. During constructing graphs by applying these rules, we never violate 1EC nor P2

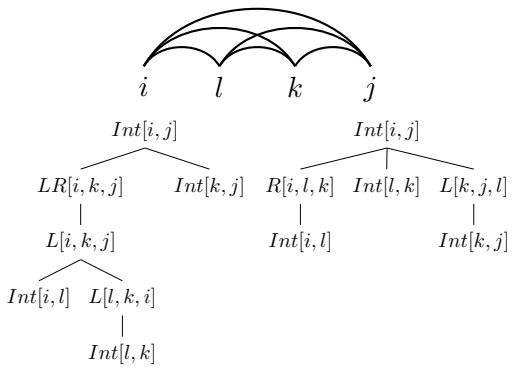


Figure 8: A maximal 1EC/P2 graph and its two derivations. For brevity, we elide the edges created in each derivation step.

restrictions. So our algorithm is sound.

4.8.2 Greedy Search during Construction

There is an important difference between our algorithm and Eisner-style MST algorithms (Eisner, 1996b; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010) for trees as well as Kuhlmann and Jonsson's Maximum Subgraph algorithm for noncrossing graphs. In each construction step, our algorithm allows multiple arcs to be constructed, but whether or not such arcs are added to the target graph depends on their arc-weights. In each step, we do greedy search and decide if adding an related arc according to local scores. If all arcs are assigned scores that are greater than 0, the output of our algorithm includes the most complicated 1EC/P2 graphs. That means adding one more arc voilates the 1EC or P2 restrictions. For all other aforementioned algorithms, in a single construction step, it is clear whether to add a new arc, and which one. There is no local search.

4.8.3 Spurious Ambiguity

To generate the same graph, even a maximal 1EC/P2 graph, we may have different derivations. Figure 8 is an example. This is similar to syntactic analysis licensed by Combinatory Categorial Grammar (CCG; Steedman, 1996, 2000). To derive one surface string, there usually exists multiple CCG derivations. A practice of CCG parsing is defining one particular derivation as the *standard* one, namely normal form (Eisner, 1996a). The spurious ambiguity in our algorithm does not affect the correctness of first-order parsing, because scores are assigned to individual dependencies, rather than derivation processes. There is no need to distinguish one special derivation here.

4.8.4 Complexity

The sub-problem **Int** is of size $O(n^2)$, each graph of which takes a calculating time of order $O(n^2)$. For sub-problems **L**, **R**, **LR**, and **N**, each has $O(n^3)$ elements, with a unit calculating time $O(n)$. **C** has $O(n^4)$ elements, with a unit calculating time $O(n)$. Therefore the full version algorithm runs in time of $O(n^5)$ with a space requirement of $O(n^4)$.

4.9 A Degenerated Version

We find that graphical structures involved in the **C** sub-problem, namely coupled staggered pattern, is extremely rare in linguistic analysis. If we ignore this special case, we get a degenerated version of dynamic programming algorithm. This algorithm can find a strict subset of 1EC/P2 graphs. We can improve efficiency without sacrificing *expressiveness* in terms of linguistic data. This degenerated version algorithm requires $O(n^4)$ time and $O(n^3)$ space.

5 Practical Parsing

5.1 Disambiguation

We extend our quartic-time parsing algorithm into a practical parser. In the context of data-driven parsing, this requires an extra disambiguation model. As with many other parsers, we employ a global linear model. Following Zhang et al. (2016)'s experience, we define rich features extracted from word, POS-tags and pseudo trees. For details we refer to the source code. To estimate parameters, we utilize the averaged perceptron algorithm (Collins, 2002).

5.2 Data

We conduct experiments on unlabeled parsing using four corpora: CCGBank (Hockenmaier and Steedman, 2007), DeepBank (Flickinger et al., 2012), Enju HPSGBank (EnjuBank; Miyao et al., 2004) and Prague Dependency TreeBank (PCEDT; Hajic et al., 2012), We use "standard" training, validation, and test splits to facilitate comparisons. Following previous experimental setup for CCG parsing, we use section 02-21 as training data, section 00 as the development data, and section 23 for testing. The other three data sets are from SemEval 2014 Task 8 (Oepen et al.,

	DeepBank			EnjuBank			CCGBank			PCEDT		
	UP	UR	UF	UP	UR	UF	UP	UR	UF	UP	UR	UF
P1	90.75	86.13	88.38	93.38	90.20	91.76	94.21	88.55	91.29	90.61	85.69	88.08
1ECP2^d	91.05	87.22	89.09	93.41	91.83	92.61	94.41	91.41	92.89	90.76	86.31	88.48

Table 2: Parsing accuracy evaluated on the development sets.

	DeepBank			EnjuBank			CCGBank			PCEDT		
	UP	UR	UF	UP	UR	UF	UP	UR	UF	UP	UR	UF
Ours	90.91	86.98	88.90	93.83	91.49	92.64	94.23	91.13	92.66	90.09	85.90	87.95
ZDSW	89.04	88.85	88.95	92.92	92.83	92.87	92.49	92.30	92.40	- -	- -	- -
MA	90.14	88.65	89.39	93.18	91.12	92.14	- -	- -	- -	90.21	85.51	87.80
DSW	- -	- -	- -	- -	- -	- -	93.03	92.03	92.53	- -	- -	- -

Table 3: Parsing accuracy evaluated on the test sets.

2014), and the data splitting policy follows the shared task. All the four data sets are publicly available from LDC (Oepen et al., 2016).

Experiments for CCG-grounded analysis were performed using automatically assigned POS-tags that are generated by a symbol-refined HMM tagger (Huang et al., 2010). Experiments for the other three data sets used POS-tags provided by the shared task. We also use features extracted from pseudo trees. We utilize the Mate parser (Bohnet, 2010) to generate pseudo trees. The pre-processing for CCGBank, DeepBank and EnjuBank are exactly the same as in experiments reported in (Zhang et al., 2016).

5.3 Accuracy

We evaluate two parsing algorithms, the algorithm for noncrossing dependency graphs (Kuhlmann and Jonsson, 2015), i.e. pagenumber-1 (denoted as *P1*) graphs, and our quartic-time algorithm (denoted as *1ECP2*^{d}). Table 2 summerizes the accuracy obtained our parser. Same feature templates are applied for disambiguation. We can see that our new algorithm yields significant improvements on all data sets, as expected. Especially, due to the improved coverage, the recall is improved more.

5.4 Comparison with Other Parsers

Our new parser can be taken as a graph-based parser which employ a different architecture from transition-based and factorization-based (Martins and Almeida, 2014; Du et al., 2015a) systems. We compare our parser with the best reported systems in the other two architectures. *ZDSW* (Zhang et al., 2016) is transition-based parser while *MA* (Martins and Almeida, 2014) and DSW (Du et al., 2015a) are two factorization-based systems. All of them achieves state-of-the-art performance. All results on the test set is shown in Table 3. We can see that our parser, as a graph-based parser, is comparable to state-of-the-art transition-based and factorization-based parsers.

6 Conclusion and Future Work

In this paper, we explore the strength of the graph-based approach. In particular, we enhance the Maximum Subgraph model with new parsing algorithms for 1EC/P2 graphs. Our work indicates the importance of finding appropriate graph classes that on the one hand are linguistically expressive and on the other hand allow efficient search. Within tree-structured dependency parsing, higher-order factorization that conditions on wider syntactic contexts than arc-factored relationships have been proved very useful. The arc-factored model proposed in this paper may be enhanced with higher-order features too. We leave this for future investigation.

Acknowledgments

This work was supported by 863 Program of China (2015AA015403), NSFC (61331011), and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

We thank the first anonymous reviewer whose valuable comments led to significant revisions. We thank Xingfeng Shi for his help in explicating the idea.

Weiwei Sun is the corresponding author.

References

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, Beijing, China, pages 89–97. http://www.aclweb.org/anthology/C10-1011.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *In Proc. EMNLP-CoNLL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1–8. https://doi.org/10.3115/1118693.1118694.

Yantao Du, Weiwei Sun, and Xiaojun Wan. 2015a. A data-driven, factorization parser for CCG dependency structures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1545–1555. http://www.aclweb.org/anthology/P15-1149.

Yantao Du, Fan Zhang, Weiwei Sun, and Xiaojun Wan. 2014. Peking: Profiling syntactic tree parsing techniques for semantic graph parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 459–464. http://www.aclweb.org/anthology/S14-2080.

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015b. Peking: Building semantic dependency graphs with a hybrid parser. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 927–931. http://www.aclweb.org/anthology/S15-2154.

Jason Eisner. 1996a. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*. Santa Cruz, pages 79–86.

Jason M. Eisner. 1996b. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 340–345.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*. pages 85–96.

Carlos Gómez-Rodríguez and Joakim Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1492–1501. http://www.aclweb.org/anthology/P10-1151.

Jan Hajic, Eva Hajicová, Jarmila Panevová, Petr Sgall, Ondej Bojar, Silvie Cinková, Eva Fucíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jirí Semecký, Jana Sindlerová, Jan Stepánek, Josef Toman, Zdenka Uresová, and Zdenek Zabokrtský. 2012. Announcing prague czech-english dependency treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*. Istanbul, Turkey.

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

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics* 33(3):355–396.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 12–22. http://www.aclweb.org/anthology/D10-1002.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*. Jeju, Republic of Korea, pages 2–11.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1–11. http://www.aclweb.org/anthology/P10-1001.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics* 3:559–570.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 471–476. http://www.aclweb.org/anthology/S14-2082.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*. volume 6, pages 81–88.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Vancouver, British Columbia, Canada, pages 523–530.

Yusuke Miyao, Takashi Ninomiya, and Jun ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. In *IJCNLP*. pages 684–693.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Zdeňka Urešová. 2016. Semantic Dependency Parsing (SDP) graph banks release 1.0 LDC2016T10. Web Download.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 63–72. http://www.aclweb.org/anthology/S14-2008.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2013. Finding optimal 1-endpoint-crossing trees. *TACL* 1:13–24. http://www.transacl.org/wp-content/uploads/2013/03/paper13.pdf.

M. Steedman. 1996. *Surface Structure and Interpretation*. Linguistic Inquiry Monographs. Mit Press. http://books.google.ca/books?id=Mh1vQgAACAAJ.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

Weiwei Sun, Junjie Cao, and Xiaojun Wan. 2017. Semantic dependency parsing via book embedding. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Weiwei Sun, Yantao Du, Xin Kou, Shuoyang Ding, and Xiaojun Wan. 2014. Grammatical relations in Chinese: GB-ground extraction and data-driven parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 446–456. http://www.aclweb.org/anthology/P14-1042.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online graph planarisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of the 21st international jont conference on Artifical intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pages 1562–1567. http://dl.acm.org/citation.cfm?id=1661445.1661696.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389. http://aclweb.org/anthology/J16-3001.

Semi-supervised Multitask Learning for Sequence Labeling

Marek Rei
The ALTA Institute
Computer Laboratory
University of Cambridge
United Kingdom
marek.rei@cl.cam.ac.uk

Abstract

We propose a sequence labeling framework with a secondary training objective, learning to predict surrounding words for every word in the dataset. This language modeling objective incentivises the system to learn general-purpose patterns of semantic and syntactic composition, which are also useful for improving accuracy on different sequence labeling tasks. The architecture was evaluated on a range of datasets, covering the tasks of error detection in learner texts, named entity recognition, chunking and POS-tagging. The novel language modeling objective provided consistent performance improvements on every benchmark, without requiring any additional annotated or unannotated data.

1 Introduction

Accurate and efficient sequence labeling models have a wide range of applications, including named entity recognition (NER), part-of-speech (POS) tagging, error detection and shallow parsing. Specialised approaches to sequence labeling often include extensive feature engineering, such as integrated gazetteers, capitalisation features, morphological information and POS tags. However, recent work has shown that neural network architectures are able to achieve comparable or improved performance, while automatically discovering useful features for a specific task and only requiring a sequence of tokens as input (Collobert et al., 2011; Irsoy and Cardie, 2014; Lample et al., 2016).

This feature discovery is usually driven by an objective function based on predicting the annotated labels for each word, without much incentive to learn more general language features from the available text. In many sequence labeling tasks, the relevant labels in the dataset are very sparse and most of the words contribute very little to the training process. For example, in the CoNLL 2003 NER dataset (Tjong Kim Sang and De Meulder, 2003) only 17% of the tokens represent an entity. This ratio is even lower for error detection, with only 14% of all tokens being annotated as an error in the FCE dataset (Yannakoudakis et al., 2011). The sequence labeling models are able to learn this bias in the label distribution without obtaining much additional information from words that have the majority label (O for outside of an entity; C for correct word). Therefore, we propose an additional training objective which allows the models to make more extensive use of the available data.

The task of language modeling offers an easily accessible objective – learning to predict the next word in the sequence requires only plain text as input, without relying on any particular annotation. Neural language modeling architectures also have many similarities to common sequence labeling frameworks: words are first mapped to distributed embeddings, followed by a recurrent neural network (RNN) module for composing word sequences into an informative context representation (Mikolov et al., 2010; Graves et al., 2013; Chelba et al., 2013). Compared to any sequence labeling dataset, the task of language modeling has a considerably larger and more varied set of possible options to predict, making better use of each available word and encouraging the model to learn more general language features for accurate composition.

In this paper, we propose a neural sequence labeling architecture that is also optimised as a language model, predicting surrounding words in the dataset in addition to assigning labels to each token. Specific sections of the network are op-

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2121–2130
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1194

timised as a forward- or backward-moving language model, while the label predictions are performed using context from both directions. This secondary unsupervised objective encourages the framework to learn richer features for semantic composition without requiring additional training data. We evaluate the sequence labeling model on 10 datasets from the fields of NER, POS-tagging, chunking and error detection in learner texts. Our experiments show that by including the unsupervised objective into the training process, the sequence labeling model achieves consistent performance improvements on all the benchmarks. This multitask training framework gives the largest improvements on error detection datasets, outperforming the previous state-of-the-art architecture.

2 Neural Sequence Labeling

We use the neural network model of Rei et al. (2016) as the baseline architecture for our sequence labeling experiments. The model takes as input one sentence, separated into tokens, and assigns a label to every token using a bidirectional LSTM.

The input tokens are first mapped to a sequence of distributed word embeddings $[x_1, x_2, x_3, ..., x_T]$. Two LSTM (Hochreiter and Schmidhuber, 1997) components, moving in opposite directions through the sentence, are then used for constructing context-dependent representations for every word. Each LSTM takes as input the hidden state from the previous time step, along with the word embedding from the current step, and outputs a new hidden state. The hidden representations from both directions are concatenated, in order to obtain a context-specific representation for each word that is conditioned on the whole sentence in both directions:

$$\overrightarrow{h_t} = LSTM(x_t, \overrightarrow{h_{t-1}}) \quad (1)$$

$$\overleftarrow{h_t} = LSTM(x_t, \overleftarrow{h_{t+1}}) \quad (2)$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \quad (3)$$

Next, the concatenated representation is passed through a feedforward layer, mapping the components into a joint space and allowing the model to learn features based on both context directions:

$$d_t = tanh(W_d h_t) \quad (4)$$

where W_d is a weight matrix and $tanh$ is used as the non-linear activation function.

In order to predict a label for each token, we use either a softmax or CRF output architecture. For softmax, the model directly predicts a normalised distribution over all possible labels for every word, conditioned on the vector d_t:

$$P(y_t|d_t) = softmax(W_o d_t) = \frac{e^{W_{o,k} d_t}}{\sum_{\tilde{k} \in K} e^{W_{o,\tilde{k}} d_t}} \quad (5)$$

where K is the set of all possible labels, and $W_{o,k}$ is the k-th row of output weight matrix W_o. The model is optimised by minimising categorical crossentropy, which is equivalent to minimising the negative log-probability of the correct labels:

$$E = -\sum_{t=1}^{T} log(P(y_t|d_t)) \quad (6)$$

While this architecture returns predictions based on all words in the input, the labels are still predicted independently. For some tasks, such as named entity recognition with a BIO[1] scheme, there are strong dependencies between subsequent labels and it can be beneficial to explicitly model these connections. The output of the architecture can be modified to include a Conditional Random Field (CRF, Lafferty et al. (2001)), which allows the network to look for the most optimal path through all possible label sequences (Huang et al., 2015; Lample et al., 2016). The model is then optimised by maximising the score for the correct label sequence, while minimising the scores for all other sequences:

$$E = -s(y) + log \sum_{\tilde{y} \in \tilde{Y}} e^{s(\tilde{y})} \quad (7)$$

where $s(y)$ is the score for a given sequence y and Y is the set of all possible label sequences.

We also make use of the character-level component described by Rei et al. (2016), which builds an alternative representation for each word. The individual characters of a word are mapped to character embeddings and passed through a bidirectional LSTM. The last hidden states from both direction are concatenated and passed through a

[1] Each NER entity has sub-tags for Beginning, Inside and Outside

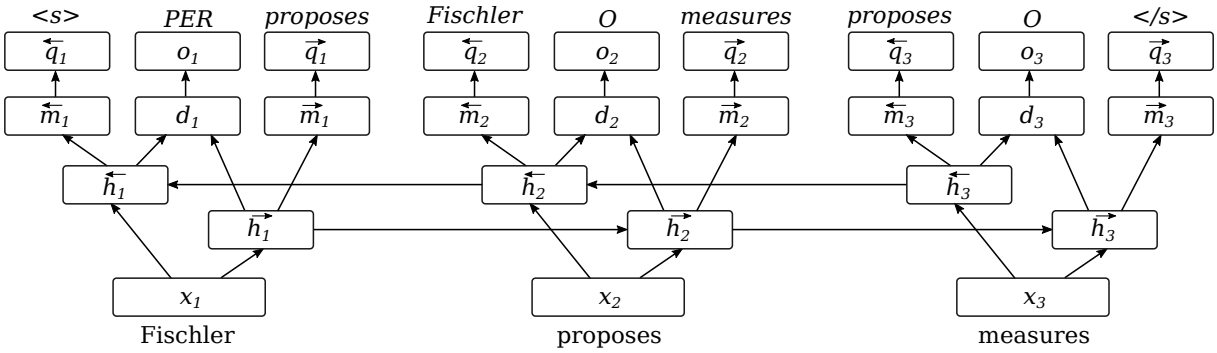


Figure 1: The unfolded network structure for a sequence labeling model with an additional language modeling objective, performing NER on the sentence *"Fischler proposes measures"*. The input tokens are shown at the bottom, the expected output labels are at the top. Arrows above variables indicate the directionality of the component (forward or backward).

nonlinear layer. The resulting vector representation is combined with a regular word embedding using a dynamic weighting mechanism that adaptively controls the balance between word-level and character-level features. This framework allows the model to learn character-based patterns and handle previously unseen words, while still taking full advantage of the word embeddings.

3 Language Modeling Objective

The sequence labeling model in Section 2 is only optimised based on the correct labels. While each token in the input does have a desired label, many of these contribute very little to the training process. For example, in the CoNLL 2003 NER dataset (Tjong Kim Sang and De Meulder, 2003) there are only 8 possible labels and 83% of the tokens have the label O, indicating that no named entity is detected. This ratio is even higher for error detection, with 86% of all tokens containing no errors in the FCE dataset (Yannakoudakis et al., 2011). The sequence labeling models are able to learn this bias in the label distribution without obtaining much additional information from the majority labels. Therefore, we propose a supplementary objective which would allow the models to make full use of the training data.

In addition to learning to predict labels for each word, we propose optimising specific sections of the architecture as language models. The task of predicting the next word will require the model to learn more general patterns of semantic and syntactic composition, which can then be reused in order to predict individual labels more accurately. This objective is also generalisable to any sequence labeling task and dataset, as it requires no additional annotated training data.

A straightforward modification of the sequence labeling model would add a second parallel output layer for each token, optimising it to predict the next word. However, the model has access to the full context on each side of the target token, and predicting information that is already explicit in the input would not incentivise the model to learn about composition and semantics. Therefore, we must design the loss objective so that only sections of the model that have not yet observed the next word are optimised to perform the prediction. We solve this by predicting the next word in the sequence only based on the hidden representation $\overrightarrow{h_t}$ from the forward-moving LSTM. Similarly, the previous word in the sequence is predicted based on $\overleftarrow{h_t}$ from the backward-moving LSTM. This architecture avoids the problem of giving the correct answer as an input to the language modeling component, while the full framework is still optimised to predict labels based on the whole sentence.

First, the hidden representations from forward- and backward-LSTMs are mapped to a new space using a non-linear layer:

$$\overrightarrow{m_t} = tanh(\overrightarrow{W}_m \overrightarrow{h_t}) \quad (8)$$

$$\overleftarrow{m_t} = tanh(\overleftarrow{W}_m \overleftarrow{h_t}) \quad (9)$$

where $\overrightarrow{W}_m$ and $\overleftarrow{W}_m$ are weight matrices. This separate transformation learns to extract features that are specific to language modeling, while the LSTM is optimised for both objectives. We also use the opportunity to map the representation to a smaller size – since language modeling is not the

main goal, we restrict the number of parameters available for this component, forcing the model to generalise more using fewer resources.

These representations are then passed through softmax layers in order to predict the preceding and following word:

$$P(w_{t+1}|\overrightarrow{m_t}) = softmax(\overrightarrow{W}_q\overrightarrow{m_t}) \quad (10)$$

$$P(w_{t-1}|\overleftarrow{m_t}) = softmax(\overleftarrow{W}_q\overleftarrow{m_t}) \quad (11)$$

The objective function for both components is then constructed as a regular language modeling objective, by calculating the negative log-likelihood of the next word in the sequence:

$$\overrightarrow{E} = -\sum_{t=1}^{T-1} log(P(w_{t+1}|\overrightarrow{m_t})) \quad (12)$$

$$\overleftarrow{E} = -\sum_{t=2}^{T} log(P(w_{t-1}|\overleftarrow{m_t})) \quad (13)$$

Finally, these additional objectives are combined with the training objective E from either Equation 6 or 7, resulting in a new cost function $\widetilde{E}$ for the sequence labeling model:

$$\widetilde{E} = E + \gamma(\overrightarrow{E} + \overleftarrow{E}) \quad (14)$$

where γ is a parameter that is used to control the importance of the language modeling objective in comparison to the sequence labeling objective.

Figure 1 shows a diagram of the unfolded neural architecture, when performing NER on a short sentence with 3 words. At each token position, the network is optimised to predict the previous word, the current label, and the next word in the sequence. The added language modeling objective encourages the system to learn richer feature representations that are then reused for sequence labeling. For example, $\overrightarrow{h_1}$ is optimised to predict *proposes* as the next word, indicating that the current word is a subject, possibly a named entity. In addition, $\overleftarrow{h_2}$ is optimised to predict *Fischler* as the previous word and these features are used as input to predict the *PER* tag at o_1.

The proposed architecture introduces 4 additional parameter matrices that are optimised during training: $\overrightarrow{W}_m$, $\overleftarrow{W}_m$, $\overrightarrow{W}_q$, and $\overleftarrow{W}_q$. However, the computational complexity and resource requirements of this model during sequence labeling are equal to the baseline from Section 2, since the language modeling components are ignored during testing and these additional weight matrices are not used. While our implementation uses a basic softmax as the output layer for the language modeling components, the efficiency during training could be further improved by integrating noise-contrastive estimation (NCE, Mnih and Teh (2012)) or hierarchical softmax (Morin and Bengio, 2005).

4 Evaluation Setup

The proposed architecture was evaluated on 10 different sequence labeling datasets, covering the tasks of error detection, NER, chunking, and POS-tagging. The word embeddings in the model were initialised with publicly available pretrained vectors, created using word2vec (Mikolov et al., 2013). For general-domain datasets we used 300-dimensional embeddings trained on Google News.[2] For biomedical datasets, the word embeddings were initialised with 200-dimensional vectors trained on PubMed and PMC.[3]

The neural network framework was implemented using Theano (Al-Rfou et al., 2016) and we make the code publicly available online.[4] For most of the hyperparameters, we follow the settings by Rei et al. (2016) in order to facilitate direct comparison with previous work. The LSTM hidden layers are set to size 200 in each direction for both word- and character-level components. All digits in the text were replaced with the character 0; any words that occurred only once in the training data were replaced by an OOV token. In order to reduce computational complexity in these experiments, the language modeling objective predicted only the 7,500 most frequent words, with an extra token covering all the other words.

Sentences were grouped into batches of size 64 and parameters were optimised using AdaDelta (Zeiler, 2012) with default learning rate 1.0. Training was stopped when performance on the development set had not improved for 7 epochs. Performance on the development set was also used to select the best model, which was then evaluated on the test set. In order to avoid any outlier results due to randomness in the model initialisa-

[2] https://code.google.com/archive/p/word2vec/
[3] http://bio.nlplab.org/
[4] https://github.com/marekrei/sequence-labeler

	FCE DEV	FCE TEST			CoNLL-14 TEST1			CoNLL-14 TEST2		
	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Baseline	48.78	55.38	25.34	44.56	15.65	16.80	15.80	25.22	19.25	23.62
+ dropout	48.68	54.11	23.33	42.65	14.29	17.13	14.71	22.79	19.42	21.91
+ LMcost	**53.17**	**58.88**	**28.92**	**48.48**	**17.68**	**19.07**	**17.86**	**27.62**	**21.18**	**25.88**

Table 1: Precision, Recall and $F_{0.5}$ score of alternative sequence labeling architectures on error detection datasets. Dropout and LMcost modifications are added incrementally to the baseline.

tion, each configuration was trained with 10 different random seeds and the averaged results are presented in this paper. We use previously established splits for training, development and testing on each of these datasets.

Based on development experiments, we found that error detection was the only task that did not benefit from having a CRF module at the output layer – since the labels are very sparse and the dataset contains only 2 possible labels, explicitly modeling state transitions does not improve performance on this task. Therefore, we use a softmax output for error detection experiments and CRF on all other datasets.

The publicly available sequence labeling system by Rei et al. (2016) is used as the baseline. During development we found that applying dropout (Srivastava et al., 2014) on word embeddings improves performance on nearly all datasets, compared to this baseline. Therefore, element-wise dropout was applied to each of the input embeddings with probability 0.5 during training, and the weights were multiplied by 0.5 during testing. In order to separate the effects of this modification from the newly proposed optimisation method, we report results for three different systems: 1) the publicly available baseline, 2) applying dropout on top of the baseline system, and 3) applying both dropout and the novel multitask objective from Section 3.

Based on development experiments we set the value of γ, which controls the importance of the language modeling objective, to 0.1 for all experiments throughout training. Since context prediction is not part of the main evaluation of sequence labeling systems, we expected the additional objective to mostly benefit early stages of training, whereas the model would later need to specialise only towards assigning labels. Therefore, we also performed experiments on the development data where the value of γ was gradually decreased, but found that a small static value performed comparably well or even better. These experiments indicate that the language modeling objective helps the network learn general-purpose features that are useful for sequence labeling even in the later stages of training.

5 Error Detection

We first evaluate the sequence labeling architectures on the task of error detection – given a sentence written by a language learner, the system needs to detect which tokens have been manually tagged by annotators as being an error. As the first benchmark, we use the publicly released First Certificate in English (FCE, Yannakoudakis et al. (2011)) dataset, containing 33,673 manually annotated sentences. The texts were written by learners during language examinations in response to prompts eliciting free-text answers and assessing mastery of the upper-intermediate proficiency level. In addition, we evaluate on the CoNLL 2014 shared task dataset (Ng et al., 2014), which has been converted to an error detection task. This contains 1,312 sentences, written by higher-proficiency learners on more technical topics. They have been manually corrected by two separate annotators, and we report results on each of these annotations. For both datasets we use the FCE training set for model optimisation and results on the CoNLL-14 dataset indicate out-of-domain performance. Rei and Yannakoudakis (2016) present results on these datasets using the same setup, along with evaluating the top shared task submissions on the task of error detection. As the main evaluation metric, we use the $F_{0.5}$ measure, which is consistent with previous work and was also adopted by the CoNLL-14 shared task.

Table 1 contains results for the three different sequence labeling architectures on the error detection datasets. We found that including the dropout actually decreases performance in the setting of

	CoNLL-00		CoNLL-03		CHEMDNER		JNLPBA	
	DEV	TEST	DEV	TEST	DEV	TEST	DEV	TEST
Baseline	92.92	92.67	90.85	85.63	83.63	84.51	77.13	72.79
+ dropout	93.40	93.15	91.14	86.00	84.78	85.67	77.61	73.16
+ LMcost	**94.22**	**93.88**	**91.48**	**86.26**	**85.45**	**86.27**	**78.51**	**73.83**

Table 2: Performance of alternative sequence labeling architectures on NER and chunking datasets, measured using CoNLL standard entity-level F_1 score.

error detection, which is likely due to the relatively small amount of error examples available in the dataset – it is better for the model to memorise them without introducing additional noise in the form of dropout. However, we did verify that dropout indeed gives an improvement in combination with the novel language modeling objective. Because the model is receiving additional information at every token, dropout is no longer obscuring the limited training data but instead helps with generalisation.

The bottom row shows the performance of the language modeling objective when added on top of the baseline model, along with dropout on word embeddings. This architecture outperforms the baseline on all benchmarks, increasing both precision and recall, and giving a 3.9% absolute improvement on the FCE test set. These results also improve over the previous best results by Rei and Yannakoudakis (2016) and Rei et al. (2016), when all systems are trained on the same FCE dataset. While the added components also require more computation time, the difference is not excessive – one training batch over the FCE dataset was processed in 112 seconds on the baseline system and 133 seconds using the language modeling objective.

Error detection is the task where introducing the additional cost objective gave the largest improvement in performance, for a few reasons:

1. This task has very sparse labels in the datasets, with error tokens very infrequent and far apart. Without the language modeling objective, the network has very little use for all the available words that contain no errors.

2. There are only two possible labels, correct and incorrect, which likely makes it more difficult for the model to learn feature detectors for many different error types. Language modeling uses a much larger number of possible labels, giving a more varied training signal.

3. Finally, the task of error detection is directly related to language modeling. By learning a better model of the overall text in the training corpus, the system can more easily detect any irregularities.

We also analysed the performance of the different architectures during training. Figure 2 shows the $F_{0.5}$ score on the development set for each model after every epoch over the training data. The baseline model peaks quickly, followed by a gradual drop in performance, which is likely due to overfitting on the available data. Dropout provides an effective regularisation method, slowing down the initial performance but preventing the model from overfitting. The added language modeling objective provides a substantial improvement – the system outperforms other configurations already in the early stages of training and the results are also sustained in the later epochs.

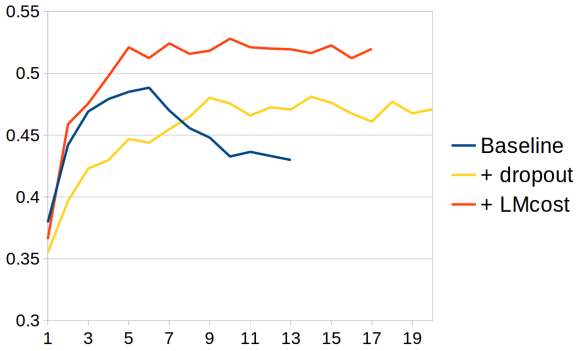


Figure 2: $F_{0.5}$ score on the FCE development set after each training epoch.

6 NER and Chunking

In the next experiments we evaluate the language modeling objective on named entity recognition and chunking. For general-domain NER, we use

	GENIA-POS		PTB-POS		UD-ES		UD-FI	
	DEV	TEST	DEV	TEST	DEV	TEST	DEV	TEST
Baseline	98.69	98.61	97.23	97.24	96.38	95.99	95.02	94.80
+ dropout	98.79	98.71	97.36	97.30	96.51	96.16	95.88	95.60
+ LMcost	**98.89**	**98.81**	**97.48**	**97.43**	**96.62**	**96.21**	**96.14**	**95.88**

Table 3: Accuracy of different sequence labeling architectures on POS-tagging datasets.

the English section of the CoNLL 2003 corpus (Tjong Kim Sang and De Meulder, 2003), containing news stories from the Reuters Corpus. We also report results on two biomedical NER datasets: The BioCreative IV Chemical and Drug corpus (CHEMDNER, Krallinger et al. (2015)) of 10,000 abstracts, annotated for mentions of chemical and drug names, and the JNLPBA corpus (Kim et al., 2004) of 2,404 abstracts annotated for mentions of different cells and proteins. Finally, we use the CoNLL 2000 dataset (Tjong Kim Sang and Buchholz, 2000), created from the Wall Street Journal Sections 15-18 and 20 from the Penn Treebank, for evaluating sequence labeling on the task of chunking. The standard CoNLL entity-level F_1 score is used as the main evaluation metric.

Compared to error detection corpora, the labels are more balanced in these datasets. However, majority labels still exist: roughly 83% of the tokens in the NER datasets are tagged as "O", indicating that the word is not an entity, and the NP label covers 53% of tokens in the chunking data.

Table 2 contains results for evaluating the different architectures on NER and chunking. On these tasks, the application of dropout provides a consistent improvement – applying some variance onto the input embeddings results in more robust models for NER and chunking. The addition of the language modeling objective consistently further improves performance on all benchmarks.

While these results are comparable to the respective state-of-the-art results on most datasets, we did not fine-tune hyperparameters for any specific task, instead providing a controlled analysis of the language modeling objective in different settings. For JNLPBA, the system achieves 73.83% compared to 72.55% by Zhou and Su (2004) and 72.70% by Rei et al. (2016). On CoNLL-03, Lample et al. (2016) achieve a considerably higher result of 90.94%, possibly due to their use of specialised word embeddings and a custom version of LSTM. However, our system does outperform a similar architecture by Huang et al. (2015), achieving 86.26% compared to 84.26% F_1 score on the CoNLL-03 dataset.

Figure 3 shows F_1 on the CHEMDNER development set after every training epoch. Without dropout, performance peaks quickly and then trails off as the system overfits on the training set. Using dropout, the best performance is sustained throughout training and even slightly improved. Finally, adding the language modeling objective on top of dropout allows the system to consistently outperform the other architectures.

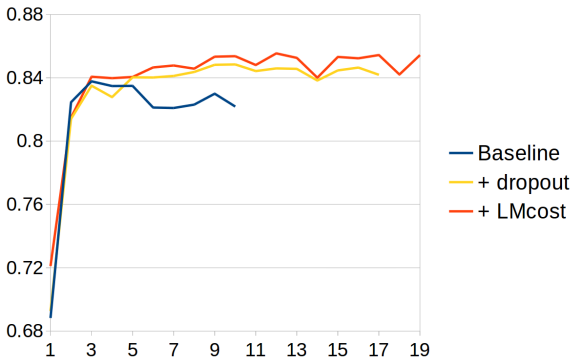


Figure 3: Entity-level F_1 score on the CHEMDNER development set after each training epoch.

7 POS tagging

We also evaluated the language modeling training objective on four POS-tagging datasets. The Penn Treebank POS-tag corpus (Marcus et al., 1993) contains texts from the Wall Street Journal and has been annotated with 48 different part-of-speech tags. In addition, we use the POS-annotated subset of the GENIA corpus (Ohta et al., 2002) containing 2,000 biomedical PubMed abstracts. Following Tsuruoka et al. (2005), we use the same 210-document test set. Finally, we also evaluate on the Finnish and Spanish sections of the Universal Dependencies v1.2 dataset (UD, Nivre et al. (2015)), in order to investigate performance on morphologically complex and Romance languages.

These datasets are somewhat more balanced in terms of label distributions, compared to error detection and NER, as no single label covers over 50% of the tokens. POS-tagging also offers a large variance of unique labels, with 48 labels in PTB and 42 in GENIA, and this can provide useful information to the models during training. The baseline performance on these datasets is also close to the upper bound, therefore we expect the language modeling objective to not provide much additional benefit.

The results of different sequence labeling architectures on POS-tagging can be seen in Table 3 and accuracy on the development set is shown in Figure 4. While the performance improvements are small, they are consistent across all domains, languages and datasets. Application of dropout again provides a more robust model, and the language modeling cost improves the performance further. Even though the labels already offer a varied training objective, learning to predict the surrounding words at the same time has provided the model with additional general-purpose features.

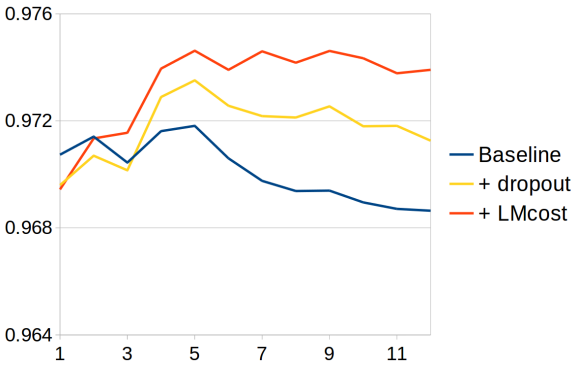


Figure 4: Token-level accuracy on the PTB-POS development set after each training epoch.

8 Related Work

Our work builds on previous research exploring multi-task learning in the context of different sequence labeling tasks. The idea of multi-task learning was described by Caruana (1998) and has since been extended to many language processing tasks using neural networks. For example, Collobert and Weston (2008) proposed a multi-task framework using weight-sharing between networks that are optimised for different supervised tasks.

Cheng et al. (2015) described a system for detecting out-of-vocabulary names by also predicting the next word in the sequence. While they use a regular recurrent architecture, we propose a language modeling objective that can be integrated with a bidirectional network, making it applicable to existing state-of-the-art sequence labeling frameworks.

Plank et al. (2016) described a related architecture for POS-tagging, predicting the frequency of each word together with the part-of-speech, and showed that this can improve tagging accuracy on low-frequency words. While predicting word frequency can be useful for POS-tagging, language modeling provides a more general training signal, allowing us to apply the model to many different sequence labeling tasks.

Recently, Augenstein and Søgaard (2017) explored multi-task learning for classifying keyphrase boundaries, by incorporating tasks such as semantic super-sense tagging and identification of multi-word expressions. Bingel and Søgaard (2017) also performed a systematic comparison of task relationships by combining different datasets through multi-task learning. Both of these approaches involve switching to auxiliary datasets, whereas our proposed language modeling objective requires no additional data.

9 Conclusion

We proposed a novel sequence labeling framework with a secondary objective – learning to predict surrounding words for each word in the dataset. One half of a bidirectional LSTM is trained as a forward-moving language model, whereas the other half is trained as a backward-moving language model. At the same time, both of these are also combined, in order to predict the most probable label for each word. This modification can be applied to several common sequence labeling architectures and requires no additional annotated or unannotated data.

The objective of learning to predict surrounding words provides an additional source of information during training. The model is incentivised to discover useful features in order to learn the language distribution and composition patterns in the training data. While language modeling is not the main goal of the system, this additional training objective leads to more accurate sequence labeling models on several different tasks.

The architecture was evaluated on a range of datasets, covering the tasks of error detection in

learner texts, named entity recognition, chunking and POS-tagging. We found that the additional language modeling objective provided consistent performance improvements on every benchmark. The largest benefit from the new architecture was observed on the task of error detection in learner writing. The label distribution in the original dataset is very sparse and unbalanced, making it a difficult task for the model to learn. The added language modeling objective allowed the system to take better advantage of the available training data, leading to 3.9% absolute improvement over the previous best architecture. The language modeling objective also provided consistent improvements on other sequence labeling tasks, such as named entity recognition, chunking and POS-tagging.

Future work could investigate the extension of this architecture to additional unannotated resources. Learning generalisable language features from large amounts of unlabeled in-domain text could provide sequence labeling models with additional benefit. While it is common to pre-train word embeddings on large-scale unannotated corpora, only limited research has been done towards useful methods for pre-training or co-training more advanced compositional modules.

References

Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, and Others. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.0:19. http://arxiv.org/abs/1605.02688.

Isabelle Augenstein and Anders Søgaard. 2017. Multi-Task Learning of Keyphrase Boundary Classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. http://arxiv.org/abs/1704.00514.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *arXiv preprint*. http://arxiv.org/abs/1702.08303.

Rich Caruana. 1998. *Multitask Learning*. Ph.D. thesis.

Ciprian Chelba, Tomáš Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In *arXiv preprint*. http://arxiv.org/abs/1312.3005.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-Domain Name Error Detection using a Multi-Task RNN. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *Proceedings of the 25th international conference on Machine learning (ICML '08)* https://doi.org/10.1145/1390156.1390177.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research* 12. https://doi.org/10.1145/2347736.2347755.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* https://doi.org/10.1109/ICASSP.2013.6638947.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation* 9. https://doi.org/10.1.1.56.7752.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv:1508.01991* http://arxiv.org/pdf/1508.01991v1.pdf.

Ozan Irsoy and Claire Cardie. 2014. Opinion Mining with Deep Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the Bio-entity Recognition Task at JNLPBA. *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications* https://doi.org/10.3115/1567594.1567610.

Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. 2015. CHEMDNER: The drugs and chemical names extraction challenge. *Journal of Cheminformatics* 7(Suppl 1). https://doi.org/10.1186/1758-2946-7-S1-S1.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT 2016*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19. https://doi.org/10.1162/coli.2010.36.1.36100.

Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)*. https://doi.org/10.1162/153244303322533223.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network based Language Model. *Interspeech* (September):1045–1048.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Neural Information Processing Systems (NIPS)*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics* https://doi.org/10.1109/JCDL.2003.1204852.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. http://www.aclweb.org/anthology/W/W14/W14-1701.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, et al. 2015. Universal dependencies 1.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11234/1-1548.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. *Proceedings of the second international conference on Human Language Technology Research* http://portal.acm.org/citation.cfm?id=1289260.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*. pages 412–418. http://arxiv.org/abs/1604.05529.

Marek Rei, Gamal K. O. Crichton, and Sampo Pyysalo. 2016. Attending to Characters in Neural Sequence Labeling Models. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING-2016)*. http://arxiv.org/abs/1611.04361.

Marek Rei and Helen Yannakoudakis. 2016. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. https://aclweb.org/anthology/P/P16/P16-1112.pdf.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)* 15. https://doi.org/10.1214/12-AOS1000.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning* 7. https://doi.org/10.3115/1117601.1117631.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*. http://arxiv.org/abs/cs/0306050.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In *Proceedings of Panhellenic Conference on Informatics*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. http://www.aclweb.org/anthology/P11-1019.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701* http://arxiv.org/abs/1212.5701.

GuoDong Zhou and Jian Su. 2004. Exploring Deep Knowledge Resources in Biomedical Name Recognition. *Workshop on Natural Language Processing in Biomedicine and Its Applications at COLING* .

Semantic Parsing of Pre-university Math Problems

Takuya Matsuzaki[1], **Takumi Ito**[1], **Hidenao Iwane**[2], **Hirokazu Anai**[2], **Noriko H. Arai**[3]
[1] Nagoya University, Japan
{matuzaki,takumi_i}@nuee.nagoya-u.ac.jp
[2] Fujitsu Laboratories Ltd., Japan
{iwane,anai}@jp.fujitsu.com
[3] National Institute of Informatics, Japan
arai@nii.ac.jp

Abstract

We have been developing an end-to-end math problem solving system that accepts natural language input. The current paper focuses on how we analyze the problem sentences to produce logical forms. We chose a hybrid approach combining a shallow syntactic analyzer and a manually-developed lexicalized grammar. A feature of the grammar is that it is extensively typed on the basis of a formal ontology for pre-university math. These types are helpful in semantic disambiguation inside and across sentences. Experimental results show that the hybrid system produces a well-formed logical form with 88% precision and 56% recall.

1 Introduction

Frege and Russell, the initiators of the mathematical logic, delved also into the exploration of a theory of natural language semantics (Frege, 1892; Russell, 1905). Since then, symbolic logic has been a fundamental tool and a source of inspiration in the study of language meaning. It suggests that the formalization of the two realms, mathematical reasoning and language meaning, is actually the two sides of the same coin – probably, we could not even conceive the idea of formalizing language meaning without *grounding it* onto mathematical reasoning. This point was first clarified by Tarski (1936; 1944) mainly on formal languages and then extended to natural languages by Davidson (1967). Montague (1970a; 1970b; 1973) further embodied it by putting forward a terrifyingly arrogant and attractive idea of seeing a natural language *as* a formal language.

The automation of end-to-end math problem solving thus has an outstanding status in the research themes in natural language processing. The conceptual basis has been laid down, which connects text to the truth (= answer) through reasoning. However, we have not seen a fully automated system that instantiates it end-to-end. We wish to add a piece to the big picture by materializing it.

> Define the two straight lines L_1 and L_2 on the xy-plane as L_1: $y = 0$ (x-axis) and L_2: $y = \sqrt{3}x$. Let P be a point on the xy-plane. Let Q be the point symmetric to P about the straight line L_1, and let R be the point symmetric to P about the straight line L_2. Answer the following questions:
>
> (1) Let (a, b) be the coordinates of P, then represent the coordinates of R using a and b.
>
> (2) Assuming that the distance between the two points Q and R is 2, find the locus C of P.
>
> (3) When the point P moves on C, find the maximum area of the triangle PQR and the coordinates of P that gives the maximum area.
>
> (Hokkaido Univ., 1999-Sci-3)

Figure 1: Example problem

Past studies have mainly targeted at primary school level arithmetic word problems (Bobrow, 1964; Charniak, 1969; Kushman et al., 2014; Hosseini et al., 2014; Shi et al., 2015; Roy and Roth, 2015; Zhou et al., 2015; Koncel-Kedziorski et al., 2015; Mitra and Baral, 2016; Upadhyay et al., 2016). In their nature, arithmetic questions are quantifier-free. Moreover they tend to include only $\wedge$ (and) as the logical connective. The main challenge in these works was to extract simple numerical relations (most typically equations) from a real-world scenario described in a text.

Seo et al. (2015) took SAT geometry questions as their benchmark. However, the nature of SAT geometry questions restricts the resulting formula's complexity. In §3, we will show that none of them includes $\forall$ (for all), $\vee$ (or) or $\rightarrow$ (implies). It suggests that this type of questions require little need to analyze the logical structure of the problems beyond conjunctions of predicates.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 2131–2141
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1195

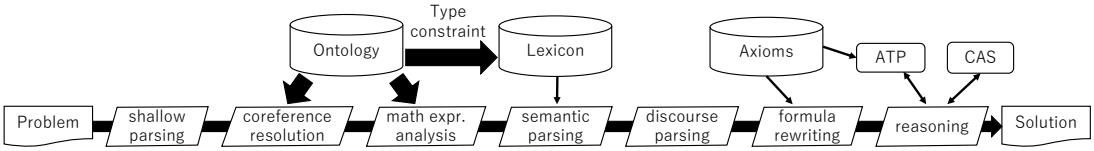


Figure 2: Overview of the end-to-end math problem solving system

We take pre-university math problems falling in the theory of real-closed fields (RCF) as our benchmark because of their variety and complexity. The subject areas include real and linear algebra, complex numbers, calculus, and geometry. Furthermore, many problems involve more than one subject: e.g., algebraic curves and calculus as in Fig. 1. Their logical forms include all the logical connectives, quantifiers, and λ-abstraction. Our goal is to recognize the complex logical structures precisely, including the scopes of the quantifiers and other logical operators.

In the rest of the paper, we first present an overview of an end-to-end problem solving system (§2) and analyze the complexity of the pre-university math benchmark in comparison with others (§3). Among the modules in the end-to-end system, we focus on the sentence-level semantic parsing component and describe an extensively-typed grammar (§4 and §5), an analyzer for the math expressions in the text (§6), and two semantic parsing techniques to fight against the scarcity of the training data (§7) and the complexity of the domain (§8). Experimental results show the effectiveness of the presented techniques as well as the complexity of the task through an in-depth analysis of the end-to-end problem solving results (§9).

2 End-to-end Math Problem Solving

Fig. 2 presents an overview of our end-to-end math problem solving system. A math problem text is firstly analyzed with a dependency parser. Anaphoric and coreferential expressions in the text are then identified and their antecedents are determined. We assume the math formulas in the problems are encoded in MathML presentation mark-up. A specialized parser processes each one of them to determine its syntactic category and semantic content. The semantic representation of each sentence is determined by a semantic parser based on Combinatory Categorial Grammar (CCG) (Steedman, 2001, 2012). The output from the CCG parser is a ranked list of sentence-level logical forms for each sentence.

Dataset	Succeeded		Failed	
	Success%	Avg. Time	Timeout	Other
DEV	75.3% (131/174)	10.5s	16.7%	8.1%
TEST	78.2% (172/220)	16.2s	15.0%	6.8%

Table 1: Performance of the reasoning module on *manually* formalized pre-university problems

After the sentence-level processing steps, we determine the logical relations among the sentence-level logical forms (discourse parsing) by a simple rule-based system. It produces a tree structure whose leaves are labeled with sentences and internal nodes with logical connectives. Free variables in the logical form are then bound by some quantifiers (or kept free) and their scopes are determined according to the logical structure of the problem. A semantic representation of a problem is obtained as a formula in a higher-order logic through these language analysis steps.

The logical representation is then rewritten using a set of axioms that define the meanings of the predicate and function symbols in the formula, such as `maximum` defined as follows:

$$\texttt{maximum}(x, S) \leftrightarrow x \in S \wedge \forall y(y \in S \rightarrow y \leq x),$$

as well as several logical rules such as β-reduction. We hope to obtain a representation of the initial problem expressed in a decidable math theory such as RCF through these equivalence-preserving rewriting. Once we find such a formula, we invoke a computer algebra system (CAS) or an automatic theorem prover (ATP) to derive the answer.

The reasoning module (i.e., the formula rewriting and the deduction with CAS and ATP) of the system has been extensively tested on a large collection of manually formalized pre-university math problems that includes more than 1,500 problems. It solves 70% of the them in the time limit of 10 minutes per problem. Table 1 shows the rate of successfully solved problems in the *manually* formalized version of the benchmark problems used in the current paper.

	Prob-lems	Avg. tokens	Avg. sents.	Uniq. words	Atoms	$\exists$	$\forall$	λ	$\wedge$	$\vee$	$\neg$	$\rightarrow$	Unique sketches
JOBS	640	9.83	1.00	391	4.63	1.71	0.00	0.00	1.06	0.01	0.13	0.00	8
GEOQUERY	880	8.56	1.00	284	4.25	1.70	0.00	1.04	1.18	0.00	0.02	0.00	20
GEOMETRY	119	23.64	1.74	202	11.00	7.45	0.00	0.06	1.00	0.00	0.04	0.00	4
UNIV (DEV)	174	70.34	3.45	363	10.99	5.10	1.10	1.11	1.71	0.02	0.49	0.35	76
UNIV (TEST)	220	70.85	4.02	366	9.70	4.58	1.10	1.00	1.62	0.02	0.28	0.23	72

Table 2: Profile of pre-university math benchmark data and other semantic parsing benchmark data sets

JOBS		GEOQUERY		GEOMETRY		UNIV (DEV)	
$\exists P$	81%	$\exists P$	46%	$\exists P$	94%	$\exists P$	25%
P	6%	$\exists P(\lambda \exists P)$	24%	$\exists(P \wedge \neg P)$	3%	$\exists(P \wedge \neg P)$	7%
$\exists(P \wedge \neg \exists P)$	5%	$P(\lambda \exists P)$	8%	$\exists(P \wedge P(\lambda P))$	2%	$P(\lambda \exists P)$	5%
$\exists(P \wedge \neg P)$	5%	$\exists(P \wedge P(\lambda \exists P))$	7%	$P(\lambda \exists P)$	1%	$\exists(P \wedge P(\lambda f))$	4%
	97%		85%		100%		41%

Table 3: Top four most frequest sketches and their coverage over the dataset

Sketch	Freq.
$\forall(P \rightarrow \exists(\forall(P \rightarrow P) \wedge P))$	2
$\exists(\exists(\neg P \wedge P) \wedge P \wedge P(\lambda f)) \wedge P(\lambda(P \rightarrow P)))$	1
$\exists(P \wedge P(\lambda(\neg P \wedge \exists(\exists P \wedge P))))$	1
$\exists(P \wedge P(\lambda f)) \wedge P(\lambda(\neg P \wedge P)) \wedge P(\lambda P))$	1

Table 4: Less frequent sketches in UNIV (DEV)

3 Profile of the Benchmark Data

Our benchmark problems, UNIV, were collected from the past entrance exams of seven top-ranked universities in Japan. In the exams held in odd numbered years from 1999 to 2013, we exhaustively selected the problems which are ultimately expressible in RCF. They occupied 40% of all the problems. We divided the problems into two sets: DEV for development (those from year 1999 to 2005) and TEST for test (those from year 2007 to 2013). DEV was used for the lexicon development and the tuning of the end-to-end system. The problem texts (both in English and Japanese) with MathML mark-up and manually translated logical forms are publicly available at https://github.com/torobomath.

The manually translated logical forms were formulated in a higher-order semantic language introduced later in the paper. The translation was done as faithfully as possible to the original wordings of the problems. They thus keep the inherent logical structures expressed in natural language.

Table 2 lists several statistics of the UNIV problems in the English version and their manual formalization. For comparison, the statistics of three other benchmarks are also listed. JOBS and GEOQUERY are collections of natural language queries against databases. They have been widely used as benchmarks for semantic parsing (e.g., Tang and Mooney, 2001; Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010, 2011; Liang et al., 2011). The queries are annotated with logical forms in Prolog. We converted them to equivalent higher-order formulas to collect comparable statistics. GEOMETRY is a collection of SAT geometry questions compiled by Seo et al. (2015). We formalized the GEOMETRY questions[1] in our semantic language in the same way as UNIV.

In Table 2, the first column lists the number of problems. The next three provide statistics of the problem texts: average number of words and sentences in a problem ('Avg. tokens' and 'Avg. sents'), and the number of unique words in the whole dataset.[2] They reveal that the sentences in UNIV are significantly longer than the others and more than three sentences have to be correctly processed for a problem.

The remaining columns provide the statistics about the logical complexities of the problems. 'Atoms' stands for the average number of the occurrences of predicates per problem. The next three columns list the number of variables bound by $\exists$, $\forall$, and λ. We count sequential occurrences of the same binder as one. The columns for $\wedge$, $\vee$, $\neg$, and $\rightarrow$ list the average number of them per problem.[3] We can see UNIV includes a wider variety of quantifiers and connectives than the others.

The final column lists the numbers of unique 'sketches' of the logical forms in the dataset. What

[1] Including *all* conditions expressed in the diagrams.

[2] All the math formulas in the texts were replaced with a special token "MATH" before counting words.

[3] $\wedge$ and $\vee$ was counted as operators with arbitrary arity. E.g., there is only one $\wedge$ in $A \wedge B \wedge C$.

we call 'sketch' here is a signature that encodes the overall structure of a logical form. Table 3 shows the top four most frequent sketches observed in the datasets. In a sketch, P stands for a (conjunction of) predicate(s) and f stands for a term. $\exists$, $\forall$, and λ stand for (immediately nested sequence of) the binders.

To obtain the sketch of a formula ϕ, we first replace all the predicate symbols in ϕ to P and function symbols and constants to f. We then eliminate all variables in ϕ and 'flatten' it by applying the following rewriting rules to the sub-formulas in ϕ in the bottom-up order:

$$f(..., f(\alpha_1, \alpha_2, ..., \alpha_n), ...) \Rightarrow f(..., \alpha_1, \alpha_2, ..., \alpha_n, ...)$$
$$P(..., f(\alpha_1, \alpha_2, ..., \alpha_n), ...) \Rightarrow P(..., \alpha_1, \alpha_2, ..., \alpha_n, ...)$$
$$\alpha \vee \alpha \vee \beta \Rightarrow \alpha \vee \beta, \quad \alpha \wedge \alpha \Rightarrow \alpha$$
$$\exists\exists\psi \Rightarrow \exists\psi, \quad \forall\forall\psi \Rightarrow \forall\psi, \quad \lambda\lambda\psi \Rightarrow \lambda\psi$$

Finally, we sort the arguments of Ps and fs and remove the duplicates among them. For instance, to obtain the sketch of the following formula:

$$\forall k \forall m \left(\begin{array}{c} \texttt{maximum}(m, \texttt{set}(\lambda e.(e < k))) \\ \rightarrow k - 1 \leq m \wedge m < k \end{array} \right),$$

we replace the predicate/function symbols as in:

$$\forall k \forall m \left(\begin{array}{c} P(m, f(\lambda e.P(e, k))) \\ \rightarrow P(f(k, f), m) \wedge P(m, k) \end{array} \right),$$

and then eliminate the variables to have:

$$\forall\forall(P(f(\lambda P)) \rightarrow P(f(f)) \wedge P),$$

and finally flatten it to:

$$\forall(P(\lambda P) \rightarrow P).$$

Table 3 shows that a wide variety of structures are found in UNIV while other data sets are dominated by a small number of structures. Table 4 presents some of less frequent sketches found in UNIV (DEV). In actuality, 67% of the unique sketches found in UNIV (DEV) occur only once in the dataset. These statistics suggest that the distribution of the logical structures found in UNIV, and math text in general, is very long-tailed.

4 A Type System for Pre-university Math

Our semantic language is a higher-order logic (lambda calculus) with parametric polymorphism. Table 5 presents the types in the language. The atomic types are defined so that they capture the selectional restriction of verbs and other argument-taking phrases as precisely as possible. For instance, an equation in real domain, e.g., $x^2 - 1 = 0$, can be regarded as a set of reals, i.e., $\{x \mid x^2 - 1 = 0\}$. However, we never say 'a solution of a set.' We thus discriminate an equation from a set in the type system even though the concept of equation is mathematically dispensable.

truth values	`Bool`
numbers	`Z` (integers), `Q` (rationals), `R` (reals), `C` (complex)
polynomials	`Poly`
single variable functions	`R2R` ($\mathbb{R} \rightarrow \mathbb{R}$), `C2C` ($\mathbb{C} \rightarrow \mathbb{C}$)
single variable equations	`EqnR` (in $\mathbb{R}$), `EqnC` (in $\mathbb{C}$)
points in 2D/3D space	`2d.Point`, `3d.Point`
geometric objects	`2d.Shape`, `3d.Shape`
vectors and matrices	`2d.Vector`, `3d.Vector`
matrices	`2d.Matrix`, `3d.Matrix`
angles	`2d.Angle`, `3d.Angle`
number sequences	`Seq`
cardinals and ordinals	`Card`, `Ord`
ratios among numbers	`Ratio`
limit values of functions	`LimitVal`
integer division	`QuoRem`
polymorphic containers	`SetOf`(α), `ListOf`(α)
polymorphic tuples	`Pair`(α, β), `Triple`(α, β, γ)

Table 5: Types defined in the semantic language

Entities of equation and set are built by constructor functions that take a higher-order term as the argument as in `eqn`($\lambda x.x^2 - 1$) and `set`($\lambda x.x^2 - 1$). Related concepts such as 'solution' and 'element' are defined by the axioms for corresponding function and predicate symbols:

$$\forall f \forall x(\texttt{solution}(x, \texttt{eqn}(f)) \leftrightarrow fx)$$
$$\forall s \forall x(\texttt{element}(x, \texttt{set}(s)) \leftrightarrow sx).$$

Distinction of cardinal numbers (`Card`) and ordinal numbers (`Ord`), and the introduction of 'integer division' type (`QuoRem`) are also linguistically motivated. The former is necessary to capture the difference between, e.g., 'k-th integer in $n_1, n_2, \ldots, n_m$' and 'k integers in $n_1, n_2, \ldots, n_m$.' An object of type `QuoRem` is conceptually a pair of integers that represent the quotient and the remainder of integer division. It is linguistically distinct from the type of `Pair(Z,Z)` because, e.g., in

> Select a pair of integers (n, m) and divide n by m. If the remainder (of ϕ) is zero, ...

the null (i.e., omitted) pronoun ϕ has 'the result of division n/m' as its antecedent but not (n, m).

Polymorphism is a mandatory part of the language. Especially, the semantics of plural noun

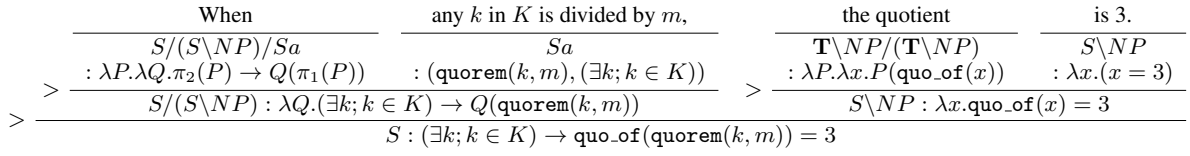


Figure 3: Sketch of the derivation tree for a sentence including an action verb and quantification

phrases is expressed by polymorphic lists and tuples: e.g., 'the radii of the circles C_1, C_2, and C_3' is of type `ListOf(R)` and 'the function f and its maximum value' is of type `Pair(R2R,R)`.

5 Lexicon and Grammar

5.1 Combinatory Categorial Grammar

An instance of CCG grammar consists of a *lexicon* and a small number of *combinatory rules*. A lexicon is a set of *lexical items*, each of which associates a word surface form with a syntactic category and a semantic function: e.g.,

$$\textit{sum} \;::\; \text{NP/PP} : \lambda x.\texttt{sum\_of}(x)$$
$$\textit{intersects} \;::\; \text{S}\backslash\text{NP/PP} : \lambda y.\lambda x.\texttt{intersect}(x, y)$$

A syntactic category is one of atomic categories, such as NP, PP, and S, or a complex category in the form of $\mathbf{X}/\mathbf{Y}$ or $\mathbf{X}\backslash\mathbf{Y}$, where $\mathbf{X}$ and $\mathbf{Y}$ are syntactic categories.

The syntactic categories and the semantic functions of constituents are combined by applying combinatory rules. The most fundamental rules are forward ($>$) and backward ($<$) application:

$$> \frac{\mathbf{X}/\mathbf{Y} : f \quad \mathbf{Y} : x}{\mathbf{X} : fx} \qquad < \frac{\mathbf{Y} : x \quad \mathbf{X}\backslash\mathbf{Y} : f}{\mathbf{X} : fx}$$

The atomic categories are further classified by *features* such as `num`(ber) and `case` of noun phrases. In the current paper, the features are written as in NP[`num=pl,case=acc`].

5.2 A Japanese CCG Grammar and Lexicon

We developed a Japanese CCG following the analysis of basic constructions by Bekki (2010) but significantly extending it by covering various phenomena related to copula verbs, action verbs, argument-taking nouns, appositions and so forth. The semantic functions are defined in the format of a higher-order version of dynamic predicate logic (Eijck and Stokhof, 2006). The dynamic property is necessary to analyze semantic phenomena related to quantifications, such as donkey anaphora. In the following examples, we use English instead of Japanese and the standard notation of higher-order logic for the sake of readability.

We added two atomic categories, Sn and Sa, to the commonly used S, NP, and N. Category Sn is assigned to a proposition expressed as a math formula, such as '$x > 0$'. Semantically it is of type `Bool` but syntactically it behaves both like a noun phrase and a sentence.

Category Sa is assigned to a sentence where the main verb is an action verb such as *add* and *rotate*. Such a sentence introduces the result of the action as a discourse entity (i.e., what can be an antecedent of coreferential expressions). The action verbs can also mediate quantification as in:

When any $k \in K$ is divided by m, the quotient is 3.
$\forall k(k \in K \rightarrow \texttt{quo\_of}(\texttt{quorem}(k,m)) = 3)$

where $\texttt{quorem}(k,m)$ represents the result of the division (i.e., the pair of the quotient and the remainder) and `quo_of` is a function that extracts the quotient from it. To handle such phenomena, we posit the semantic type of Sa as `Pair(`α`,Bool)` where the two components respectively bring the result of an action and the condition on it (including quantification). Fig. 3 presents a derivation tree for the above example.[4]

The atomic category NP, N, and Sa in our grammar have `type` feature. Its value is one of the types defined in the semantic language or a *type variable* when the entity type is underspecified. The lexical entry for '(an integer) *divides* (an integer)' and '(a set) *includes* (an element)' would thus have the following categories (other features than `type` are not shown):

$$\textit{divides} \;::\; \text{S}\backslash\text{NP[type=Z]/NP[type=Z]}$$
$$\textit{includes} \;::\; \text{S}\backslash\text{NP[type=SetOf}(\alpha)\text{]/NP[type=}\alpha\text{]}$$

When defining a lexical item, we don't have to explicitly specify the `type` features in most cases. They can be usually inferred from the definition of

[4] In Fig. 3, the semantic part is in the dynamic logic format as in our real grammar where the dynamic binding $(\exists x; \phi) \rightarrow \psi$ is interpreted as $\forall x(\phi \rightarrow \psi)$ in the standard predicate logic. Following our analysis of an analogous construction in Japanese, the null pronoun after 'the quotient' is filled by analysing the second clause as including a gap rather than filling it by zero-pronoun resolution.

the semantic function. In the above example, *divides* will have $\lambda y.\lambda x.(x|y)$ and *includes* will have $\lambda y.\lambda x.(y \in x)$ as their semantic functions. For both cases, the `type` feature of the NP arguments can be determined from the type definitions of the operators | and $\in$ in the ontology.

The lexicon currently includes 54,902 lexical items for 8,316 distinct surface forms, in which 5,094 lexical items for 1,287 surface forms are for function words and functional multi-word expressions. The number of unique categories in the lexicon is 10,635. When the `type` features are ignored, there are still 4,026 distinct categories.

6 Math Expression Analysis

The meaning of a math expression is composed with the semantic functions of surrounding words to produce a logical form. We dynamically generate lexical items for each math expression in a problem. Consider the following sentence including two 'equations':

If $a^2-4=0$, then $x^2+ax+1=0$ has a real solution.

The latter, $x^2+ax+1=0$, should receive a lexical item of a noun phrase, NP : $\texttt{eqn}(\lambda \texttt{x}.\texttt{x}^2 + \texttt{a} + 1)$, but the former, $a^2-4=0$, should receive category S since it denotes a proposition. Such disambiguation is not always possible without semantic analysis of the text. We thus generate more than one lexical item for ambiguous expressions and let the semantic parser make a choice.

To generate the lexical items, we first collect appositions to the math expressions, such as 'integer n and m' and 'equation $x^2 + a = 0$,' and use them as the type constraints on the variables and the compound expressions. Compound expressions are then parsed with an operator precedence parser (Aho et al., 2006). Overloaded operators, such as $+$ for numbers and vectors, are resolved using the type constrains whenever possible. Finally, we generate all possible interpretations of the expressions and select appropriate syntactic categories.

We have seen only three categories of math expressions: NP, Sn, and **T**/(**T**\NP). The last one is used for a NP with post-modification, as in:

$$\dfrac{\dfrac{n > 0}{\mathbf{T}/(\mathbf{T}\backslash NP) \;:\; \lambda P.(n > 0 \wedge P(n))} \quad \dfrac{\text{is an even number}}{S\backslash NP \;:\; \lambda x.(\texttt{even}(x))}}{S : n > 0 \wedge \texttt{even}(n)} \;>$$

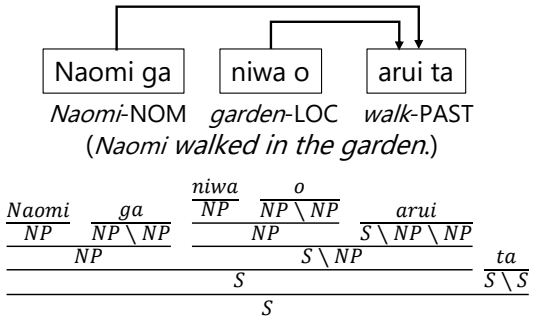


Figure 4: Bunsetsu dependency structure (top) and CCG derivation tree (bottom)

7 Two-step Semantic Parsing

Two central issues in parsing are the cost of the search and the accuracy of disambiguation. Supervised learning is commonly used to solve both. It is however very costly to create the training data by manually annotating a large number of sentences with CCG trees. Past studies have tried to bypass it by so-called weak supervision, where a parser is trained only with the logical form (e.g., Kwiatkowski et al. 2011) or even only with the answers to the queries (e.g., Liang et al. 2011).

Although the adaptation of such methods to the pre-university math data is an interesting future direction, we developed yet another approach based on a hybrid of shallow dependency parsing and the detailed CCG grammar. The syntactic structure of Japanese sentences has traditionally been analyzed based on the relations among word chunks called *bunsetsu*s. A bunsetsu consists of one or more content words followed by zero or more function words. The dependencies among bunsetsus mostly correspond to the predicate-argument and inter-clausal dependencies (Fig. 4). The dependency structure hence matches the overall structure of a CCG tree only leaving the details unspecified.

We derive a full CCG-tree by using a bunsetsu dependency tree as a constraint. We assume: (i) the fringe of each sub-tree in the dependency tree has a corresponding node in the CCG tree. We call such a node in the CCG tree 'a matching node.' We further assume: (ii) a matching node is combined with another CCG tree node whose span includes at least one word in the head bunsetsu of the matching node. Fig. 5 presents an example of a sentence consisting of four bunsetsus (rounded squares), each of which contains two words. In the figure, the i-th cell in the k-th row from the bottom is the CKY cell for the span from i-th to

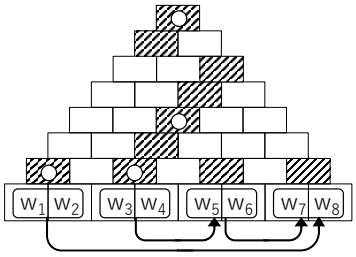


Figure 5: Restricted CKY parsing based on a shallow dependency structure

(i+k-1)-th words. Under the two assumptions, we only need to fill the hatched cells given the dependency structure shown below the CKY chart. The hatched cells with a white circle indicate the positions of the matching nodes.

Even under the constraint of a dependency tree, it is impractical to do exhaustive search. We use beam search based on a simple score function on the chart items that combines several features such as the number of atomic categories in the item. We also use N-best dependency trees to circumvent the dependency errors. The restricted CKY parsing is repeated on the N-best dependency trees until a CCG tree is obtained. Our hope is to reject a dependency error as violation of the syntactic and semantic constraints encoded in the CCG lexicon. In the experiment, we used a Japanese dependency parser developed by Kudo and Matsumoto (2002). We modified it to produce N-best outputs and used up to 20-best trees per sentence.

8 Global Type Coherency

The well-typedness of the logical form is usually guaranteed by the combinatory rules. However, they do not always guarantee the type coherency among the interpretations of the math expressions.

For instance, consider the following derivation:

$$> \frac{\dfrac{\text{if } x+y \in U,}{S/S : \lambda P.(\mathtt{add_R}(x,y) \in U \to P)} \quad \dfrac{\text{then } x+z \in V.}{S : \mathtt{add_V}(x,y) \in V}}{S : \mathtt{add_R}(x,y) \in U \to \mathtt{add_V}(x,z) \in V}$$

The $+$ symbol is interpreted as the addition of real numbers ($\mathtt{add_R}$) in the first clause but that of vectors ($\mathtt{add_V}$) in the second one. The logical form is not typable because the two occurrences of x must have different types. The forward application rule does not reject this derivation since the categories of the two clauses perfectly match the rule schema.

We can reject such inconsistency by doing type checking on the logical form at every step of the derivation. It is however quite time consuming because we cannot use dynamic programming any more and need to do type checking on numerous chart items. Furthermore, such type inconsistency may happen *across* sentences. Instead, we consider the *type environment* while parsing. A type environment, written as $\{v_1 : T_1, v_2 : T_2, \dots\}$, is a finite function from variables to type expressions. A pair $v : T$ means that the variable v must be of type T or its instance (e.g., $\mathtt{SetOf(R)}$ is an instance of $\mathtt{SetOf}(\alpha)$). For example, the logical form of the first clause of the above sentence is typable under $\{x:\mathtt{R}, y:\mathtt{R}, z:\alpha, U:\mathtt{SetOf(R)}, V:\beta\}$, but that of the second clause isn't. Please refer, e.g., to (Pierce, 2002) for the formal definitions. Two environments Γ_1 and Γ_2 are *unifiable* iff there exists a substitution σ that maps the type variables in Γ_1 and Γ_2 to some type expressions so that $\Gamma_1\sigma = \Gamma_2\sigma$ holds. We write $\Gamma_1 \sqcup \Gamma_2$ for the result of such substitution (i.e., unification) with the

Algorithm 1 Global type coherence check

procedure PARSEPROBLEM
  Envs ← ∅; *AllDerivs* ← []
  for each sentence s in the problem **do**
    Chart ← INITIALIZECKYCHART(s, *Envs*)
    Derivs ← TWOSTEPPARSING(s, *Chart*)
    Envs ← UPDATEENVIRONMENTS(*Envs*, *Derivs*)
    AllDerivs ← *AllDerivs* ⊕ [*Derivs*]
  return *AllDerivs*

// s: a sentence; *Envs*: a set of environments
procedure INITIALIZECKYCHART(s, *Envs*)
  Chart ← empty CKY chart
  for each token t in s **do**
    for each lexical item $C : f$ for t **do**
      // C: category, f: semantic function
      if t is a math expression **then**
        for each environment $\Gamma \in$ *Envs* **do**
          if Γ is unifiable with FV(f) **then**
            add $(C, \Gamma \sqcup \mathrm{FV}(f))$ to *Chart*
      else // t is a normal word
        add $(C, \emptyset)$ to *Chart*
  return *Chart*

FV(f): the environment that maps the free variables in a semantic function f to their principal types determined by type inference on f.

// *Envs*: a set of environments; *Derivs*: derivations trees
procedure UPDATEENVIRONMENTS(*Envs*, *Derivs*)
  NewEnvs ← ∅ // environments for the next sentence
  for each derivation $d \in$ *Derivs* **do**
    Γ ← the environment at the root of d
    if $\Gamma \neq \emptyset$ **then** // update the environments
      NewEnvs ← *NewEnvs* ∪{Γ}
    else // no update: there was no math expression
      NewEnvs ← *NewEnvs* ∪ *Envs*
  // eliminate those subsumed by other environments
  return MOSTGENERALENVIRONMENTS(*NewEnvs*)

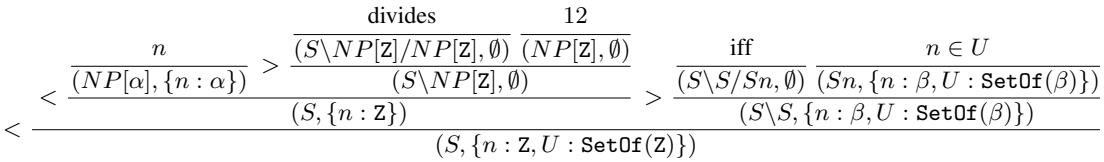


Figure 6: CCG parsing with type environment

Dataset	Correct	Time-out	Wrong	No RCF	Parse failure
DEV	27.6%	10.9%	12.1%	12.1%	37.4%
TEST	11.4%	1.8%	11.4%	6.8%	68.6%

(Correct: correct answer; Timeout: reasoning did not finish in 10 min; Wrong: wrong answer; No RCF: no RCF formula was obtained by rewriting the logical form; Parse failure: at least one sentence in the problem did not receive a CCG tree)

Table 6: Result of end-to-end problem solving

Dataset	Dep. train	Parsed Sentences (%) N=1	N=5	N=10	N=20
DEV	News	48.9	69.1	72.6	76.6
	News+Math	70.5	81.6	84.6	86.4
TEST	News	46.6	58.7	61.9	64.7
	News+Math	59.3	65.3	66.9	68.3

Table 7: Fraction of sentences on which a CCG tree was obtained in top N dependency trees

most general σ (most general unifier, mgu).

We associate a type environment with each chart item and refine it through parsing. The type constraints implied in a discourse are accumulated in the environment and block the generation of incoherent derivations (Algorithm 1). Fig. 6 presents an example of a parsing result, in which the type constraints implied in the two clauses are unified at the root and the type of U is determined. When we apply a combinatory rule, we first check if the environments of the child chart items are unifiable. If so, we put the unified environment in the parent item and apply the unifier to the `type` features in the parent category. For instance, the forward application rule is revised as follows:

$$(X/Y, \Gamma_1) + (Y, \Gamma_2) \rightarrow (X\sigma, \Gamma_1 \sqcup \Gamma_2),$$

where σ is the mgu of Γ_1 and Γ_2 and $X\sigma$ means the application of σ to the `type` features in X.[5]

[5] To be precise, we also consider the type constraints induced through the unification of the *categories*. It can be seen in the derivation step for "n divides 12" in Fig. 6, where the new constraint n :Z is induced by the unification of NP[α] and NP[Z] and merged into the environment of the parent.

9 Experiments and Analysis

This section presents the overall performance of the current end-to-end system and demonstrates the effectiveness of the proposed parsing techniques. We also present an analysis of the failures.

Table 6 presents the result of end-to-end problem solving on the UNIV data. It shows the failure in the semantic parsing is a major bottleneck in the current system. Since a problem in UNIV includes more than three sentences on average, parsing a whole problem is quite a high bar for a semantic parser. It is however necessary to solve it by the nature of the task. Once a problem-level logical form was produced, the system yielded a correct solution for 44% of such problems in DEV and 36% in TEST.

Table 7 lists the fraction of the sentences on which the two-step parser produced a CCG tree within top-N dependency trees. We compared the results obtained with the dependency parser trained only on a news corpus (News) (Kurohashi and Nagao, 2003), which is annotated with bunsetsu level dependencies, and that trained additionally with a math problem corpus consisting of 6,000 sentences[6] (News+Math). The math problem corpus was developed according to the same annotation guideline for the news corpus. The attachment accuracy of the dependency parser was 84% on math problem text when trained only on the news corpus but improved to 94% by the addition of the math problem corpus. The performance gain by increasing N is more evident in the results with the News parser than that with the News+Math parser. It suggests the grammar properly rejected wrong dependency trees, which were ranked higher by the News parser. The effect of the additional training is very large at small Ns and still significant at $N = 20$. It means that we successfully boosted both the speed and the success rate of CCG parsing only with the shallow dependency annotation on in-domain data.

[6] No overlap with DEV and TEST sections of UNIV.

Dataset	Parsing w/ type env.	Typing failure (%)	Correct answer (%)
DEV	no	9.8%	21.8%
	yes	0.6%	27.6%
TEST	no	8.6%	8.6%
	yes	0.0%	11.4%

Table 8: Effect of parsing with type environment

Freq.	Reason for the parse failures (on TEST-2007)
17	Unknown usage of known content words
9	Unknown content words
8	Errors in coreference resolution
4	Missing math expression interpretaions
3	Unknown usage of known function words
3	Unknown function words
2	No correct dependency tree in 20-best

Table 9: Reasons for the parse failures

	Dataset	Precision	Recall
sentence-level	DEV-1999	83% (64/77)	72% (64/ 89)
	TEST-2007	88% (64/73)	56% (64/114)
problem-level	DEV-1999	75% (18/24)	45% (18/40)
	TEST-2007	50% (8/16)	15% (8/53)

Table 10: Accuracy of logical forms

Error type	DEV-1999	TEST-2007
Bind a variable or leave it free	6	2
Wrong math expr. interpretaion	6	1
Quantifier choice	0	3
Quantifier scope	1	1
Logical connective choice	1	1
Logical connective scope	1	0
Others	1	2

Table 11: Types of errors in the logical forms

Table 8 shows the effect of CCG parsing with type environments. The column headed 'Typing failure' is the fraction of the problems on which no logical form was obtained due to typing failure. Parsing with type environment eliminated almost all such failures and significantly improved the number of correct answers. The remaining type failure was due to beam thresholding where a necessary derivation fell out of the beam.

Table 9 lists the reasons for the parse failures on 1/4 of the TEST section (the problems taken from exams on 2007). In the table, "unknown usage" means a missing lexical item for a word already in the lexicon. "Unknown word" means no lexical item was defined for the word. Collecting unknown usages (especially that of a function word) is much harder than just compiling a list of words. Our experience in the lexicon development tells us that once we find a usage example, in the large majority of the cases, it is not difficult to write down its syntactic category and semantic function. Table 9 suggests that we can efficiently detect and collect unknown word usages through parsing failures on a large raw corpus of math problems.

Table 10 presents the accuracy of the sentence- and problem-level logical forms produced on the year 1999 subset of DEV and the year 2007 subset of TEST. Although the recall on the unseen test data is not as high as we hope, the high precision of the sentence-level logical forms is encouraging.

Table 11 provides the counts of the error types found in the wrong sentence-level logical forms produced on DEV-1999 and TEST-2007. It reveals the majority of the errors are related to the choice of quantifier ($\exists$, $\forall$, or free) and logical operators (e.g., $\rightarrow$ vs. $\leftrightarrow$) as well as the determination of their scopes. Meanwhile, we did not find an error related to the predicate-argument structure of a logical form. This fact and the results in Table 6 suggest that the selectional restrictions, encoded in the lexicon, properly rejected nonsensical predicate-argument relations. Our next step is to introduce a more sophisticated disambiguation model on top of the grammar, enjoying the properly confined search space.

10 Conclusion

We have explained why the task of end-to-end math problem solving matters for a practical theory of natural language semantics and introduced the semantic parsing of pre-university math problems as a novel benchmark. The statistics of the benchmark data revealed that it includes far more complex semantic structures than the other benchmarks. We also presented an overview of an end-to-end problem solving system and described two parsing techniques motivated by the scarcity of the annotated data and the need for the type coherency of the analysis. Experimental results demonstrated the effectiveness of the proposed techniques and showed the accuracy of the sentence-level logical form was 88% precision and 56% recall. Our future work includes the expansion of the lexicon with the aid of the semantic parser and the development of a disambiguation model for the binding and scoping structures.

References

Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. 2006. *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Addison-Wesley Longman Publishing Co., Inc.

Daisuke Bekki. 2010. *Nihongo-bunpou no keishikiriron (in Japanese)*. Kuroshio Shuppan.

Daniel Gureasko Bobrow. 1964. *Natural language input for a computer problem solving system*. Ph.D. thesis, Massachusetts Institute of Technology.

Eugene Charniak. 1969. Computer solution of calculus word problems. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*. San Francisco, CA, USA, pages 303–316. http://dl.acm.org/citation.cfm?id=1624562.1624593.

Donald Davidson. 1967. Truth and meaning. *Synthese* 17(1):304–323.

Jan Van Eijck and Martin Stokhof. 2006. The gamut of dynamic logic. In *Handbook of the History of Logic, Volume 6 Logic and the Modalities in the Twentieth Century*, Elsevier, pages 499–600.

Gottlob Frege. 1892. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik* 100:25–50.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 523–533. http://aclweb.org/anthology/D/D14/D14-1058.pdf.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics* 3:585–597. https://transacl.org/ojs/index.php/tacl/article/view/692.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *CoNLL 2002: Proceedings of the 6th Conference on Natural Language Learning 2002 (COLING 2002 Post-Conference Workshops)*. pages 63–69. http://aclweb.org/anthology/W/W02/W02-2016.pdf.

Sadao Kurohashi and Makoto Nagao. 2003. *Building A Japanese Parsed Corpus*, Springer Netherlands, Dordrecht, pages 249–260.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 271–281. http://www.aclweb.org/anthology/P14-1026.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. pages 1223–1233. http://dl.acm.org/citation.cfm?id=1870658.1870777.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 1512–1523. http://dl.acm.org/citation.cfm?id=2145432.2145593.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 590–599. http://www.aclweb.org/anthology/P11-1060.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 2144–2153. http://www.aclweb.org/anthology/P16-1202.

Richard Montague. 1970a. English as a formal language. In Bruno Visentini, editor, *Linguaggi nella Societa e nella Tecnica*, Edizioni di Communità, pages 189–224.

Richard Montague. 1970b. Universal grammar. *Theoria* 36(3):373–398. https://doi.org/10.1111/j.1755-2567.1970.tb00434.x.

Richard Montague. 1973. The proper treatment of quantification in ordinary english. In Patrick Suppes, Julius Moravcsik, and Jaakko Hintikka, editors, *Approaches to Natural Language*, Dordrecht, pages 221–242.

Benjamin C. Pierce. 2002. *Types and Programming Languages*. MIT Press.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1743–1752. http://aclweb.org/anthology/D15-1202.

Bertrand Russell. 1905. On denoting. *Mind* 14(56):479–493. http://www.jstor.org/stable/2248381.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1466–1476. http://aclweb.org/anthology/D15-1171.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1132–1142. http://aclweb.org/anthology/D15-1135.

Mark Steedman. 2001. *The Syntactic Process*. Bradford Books. MIT Press.

Mark Steedman. 2012. *Taking Scope - The Natural Semantics of Quantifiers*. MIT Press. http://mitpress.mit.edu/books/taking-scope.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*. pages 466–477. http://www.cs.utexas.edu/users/ailab/?tang:ecml01.

Alfred Tarski. 1936. The concept of truth in formalized languages. In A. Tarski, editor, *Logic, Semantics, Metamathematics*, Oxford University Press, pages 152–278.

Alfred Tarski. 1944. The semantic conception of truth: and the foundations of semantics. *Philosophy and Phenomenological Research* 4(3):341–376. http://www.jstor.org/stable/2102968.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 297–306. https://aclweb.org/anthology/D16-1029.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 678–687. http://www.aclweb.org/anthology/D07-1071.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*. pages 658–666.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 817–822. http://aclweb.org/anthology/D15-1096.

Author Index